Enterprise Knowledge Infrastructures

Ronald Maier · Thomas Hädrich
René Peinl

Enterprise Knowledge Infrastructures

With 68 Figures
and 50 Tables

Springer

Professor Dr. Ronald Maier
Dipl.-Wirtsch.-Inf. Thomas Hädrich
Dipl.-Wirtsch.-Inf. René Peinl
Martin-Luther-University Halle-Wittenberg
School of Business and Economics
Department of Management Information Systems
D-06099 Halle (Saale)
Germany
E-mail: maier@wiwi.uni-halle.de
E-mail: haedrich@wiwi.uni-halle.de
E-mail: peinl@wiwi.uni-halle.de

Cataloging-in-Publication Data
Library of Congress Control Number: 2005923263

ISBN 3-540-23915-4 Springer Berlin Heidelberg New York

Springer is a part of Springer Science+Business Media
springeronline.com

Printed in Germany

Cover design: Erich Kirchner
Production: Helmut Petri
Printing: Strauss Offsetdruck

SPIN 11357070 Printed on acid-free paper – 42/3153 – 5 4 3 2 1 0

Preface

Both, academics and practitioners alike have spent considerable efforts during the last years to establish ICT support for the handling of knowledge, an idea that is almost as old as the field of computer science. Not surprisingly, the solution is still not there and many businesses trying to implement these technologies have been frustrated by the fact that the technologies certainly could not live up to the overly high expectations. However, there are still numerous projects in organizations that try to tackle the fundamental challenge of how to increase productivity of knowledge work. People do not believe in quick solutions to this problem any more - and they are right. Knowledge management is dead. Long live knowledge management!

Central hypothesis of this book is that the implementation of KM technology in organizations has entered a new stage. In the last years, many vendors jumped on the bandwagon and insisted that their products had "knowledge management technology inside". More recently, however, it seems that many technologies provided by avantgarde systems to support handling of (documented) knowledge, finding of, collaboration between and learning by people doing knowledge work, were weaved into the enterprise infrastructure implemented in many organizations. It is not anymore the quest for the best individual tool targeting a specific KM problem that organizations should engage in. Organizations should strive for improving their information and communication infrastructures so that they are able to handle semantic descriptions of integrated, semi-structured data and offer advanced knowledge services on top of them.

Within this field, the book combines a thorough treatment of the vision of an ideal enterprise knowledge infrastructure on the one hand with a comprehensive description of concepts, standards, tools and systems that are already available and can help to implement this vision on the other hand. We hope that the book will help you to understand the complex matter, that you will enjoy the ideas presented here, discuss them in teams and communities, gain new insights by answering the questions and exercises and that you will be motivated to develop them further. Additional support and contents can be found at the supporting Web site. Please visit: http://www.wiwi.uni-halle.de/maier/EKI/. You can also contact us by email. Any comments are most welcome at: [maier, haedrich, peinl]@wiwi.uni-halle.de!

The book presents the results of the development of courses and programs in knowledge management (systems) for the University of Regensburg, Danube-University of Krems, Austria, and Martin-Luther-University of Halle-Wittenberg. In the last two years, the authors have jointly developed five courses that together present and train to use the concepts in this book at the Department of Management Information Systems, Information Systems Leadership of the Martin-Luther-University of Halle-Wittenberg. During this period we also established a basic knowledge infrastructure at our Department that helped us to exchange ideas and step-by-step develop the concepts that are now part of this book.

Many people have contributed to our thoughts on enterprise knowledge infrastructures. We would like to thank our students for sharing their experiences gained in many organizations implementing KM technologies who inspired us to come up with a book that consis-

tently presents the material scattered across a large number of sources, our teaching assistants for drawing some of the figures, for their support in implementing some of the tools and for numerous remarks on the material and last but not least all colleagues at our Department as well as our friends working in other Departments spread all over the world for many fruitful discussions and proofreading of the manuscript.

Ronald Maier
Thomas Hädrich
René Peinl

Halle / Saale, February 2005

Contents

1 Foundation

Knowledge

Information and communication technologies (ICT) to support the handling of *knowledge* in organizations have been discussed for quite a long time. Back in the 50s to 80s of the last century, various waves of systems applying artificial intelligence technologies had a powerful impact on the conceptualization of knowledge, not only in the discipline computer science, but also in fields, such as management science, organization science or psychology. However, many business organizations trying to implement these technologies, first advertised as the advent of the general problem solving machine, were frustrated by the fact that the technologies certainly could not live up to the overly high expectations. Instead, they showed comparably high complexity and difficulties in applying them to business challenges. Thus, artificial intelligence technologies survived only in special and narrow application fields.

Knowledge work

In the 90s, after a period of high attention to the increase of efficiency, organizations were faced with the transformation of the society into a *knowledge society*, of the economy into a *knowledge economy* and its challenges to significantly increase the speed of innovation and improve the way organizations handle (distributed) knowledge. For those countries that have not (any more) the possibility to exploit some form of natural resources, it is knowledge that creates wealth. Today, the world's wealthiest person, Bill Gates, has developed a company that primarily is made of knowledge work. Knowledge work requires a high level of skills and expertise from the employees and an organizational design that creates an environment conducive for this type of work.

Knowledge management

Concepts of *knowledge management* (KM) were suggested to meet these challenges, starting with the highly innovative work by authors such as Davenport, Nonaka, Sveiby or Wiig, just to name a few. Many authors from a variety of disciplines created, applied and reflected a number of approaches, concepts, methods, tools and strategies for knowledge management. In its short history, knowledge management has absorbed a wide array of research questions which made it interesting and attractive for a large community as diverse as its authors with backgrounds in psychology, organization science, management science or computer science. At the same time, however, the field of knowledge management struggles with the large number of terms that are used differently, the approaches that are incommensurable and its lack of applicability in a business context. More recently, however, a number of instruments have emerged as state-of-the-art of KM practice. Examples are competence management, community management, knowledge maps or semantic content management (section 1.4, 39ff).

Knowledge management systems

Backed by tremendous interest in KM, both, in the academic field and in business practice, vendors of information and communication systems as well as researchers in the field of computer science and management information systems showed prototypes, tools and systems to support knowledge management: knowledge management systems (KMS). This term, however, was a misnomer. On the one hand, knowledge in many definitions as used in the discipline management information systems is either bound to people or extracted from an expert and made available in specially designed systems, so-called knowledge-based systems. On the other hand, management is a term that denotes the software-supported handling, e.g., storing, administering, updating and retrieving of (business) objects when used in connection with information and communication technology (ICT). Examples are data base management systems or document management systems. However, strictly speaking, knowledge management systems neither contain knowledge nor do they manage it.

The term KMS has been a strong metaphor for the development of a new breed of ICT systems. In this view, KMS combine, integrate and extend a number of heterogeneous ICT. Examples are AI technologies, communication systems, content and document management systems, group support systems, Intranet technologies, learning environments, search engines, visualization technologies and workflow management systems. Given the complexity of these technologies, it seems obvious that the development of knowledge management systems is a complex undertaking.

Knowledge infrastructure

In the last years, many vendors have insisted that their products have "knowledge management technology inside". More recently, however, it seems that many technologies provided by the avantgarde systems to support the handling of (documented) knowledge, finding of, collaboration between and learning by people doing knowledge work, were weaved into the enterprise infrastructure implemented in many organizations. Whereas enterprise resource planning systems target the informational representation of business transactions, *enterprise knowledge infrastructures* create an ICT environment for *knowledge work* throughout the organization.

Overview

Chapter 1 provides the foundation for the large number of technologies that can be used to build enterprise knowledge infrastructures. Section 1.1 discusses the term knowledge and distills the specifics that are required to understand its use in connection with the terms management, work and infrastructure. Section 1.2 reflects on the underlying characteristics of the type of work that has to be supported by enterprise knowledge infrastructures called knowledge work. Section 1.3 discusses the most important approaches and concepts of knowledge management. Section 1.4 specifically targets instruments of knowledge management that have emerged from a decade of research on knowledge management and can more or less

readily be applied in organizations. Finally, section 1.6 introduces the key term in this book, enterprise knowledge infrastructure.

Learning objectives

On completion of this chapter, you should be able to

- define the most important knowledge management concepts and approaches,
- identify the many facets that the term knowledge has in different perspectives and analyze the challenges for a systematic management thereof,
- employ the presented framework to classify knowledge along a number of important dimensions,
- compare knowledge management to other approaches in organization science,
- analyze the potentials of knowledge management in organizations,
- appreciate the need for a systematic handling of knowledge to improve productivity of knowledge work,
- identify the changed requirements for the design of information and communication technologies posed by knowledge work,
- describe the state-of-the-art KM instruments that can be applied in organizations,
- define the concept of architecture and discuss its benefits,
- distinguish types and alternatives of architectures in organizations,
- explain the concept of enterprise knowledge infrastructure and relate it to the broader concept of an organization's information and communication landscape,
- identify layers of enterprise knowledge infrastructures.

1.1 Knowledge

Roots in philosophy

The importance of knowledge for societies in general and organizations in particular is rarely questioned and has been studied for a long time. The foundation for Western thinking about knowledge can be traced back to Greek philosophy. However, it is neither intended to give a comprehensive overview of knowledge definitions because even a limited review of the work done in philosophy would fill bookshelves, nor is it intended to give an all-encompassing definition of knowledge.

Organizational perspective

Instead, the most important conceptualizations of knowledge will be reviewed from an organizational perspective (section 1.1.1) which have made their way into various classes of KM approaches. Due to the major role that organizational knowledge plays, there are a number of related

terms that have to be clarified, such as capability, competence, expertise or intellectual capital. Some facets of the term knowledge will be selected to discuss the implications on the definition, the design and the implementation of EKI. Then, the term is defined in the context of enterprise knowledge infrastructures keeping its limitations well in mind (section 1.1.2). Also, important dimensions will be distinguished that help to classify knowledge used in organizations.

1.1.1 Knowledge in Organizational Settings

Knowledge transforms organizations

The transformation of organizations into knowledge-intensive and knowledge-aware organizations takes place at an ever-increasing pace. Knowledge as the key resource, not labor, raw material or capital, changes production functions in organizations significantly. *Knowledge* represents the key concept to explain the increasing velocity of the transformation of social life in general and the way businesses and social institutions work in particular (Drucker 1994). Up to 60% of the gross national product in the United States is supposedly based on information as opposed to physical goods and services (Delphi 1997, 10). This is not surprising as it is estimated that the knowledge-intensive construction and development process of new products and services potentially determines 80 to 90% of the resulting production costs (Scherrer 1999, 131).

Use of the term knowledge

The term *knowledge* is used widely, but often quite vaguely within business administration and MIS in general and within knowledge management in particular. There are a large number of definitions which differ not only between scientific disciplines contributing to KM, but also within these disciplines and consequently also within the KM field. Moreover, the different definitions of the term knowledge lead to different perspectives on organizational knowledge and, thus, to different concepts of interventions into an organization's way of handling knowledge.

Relation to other concepts

Knowledge is related to many other concepts. The most often cited relationships are those to data and information. Figure 1-1 shows the common depiction of the relationships between data, information and knowledge.

Data

Data refers to symbols that are ordered to an elementary description of a person, thing, event, activity, transaction or state in the perceived reality or imagination of persons. Data can be recorded, classified, and stored, but are not organized to convey any specific meaning. Data items can be numeric, alphanumeric, figures, sounds, or images. With respect to ICT, data items are stored in a data base organized for retrieval.

Information

Information is seen in a multitude of ways, but most definitions draw the line between data and information with respect to meaning, the semantics that are commonly assigned to interpreted data, also called information, but not to (raw) data. There are basically two main perspectives:

- Information is data that have been organized so that they have meaning and value to the recipient. The recipient interprets the meaning and draws conclusions and implications.
- Information is the result of a person's interpretation of signals from his or her environment, whereby the result depends on the person's knowledge and the context of the interpretation.

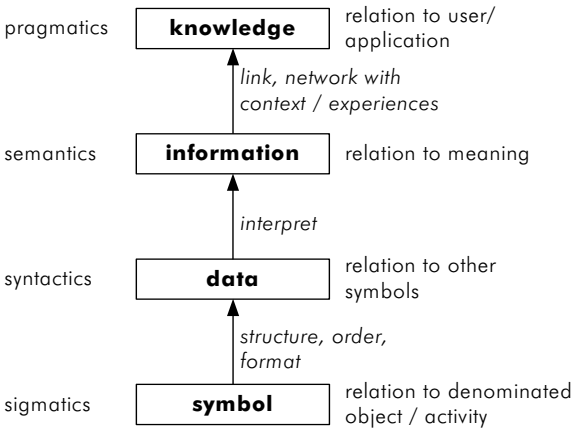

Figure 1-1. Data, information and knowledge as hierarchy of terms

Classes of knowledge

Knowledge is also a key term used in many disciplines and there are many definitions, classifications or categorizations of knowledge. Many classifications use a dichotomy to describe one type of knowledge and its opposite. Table 1-1 presents some examples for important classes of knowledge that are organized with respect to person, organization, context and ICT (Maier 2004, 63f).

Table 1-1. Classification of knowledge

area	dimension	values
context	abstraction	narrative/concrete - scientific/abstract
	generalization	particular - universal
	representation	declarative - procedural
ICT	access	accessible - inaccessible
	medium	electronic - non-electronic
	codability	codable - non-codable
organization	relevance	relevant - irrelevant
	authorization	informal - formal
	security	private - public
	ownership	internal - external
person	value	valuable - not valuable
	awareness	implicit/tacit - articulated/explicit
	support	supported/dominant - unsupported/minority
	existence	knowledge - not knowledge

Agreement about important dimensions

The variety of definitions of the term knowledge is due to the variety of research subjects which require more or less focus on knowledge. At least to some extent, there is agreement among KM researchers about the most important dichotomies and characteristics of knowledge, such as individual versus organizational, implicit versus explicit, organization-internal versus organization-external knowledge (section 1.1.2).

Consequences for KM

In the following, the most important characteristics of knowledge will be summarized which have consequences or provide challenges for the design of enterprise knowledge infrastructures:

"Transfer" of knowledge

Several authors dealing with ICT support for KM have written about KMS which support the transfer or distribution of knowledge. In this area, not only explicit knowledge is considered which can be transferred with the help of knowledge products, but also the tacit side of knowledge. The latter can only be handed on directly from teacher to apprentice (socialization).

Transfer of data vs. transfer of knowledge

According to most definitions of data, information and knowledge *only data can be transported or communicated* which in turn is interpreted by individuals or social systems. Therefore, even knowledge infrastructures essentially contain and support the communication of data, not knowledge. However, the "transfer" or "distribution" of knowledge denotes the simplified and shortened process including interpretation of the message (information) and actualization or extension of knowledge by the receiving sys-

tem. Figure 1-2 shows the complete communication process of data, information and knowledge. Transfer of knowledge implies that the sender is quite certain that the receiver will interpret the data accordingly, (re-) construct the knowledge and use it to actualize the receiver's knowledge in a way that the sender intends.

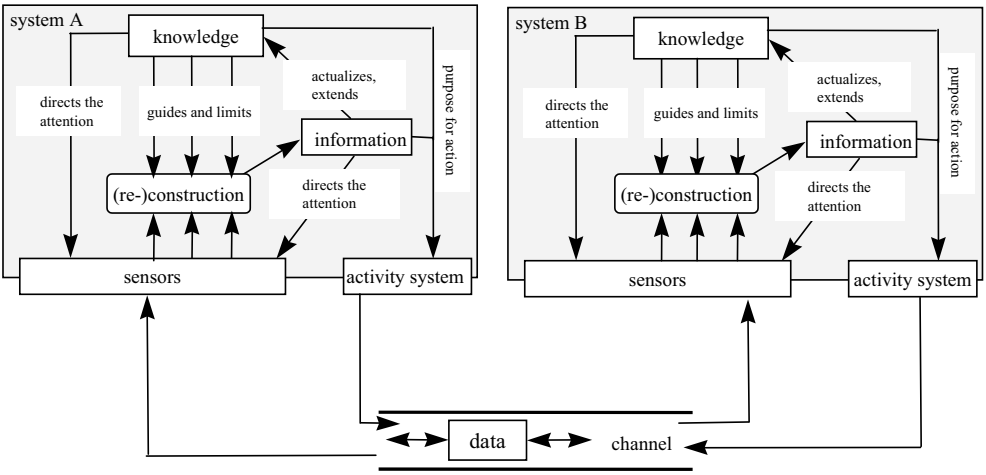

Figure 1-2. The transfer of information and knowledge

Reconstructing knowledge

It must be noted that the sender cannot be sure that the receiver will interpret the data in a way that the sender intended. Additionally, according to modern theories in the cognitive sciences with each transfer of knowledge, the knowledge itself is changed not only at the receiving end, but also at the sending end of the communication as it is not just "retrieved" in memory, but reconstructed and the knowledge's context is thus changed with each transfer.

Relation to context

Knowledge is developed in a cultural context with social, political, economic and ideological dimensions that exert continual forces on both the substance and the process of scientific knowledge creation. What has been said about scientific knowledge creation is all the more true in organizational settings. Organizations are not regularly striving for absolute truth, but for a socially constructed reality that allows for successful organizational actions. Knowledge cannot be separated easily from the social context of its generation and reception.

Economic differences to information

Unlike information, knowledge is not easily transferred between different settings. The costs for the "distribution" of knowledge can be very high. It takes time until individuals take over knowledge. Thus, it requires substantially more effort to implement a systematic management of knowledge transfer as compared to the transfer of information. There are a number of institutions that provide an environment conducive to knowl-

edge transfer or learning. This environment can be viewed as an activity system in which "knowledge seekers", "students" or "apprentices" not only directly learn from "knowledge providers", "teachers" or "masters", but also from participating in a community of practice of all the knowledge seekers and knowledge providers in a joint setting (e.g., schools, Universities, management centers, corporate Universities, industry organizations offering apprenticeships).

Protection of knowledge

Another important challenge in organizations is the protection of valuable knowledge, e.g., against industrial espionage. Examples for measures that prevent the unwanted use of organizational knowledge are classification or property laws and organizational instruments such as incentives, conduct rules or postponing of rewards because a great deal of knowledge valuable to an organization resides with (individual) employees.

In some cases, it is opportune for organizations to share knowledge with competition (coopetition) and thus systematically manage diffusion of otherwise restricted (patented, classified, confidential) knowledge, e.g., through mechanisms such as visiting each other's production facilities, consortia, benchmarking. One implication on the design of EKI is that as valuable knowledge must be protected from leaving the organization unintentionally, it might not be appropriate to make it completely transparent (e.g., to publish it on the organization's Intranet), but instead to disaggregate the knowledge so it cannot be taken easily to a competitor.

Knowledge as product vs. process

Knowledge can be conceptualized as a product or as a process. Both concepts are important, though they have differing implications on the design of EKI. Basically, explicit knowledge can be documented and stored in knowledge repositories whereas (more) implicit knowledge has to be supported indirectly through ICT used to broker and handle communications.

"Right" quantity of knowledge

Many KM approaches implicitly hold the presupposition that *the more knowledge* an organization holds*, the better* for the organization. The application of this simple equation can be dangerous because it does not consider e.g.

- that knowledge that is built in an organization may not be useful,
- that communicating knowledge expects quite a lot from the receiving system (individual or social), namely that the system rebuilds its knowledge structures,
- that knowledge is in a sense provisional and is held until better knowledge is generated,
- that more measurable knowledge in terms of e.g., publications or documents not necessarily means that the organization can act or interpret more intelligently,
- that knowledge increases "not knowledge" which causes the paradox that the more an organization knows, the more knowledge it demands which in turn leads to less efficient daily operations.

As a consequence, EKI have to consider this danger of information overload and inefficient "oversupply" of knowledge. For example, attention has to be paid to e.g., contextualization, filtering, profiling and to determining the optimal portion, level and granularity of knowledge that is presented.

Multi-faceted knowledge

Design and implementation of EKI differ from design and implementation of more traditional application systems. The term knowledge as used here comprises among others valuations, opinions or forecasts, whereas more traditional application systems focus more or less exclusively on hard data. Also, the design of EKI has to consider the multiple electronically available sources of data such as documents, files, messages, contributions in newsgroups, multimedia elements or links to these sources which all might contain useful knowledge once structured, linked and contextualized. Thus, EKI can be combined with an organization's already existing information systems.

Role of knowledge in different types of organizations

Classifications of knowledge can be used to postulate different requirements or perspectives for KM initiatives and supporting ICT. Table 1-2 shows four types of organizations that differ with respect to the focussed type of knowledge and thus require the support of different ICT.

The distinction uses the organizational level from which the primary contributions to the fulfilment of organizational goals is expected (individual versus collective) and whether the focus is on familiar or on novel problems. Empirical analysis suggests trends that organizations are transformed from type I, II and III into type IV organizations.

Table 1-2. Organizations according to types of knowledge (Blackler 1995, 1030)

	Type I: expert-dependent	**Type II: knowledge-routinized**	**Type III: symbolic-analyst-dependent**	**Type IV: communication-intensive**
organizational level	focus on individual	focus on collective	focus on individual	focus on collective
type of problems	familiar problems	familiar problems	novel problems	novel problems
type of knowledge	embodied competencies of key members	knowledge embedded in technologies, rules and procedures	embrained skills of key members	encultured knowledge and collective understanding
characterization	performance of specialist experts is crucial; status and power from professional reputation	capital, technology or labor-intensive; hierarchical division of labor and control	entrepreneurial problem solving; status and power from creative achievements	key processes: communication, collaboration, empowerment through integration
example	professional bureaucracy, e.g., hospital	machine bureaucracy, e.g., traditional factory	knowledge-intensive firm, e.g., software house	adhocracy, innovation-mediated production
role of ICT	computer displacement of action skills	computer integrated work systems	information support and XPS design	development of cooperation systems (CSCW)

Strategic aspects of knowledge

In the context of management science and the strategic perspective of knowledge, a large number of related concepts can be distinguished that stress the importance of knowledge as an organizational resource. It is well worth to briefly review these concepts and their theoretical basis because the distinctive definitions of knowledge (and related concepts) help to understand the different perspectives taken in the literature and also allow for a characterization of KM approaches.

Resource-based view

The theory underlying this perspective is called the resource-based view and builds on ideas presented in the theory of the growth of the firm (Penrose 1959, Wernerfelt 1984). Central idea of the resource-based view is that an organization's success is determined by the existence of organization-specific unique resources. As opposed to the market-based view (Learned et al. 1965, Porter 1980), competitive advantages are not due to superior positioning of an organization in an industry, but due to superior quality of resources or a superior use of the organizational resources. Heterogeneity of resources in different organizations which enables sustained competitive advantages is determined by the individual historical developments of the organization, the development of specific material and immaterial resources, the creation of complex organizational routines which in

turn causes specific historical trajectories and lead to unique idiosyncratic combinations of resources in organizations.

Another central hypothesis of the resource-based view is that in an uncertain and dynamic competitive environment, products and services demanded in the market change quickly, whereas resources and capabilities are more enduring. As a consequence, proponents of the resource-based view suggest to base a strategy on resources and capabilities rather than on product-market combinations as suggested in the market-based view. Resources are seen as platforms for the development of varying products and services.

Knowledge-based view

A more focused version of the resource-based view is called the knowledge-based view, stresses the importance of knowledge as an organizational resource and plays a role in embedding knowledge management into corporate strategy.

Terms stressing strategic relevance

In order to avoid confusion with the traditional view on the term resource and stress the strategic relevance of organization-internal assets, several terms have been proposed which are discussed in the following.

Organizational resource. A firm's resources at a given time could be defined as those (tangible and intangible) assets which are tied semi-permanently to the firm. This organization-specific element is what distinguishes resources in the resource-based view from the traditional viewpoint in economics or business administration with its primary production factors land, labor and capital. Resources in the resource-based view typically have to be built and cannot be bought.

Classification of resources

Organization-specific resources can be classified in a multitude of ways similar to knowledge (see Table 1-1 on page 6). Figure 1-3 presents a typical classification of resources with some examples that give an indication of what is meant by the terms. *Tangible resources* are detailed in financial and physical resources. *Intangible resources* are classified into person-dependent and person-independent ones. *Person-independent resources* are further divided into intangible and organizational assets. *Intangible assets* have a relationship to the organization's environment because they are either legally secured (e.g., patents, intellectual property), or refer to business partners (e.g., networks, customer relationships, reputation). *Organizational assets* refer to the organization's culture (e.g., willingness to share knowledge, perception of service and quality) and routines (e.g., learning cycles, managerial systems) and do not have a direct relationship to the organization's environment.

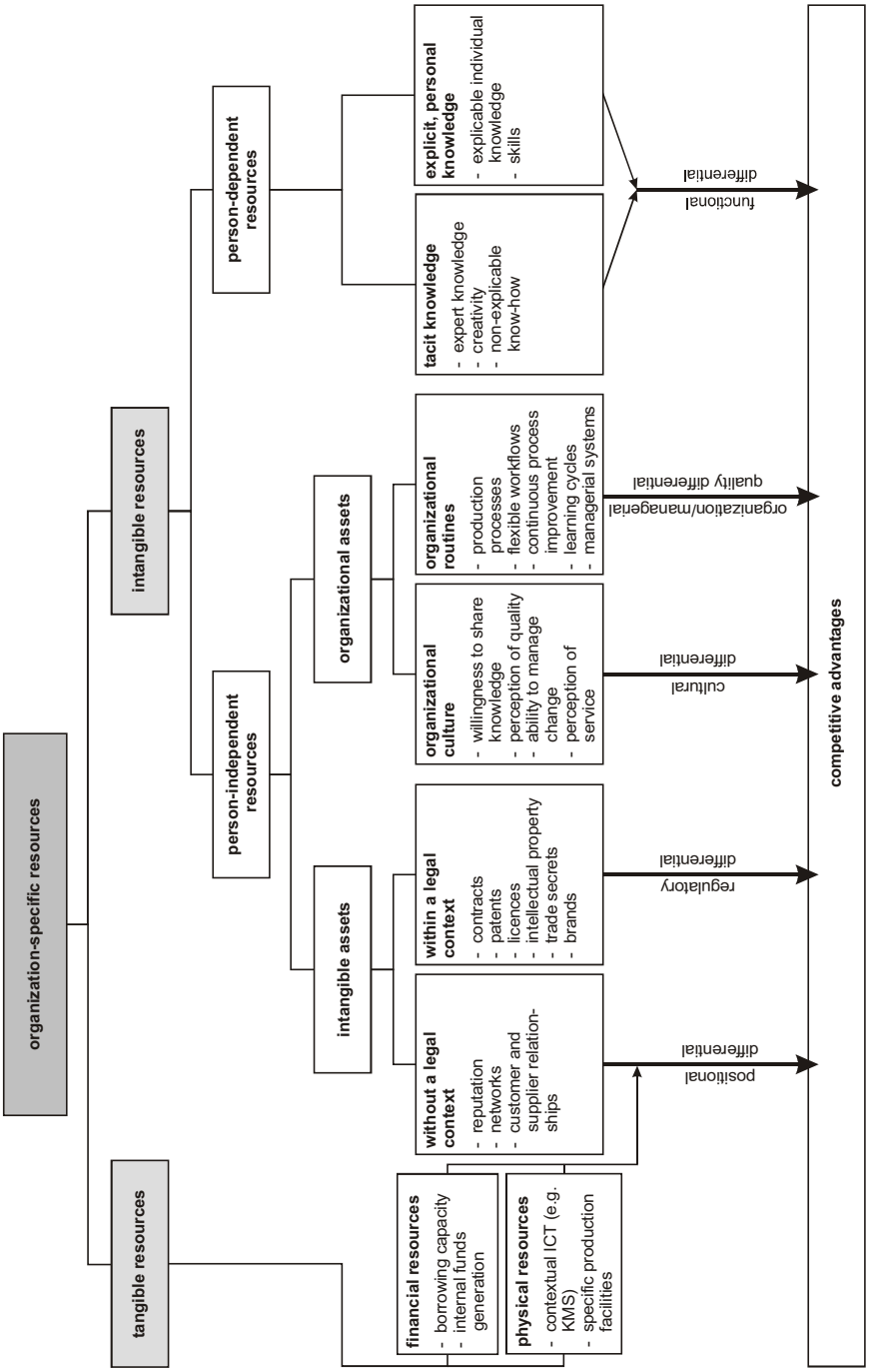

Figure 1-3. Classification of resources (Maier 2004, 93)

Classes overlap

The detailed classes overlap to some extent, especially with respect to the dimension person-dependency as e.g., the smooth functioning of networks (classified here as person-independent) certainly depends on the contacts of individual employees. Their combination is termed an organizational capability.

Organizational capabilities. Capabilities are seen as integrated combinations, consolidations or applications of resources in an organizational context, as teams of resources working together or an interconnected set of knowledge collections — a tightly coupled system. In recent years, some authors pointed out that in situations of quickly changing complex environments, dynamic capabilities are crucial. Dynamic capabilities are defined as the firm's ability to integrate, build, and reconfigure internal and external competencies to address rapidly changing environments.

Capability differentials

Figure 1-3 also shows that the value of (sets of) organizational resources has to be determined in relation to the competition. A comparison reveals so-called capability differentials. Five types of capability differentials can be distinguished:

- *functional/business system differentials*: result from the knowledge, skills and experience of employees and others in the value chain, e.g., suppliers, distributors, lawyers, agents working for the organization etc.,
- *cultural differentials*: applies to the organizational culture as a whole; however, organizational routines are considered as functional differentials because they are transparent and subject to systematic and intended change as opposed to the organizational culture. Cultural differentials are closely related to
- *organization or managerial quality differentials*: result from an organization's ability to consistently innovate and adapt more quickly and effectively than its competitors. As it is probably easier to systematically influence the quality of managerial systems than it is to influence the organizational culture, managerial systems might constitute a factor that can be distinguished from cultural differentials,
- *positional differentials*: are a consequence of past actions which build reputation with business partners, especially customers,
- *regulatory/legal differentials*: result from governments limiting competitors to perform certain activities. Regulatory differentials thus are based on those resources that are legally secured, such as patents, contracts, licences, trade secrets.

To sum up, resources are the basis for capability differentials. Capability differentials provide competitive advantages which can be leveraged in order to produce superior products and services.

Depreciation of knowledge

As organizational capabilities are always determined with respect to competitors, it is only a certain time frame during which certain knowl-

edge can generate competitive advantages. The speed with which innovations penetrate whole industries creates the need to reduce the cycle-time with which new processes and technologies are applied in an organization and turned into new products or services.

Characteristics of strategic resources

Strategic capabilities or (core) competencies. Capabilities and competencies are often used synonymously. However, competencies are often focussed on knowledge as the underlying resource and are directly related to an organization's strategic choices. Organizational competencies are based on a combination or integration of the (individual and common or organizational) knowledge in an organization. In order to be *strategically relevant* and capable of generating sustained competitive advantages, resources must have the following characteristics:

Scarce

Resources must be rare, otherwise competitors can access them easily and no competitive advantage can be gained from their use.

Competitively superior

Resources must either enable organizations to create value for their customers, thus contributing significantly to the perceived customer benefits or to substantially improve effectiveness and efficiency of the organization's processes. Additionally, the value of a resource depends on the relative advantage it bears when compared to the competition.

Multi-purposeful

Resources must provide potential access to a variety of markets. In other words, resources must be applicable in a multitude of products and services and a multitude of markets in order to be of strategic relevance.

Non- or imperfectly imitable

Resources must not be easily replicated in a rival organization. Replication is difficult, e.g., due to *unique historical conditions* in the creation of the resources, *causal ambiguity* (i.e., imperfect information and/or lack of transparency), *social complexity* (i.e., several individuals jointly provide the competitive advantages) or *embedding in organizations* (i.e., several resources can be complexly interrelated and integrated within an organization's routines and/or culture). Thus, there exist so-called barriers to imitation in analogy to the entry or mobility barriers in the market-based view.

Non-substitutable

Resources must not be easily substituted by other resources in order to generate sustained competitive advantages. Thus, a variety of other resources can threaten strategic relevance of a resource.

Non-transferable

A competitive advantage will be the more sustained, the more difficult it is to purchase the resource on the market or to acquire it in cooperation with other organizations. The reasons for a lack of transferability are partly the same as the ones presented for lack of imitability, e.g., the geographical immobility, imperfect information or the fact that resources are firm-specific.

Durable

The longevity of competitive advantages depends upon the rate at which the underlying resources depreciate or become obsolete. Durability varies considerably, e.g., technological resources depreciate quickly due to the increasing pace of technological change whereas reputation and brands are a lot more durable.

Appropriable

Resources must be legally undisputed. Profits from a resource can be subject to bargaining, e.g., with business partners, such as customers, suppliers or distributors, and employees. The more knowledge work is on the rise, the more employees know of their capabilities and negotiate with their employers about the value of their contributions. The more an employee's contribution is clearly identifiable, the more mobile this employee is and the easier his or her capabilities can be transferred to other organizations, the stronger is the employee's position in the negotiations with the organization.

From resources to competitive advantage

The relationship between resources and the more recent concept of organizational capabilities or competencies and in turn their relationship with competitive advantages has been subject to discussion during the last years. Figure 1-4 depicts a framework which shows the chain of arguments used in the resource-based view. A consequent management of the organizational resources thus has to handle the identification, selection, development, synergistic connection, transformation and retention of organizational resources and their integration into capabilities.

According to the knowledge-based view, competitive advantage of an organization depends on how successful it is in exploiting, applying and integrating its existing capabilities and in exploring and building new capabilities that can be applied to the market.

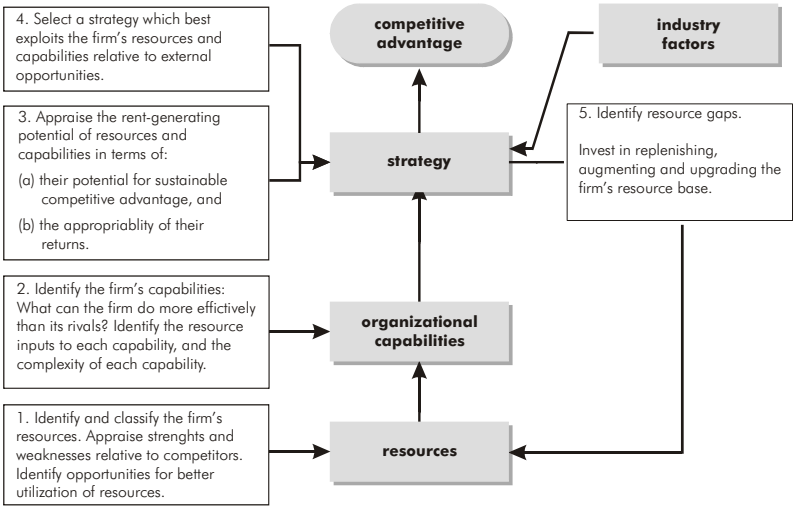

Figure 1-4. Relationship between resources, capabilities, competitive advantages and strategy (Grant 1991, 115, 1998, 113)

Expertise. Experiences or expertise are concepts that refer to an individual's knowledge base rather than the organizational connotation of terms such as capability or competence. Research on expertise has a long tradi-

tion in psychology and pedagogy. However, there is a model on the development of expertise well-received in the literature. The model describes the development of expertise as applied to unstructured situations for which there is no set of facts and factors which fully determine the problem, the possible actions and the goal of the activity (e.g., patient care, business forecasts, social interactions). It stresses the importance of implicit knowledge for expert problem solving.

From novices to experts

In the step-wise course of becoming an expert thinking is reorganized qualitatively which means that expert knowledge is organized differently from explicit knowledge about facts and rules. Teaching means to subsequently lead the learning person from an analytic via a planning to an intuitive way of problem solving. A central concept is "power of judgement" as a holistic way of pattern recognition which is highly adapted to contexts. Adapting a person's organization of knowledge means replacing knowledge about facts and rules with practical cases. The cases are used as patterns to intuitively judge the adequate actions required in a specific situation. The model distinguishes five steps that are briefly described in the following (Dreyfus/Dreyfus 1986, 19ff):

Novice

When observing an expert, *novices* are overwhelmed by the complexity of a situation so that they are unable to imitate an expert. In the first stage of learning, novices are provided with non-situational or context-free *attributes* and *rules*. These do not reflect the total situation, ignore the total context and do not require the novice to understand the total structure of the situation. The novice analyzes a situation by spotting single attributes and selects actions according to the rules remembered. The attributes are not implicitly integrated, but explicitly focused and summed up.

Advanced beginner

The *advanced beginner* has extensive practical experience in the domain. Thus, more context-free attributes can be used in judging the situation and more complex rules determine actions. The most important difference to the novice's problem solving is the use of so-called *aspects*. These are situational or context-specific attributes that the advanced beginner has encountered in a greater number of "similar" practical cases. The selection of actions is not only based on context-free rules, but also on context-specific *guidelines*. However, problem solving is still not integrated as there is no conscious examination of configurations of attributes. The single attributes and aspects are considered as being of equal value. The number of attributes and aspects considered increase to a point where the learner is confronted with an overwhelming number of elements.

Competent

The central skill differentiating the *competent* from the two levels before is the potential to analyze a situation with the help of a perspective. The person is able to plan consciously and thoughtfully. *Goals* and *plans* increase the complexity of the analysis, but reduce the complexity of the situation because not all attributes and aspects have to be considered anymore. Conscious, analytical problem solving is maximized on this level of expertise. Actions are selected with the help of a chosen *perspective*. As a

consequence of the subjective selection of a plan, competent employees will feel responsible for their actions (emotional involvement). On the two levels before, actions were taken by strictly applying rules and guidelines. Thus, unwanted results could be attributed to inadequate rules or guidelines.

Skillful master

The central new skill of the *skillful master* is the ability to perceive situations as a whole as opposed to observing single attributes and aspects of a situation. This means a holistic recognition of similarities of the current situation with situations the master encountered before. The master has a "mental library" of *typical situations* perceived using a specific perspective. New situations are perceived from a specific perspective without consciously selecting it. The relative importance of attributes and aspects in the problem domain is not analyzed consciously anymore. The situation rather presents itself accentuated to the master who intuitively expects which situations could follow the current situation. Actions are still selected consciously on the basis of *maxims*, heuristic principles relating an action to a configuration of attributes and aspects. The master consciously selects actions that have proven successful in a similar situation.

Expert

Every specific situation the *expert* encounters will automatically trigger *intuitively appropriate action(s)*. Experts not only store perspective-based types of situations, but associations of types of situations with corresponding actions. Situations are grouped so that they require the same decisions and actions. They are stored in such a number that they cannot be verbally described. Thus, the expert does not process atomic facts logically, but perceives holistic similarities between the current situation and situations encountered before without having to study isolated elements. Strategic planning does not occur anymore at stage 5. The expert can handle situations in a goal-oriented way, but without consciously set goals.

Table 1-3 shows the five levels of the model with those elements of problem-solving highlighted which determine the central shifts between the stages.

Experts and novices compared

Experts not only have more profound area-specific knowledge, but also apply so-called schemes to analyze situations which allow them to consider more information quicker than novices. Experts are also quicker in deciding between relevant and irrelevant information due to many automated cognitive processes. This automation might also be disadvantageous, though, if experts experience difficulties to adapt to new problem settings or to accept new and revolutionary ideas. Experts spend more time to analyze the situation in difficult problem settings, are different from novices in their selection of problem solving strategies and are more able to control their cognitive processes than novices.

Table 1-3. Model of the acquisition of expertise[a]

skill level	components	perspective	decision	commitment
1. novice	context-free	none	analytical	detached
2. advanced beginner	*context-free and situational*	none	analytical	detached
3. competent	context-free and situational	*chosen*	analytical	detached understanding/deciding; *involved in outcome*
4. proficient/skillful master	context-free and situational	*experienced*	analytical	*involved understanding*; detached deciding
5. expert	context-free and situational	experienced	*intuitive*	*involved*

a. Dreyfus/Dreyfus (1986, 50)

Consequences for knowledge infrastructures

The differences between experts and novices have substantial consequences for the design of knowledge infrastructures. This is especially true for functions such as *personalization*, system-supported *recommendations* and *collaboration*. Novices not only require a different presentation of knowledge elements than experts. Novices searching knowledge infrastructures for information on whom they could ask for help might need support by intermediates — participants just one or two skill levels above their own, not experts. The effort would be higher for both, for experts to reflect their decisions and for novices to learn from them.

Experts on the other hand might be best "teachers" for workers at the skill level *proficient* and possibly *competent*. The expert's knowledge is best analyzed with the help of story-telling. The expert should report critical situations together with context in which they occurred, subjective assessments of the situations and actions taken. Summing up, tutorials, case-oriented learning, exchanging stories and peer-to-peer learning deserves much more attention than the single-minded focus on experts teaching and answering questions of the rest of the employees.

Intellectual Capital. One of the most prevalent questions in the knowledge management area widely discussed in literature and practice is how to determine the value created and the benefits gained by the application of such efforts. Apart from the traditional measures for firm performance (e.g., ROA, ROE or EVA), several approaches to this problem can be distinguished, e.g., human resource accounting, the balanced scorecard or the intellectual capital approach.

The Intellectual Capital (IC) approach is a general, holistic perspective to the intangible assets — the intellectual capital or knowledge capital — of a company. The fundament is based on the observation that the market value of a company[1] is usually higher than its monetary and non-monetary assets. The intellectual capital comprises the immaterial values which have been created by intellectual activities, e.g., human capital, customer capital, process capital, intellectual property.

Some organizations, the best known probably being Skandia have extended their reports on firm performance to include non-financial indicators, especially indicators of intellectual capital.

Examples for IC instruments

Even though the IC approach provides a sound theoretical basis to determine the value of knowledge in organizations, the corresponding methods of measurement are (so far) pragmatic ones. The more abstract the notion of knowledge is, the harder it is to estimate its value. Still, the approach is used widely. Examples for concrete instruments to measure the IC of organizations are the Intangible Assets Monitor, the Intellectual Capital Navigator, the Skandia Navigator, the Balanced Scorecard as well as single measures assessing intangible assets, such as Tobin's q, the IC-index and the Calculated Intangible Value.

1.1.2 Definition

Keeping the abundance of classifications of knowledge in mind, it is clear that the conceptualizations influence the design of KM initiatives and the implementation of KMS in many ways. Thus, it is probably best to define knowledge broadly and openly and discuss some implications of the term in detail:

Definition of knowledge

> Knowledge comprises (1) all cognitive expectancies (2) that an individual or organizational actor uses (3) to interpret situations and to generate activities, behavior and solutions (4) no matter whether these expectancies are rational or used intentionally.
>
> Cognitive expectancies are (5) observations that have been meaningfully organized, accumulated and embedded in a context (6) through experience, communication, or inference.

Description

(1) Knowledge is a cognitive entity and refers to expectations about future events, acts or states.

(2) Actor is meant here in the sense of an agent. Thus, both individuals such as employees, or people working for customer or partner organizations and social entities such as teams, communities or entire organizations

[1] The market value of a company is usually determined by the capitalization (value of the shares on the stock market) of a company.

might act as knowledge-processing entities[2]. Examples of knowledge are scientific findings and theories, heuristics, rules of thumb, techniques, experiences, opinions, cultural customs and norms, world views.

(3) Actors are always part of a *social context* which influences the processing of knowledge (organization, accumulation and embedding in a context) of the actor and thus both the interpretation and the actions. Put in a nutshell, knowledge can be defined as the *capacity to interpret and act*.

(4) In an organizational setting, knowledge that is used for interpretation and action not necessarily is true and, in the event of tacit knowledge, the knowing person might not even be aware of the actor's knowledge.

Cognitive expectancies

(5) Cognitive expectancies are contextualized information entities and therefore organization with the help of meta-data (meaning), linking with other knowledge elements (accumulation) and consideration of the context of creation and application of knowledge (contextualization) distinguish knowledge from information with respect to the design of knowledge infrastructures as compared to information infrastructures.

(6) These are non-trivial tasks that require experience and expertise of individuals, their communication or, last, but not least, logical inference. All of these processes can be supported by ICT.

Knowledge and enterprise knowledge infrastructures

The definition of knowledge as presented here describes the KM perspective. Figure 1-5 summarizes our discussion and shows four central perspectives on knowledge with respect to organizational settings, media to which knowledge is bound in these perspectives, a selection of seven paired types of knowledge which are used in knowledge processes and in turn are supported by EKI. In the following, the implications of EKI support will be discussed for the various *types of knowledge* and the *medium* to which knowledge is bound.

Types of knowledge. Figure 1-5 shows six types of knowledge which are discussed in the following.

Source

The dimension source distinguishes between *organization-internal* and *organization-external* knowledge. Even though organizational boundaries are increasingly blurry, organization as a legal or social institution remains a focal point for the distinction of internal and external knowledge. Internal knowledge is knowledge that originates from within the organization either from members of the organization or in the form of e.g., organizational routines or documented experiences. Organization-external knowledge is brought into the organization, either personally or in documented form.

---

[2] The term *actor* is preferred to *agent* as in the MIS literature *agent* regularly also refers to computer systems (intelligent agents). The old question whether computers can "think" and thus process and apply knowledge is out of the focus of this book (for a brilliant treatise of this topic see e.g., Dreyfus/Dreyfus 1986).

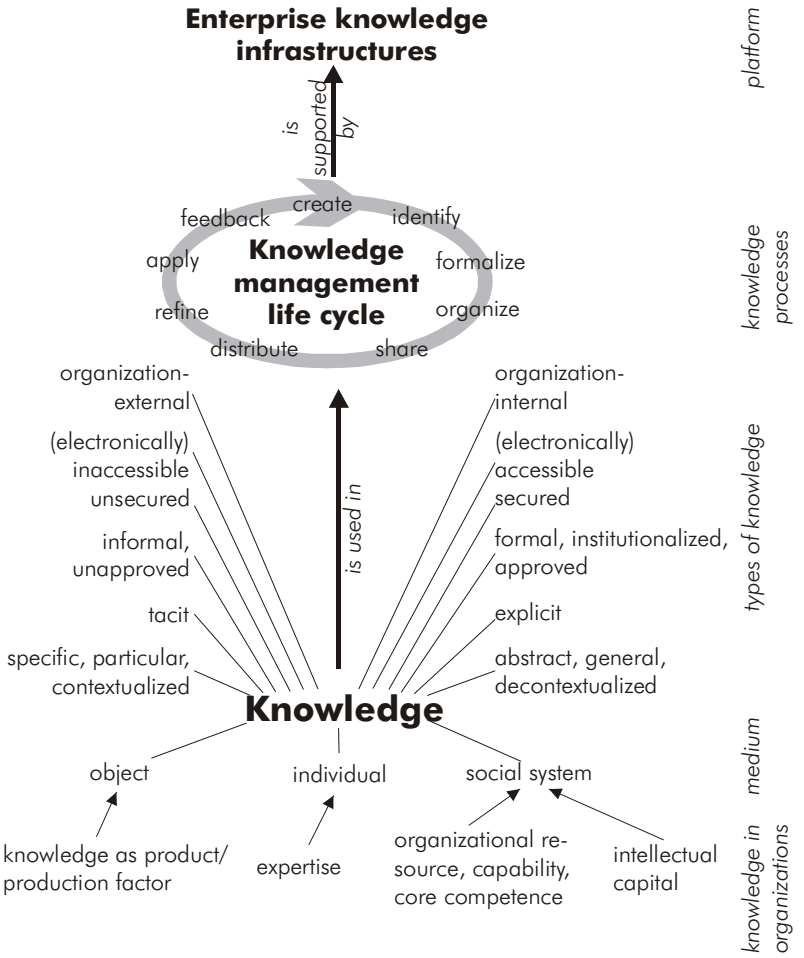

Figure 1-5. Knowledge and its application in knowledge management

Accessibility

This dimension contrasts *electronically accessible* and *electronically inaccessible* knowledge. Knowledge that is published e.g., on an organization's Intranet or in a document management system can be accessed by all members of the organization that have access to these systems whereas documented knowledge that is stored on the individual hard disc of one employee cannot be found by interested knowledge seekers. Additionally, it refers to access to experts that hold knowledge about a specific topic.

Security

The dimension security comprises *secured* and *unsecured* knowledge. Higher visibility of knowledge, experts, networks and structures increases the risk that important knowledge is disseminated to competitors and threatens an organization's competitive advantages. Thus, security is an

important issue at hand. It refers to legal mechanisms such as patents and licenses, copyrights and trade secrets, organizational mechanisms such as incentives to employees, employee conduct rules or job design to secure knowledge and IT measures that prevent unauthorized access to EKI and prevent loss and manipulation of knowledge.

Formality

This dimension ranges from *formal, institutionalized, approved* to *informal, unapproved* knowledge and reflects the degree of institutionalization of knowledge in an organization. Employees develop and apply knowledge independently of the formal approval system and might also share it within their community. This knowledge evolves as a group of employees commits itself to use knowledge in a specified way and is further institutionalized when knowledge is formally approved as part of the standard procedures in the organization. Business organizations rely on rules, roles and (standard operating) procedures, so there is a host of institutionalized knowledge which is applied by the organization's members. The informal part of an organization's knowledge base is rarely well supported and thus needs special treatment in EKI.

Externalization

Externalization turns *tacit knowledge* into *explicit knowledge*. Ever since Polanyi postulated that "we know more than we can tell" (Polanyi 1966), the tacit dimension has been popular. Many authors distinguish between tacit and explicit knowledge, however, Polanyi postulated that *every knowledge* has got a tacit *dimension*. Tacit *knowledge* is subconsciously understood and applied, difficult to articulate, developed from direct experience and handed on through conversation and shared experience (*socialization*, Nonaka 1991, 1994). Explicit knowledge can be formally articulated and shared through e.g., meetings, conversations, mathematical formulas, models or even documents (*combination*). If explicit knowledge is documented, it is removed from its original context of creation or use. EKI can help the receivers of explicit knowledge to reconstruct its context. The process of turning implicit into explicit knowledge is called *externalization*, the reverse process of turning explicit into implicit knowledge is called *internalization*. The distinction between tacit (or sometimes called implicit) and explicit knowledge helps to postulate different KM activities and different systems to support these activities. Figure 1-6 summarizes the processes of knowledge conversion.

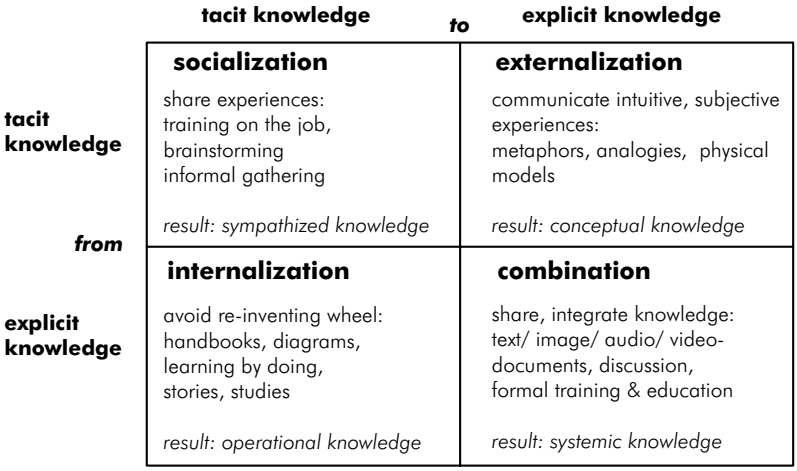

Figure 1-6. Processes of knowledge conversion (Nonaka/Takeuchi 1995)

Generalization

The level of context of knowledge defines a continuum from *specific, particular, contextualized* knowledge describing one particular episode or event e.g., in a story to *abstract knowledge, general, decontextualized knowledge* captured e.g., in a mathematical formula. Before knowledge is distributed to a larger group of people, particular experiences can be generalized to lessons learned e.g., by extracting factors that might have influenced the outcome or by aggregating similar experiences to describe a practice (good or best practice). The more specific a knowledge element is, the more context has to be provided by EKI in order for knowledge seekers to be able to understand, learn and reuse the knowledge element.

Medium. The medium on which knowledge resides can be an *object*, an *individual* or a *social system*.

Knowledge as object

Knowledge as an object stresses the higher value of documented knowledge as opposed to data or (documented) information. Examples are lessons learned, best practices, experience data bases, customized reports or context-enriched documents. In this perspective, knowledge is basically seen as *information plus context*, as networked information. The distinction between information and knowledge is a gradual one, a continuum. The common denominator of this perspective is that (a portion of the) knowledge used in organizations can be explicated and externalized and as a consequence untied from its creator and made available for "easy" reuse by other members of the organization.

Knowledge element

A knowledge unit or *knowledge element* can be viewed as "a formally defined, atomic packet of knowledge content that can be labeled, indexed, stored, retrieved, and manipulated. Examples for knowledge elements are:

concepts, categories and definitions (declarative knowledge), processes, actions and sequences of events (procedural knowledge), rationale for actions or conclusions (causal knowledge), circumstances and intentions of knowledge development and application (specific contextual knowledge).

Individual knowledge

Some authors mix the notion of *knowledge as an object* and *explicit knowledge* although explicit knowledge not necessarily has to be documented. Thus, we have to distinguish between the dimension *relation to individual* with knowledge either being part of an individual's mind or separate as an object and the dimension *explicitness* with knowledge either being implicit and unreflected or knowledge being explicit and thus communicable by the individual. Only explicit knowledge can be documented. Individual knowledge is non-objectified knowledge bound to a single employee. The distinction between explicit and tacit knowledge stresses the importance that employees are not fully aware or cannot fully explicate their knowledge. Examples for categories of individual knowledge include skills, personal experiences and levels of expertise.

Organizational knowledge

In an organizational view, collective knowledge is viewed as knowledge bound to a social system. organizational assets and resources other than individual knowledge are targeted by this approach. Also, the strategic process of selecting, fostering, exploring and exploiting (strategically relevant) organizational capabilities takes place on this level.

Summing up, a well-founded concept of knowledge thus helps to study the possibilities to support the handling of knowledge processes by EKI. The design and implementation of EKI therefore depends on the KM initiative's perspective on knowledge.

1.2 Knowledge Work

Some data on knowledge work

Already in 1997, an estimated 60% of the gross national product in the United States was based on information as opposed to physical goods and services (Delphi 1997, 10). This is not surprising as the knowledge-intensive construction and development process of new products and services potentially determines 80 to 90% of the resulting production costs. There is also a trend towards more complex problem-solving services where the majority of employees are well-educated and creative, self-motivated people. Employees' roles and their relationships to organizations are changed dramatically as so-called knowledge workers replace industrial workers as the largest group of the work force. 60% of US organizations think that between 60% and 100% of their employees are knowledge workers (Delphi 1997, 10) and by the fact that in 2002, about 75% of workers were employed in the service sector in the United States (U.S. Department of

Labor 2003) or about 65% in Germany respectively (Federal Republic of Germany, Common Statistics Portal 2003).

Knowledge changes work

This scenario has been termed the information or *knowledge economy* and has dramatically changed valuation of knowledge work. The concept of knowledge work was coined in order to stress the corresponding changes in work processes, practices and places of employees and thus the differences to traditional (often manual) work.

1.2.1 Definition and Characteristics

Characteristics

Knowledge work is characterized as follows:

- *target*: solves ill-structured problems in complex domains with a high degree of variety and exceptions,
- *content*: is creative work, requires creation, acquisition, application and distribution of knowledge and bases inputs and outputs primarily on data and information,
- *mode of work:* consists of a number of specific practices, such as expressing or extracting experiences, monitoring what can be learned from happenings, translating knowledge to other domains, interpreting and absorbing knowledge and networking with other people,
- *personal skills and abilities*: uses intellectual abilities and specialized knowledge rather than physical abilities and requires a high level of education, training and experiences resulting in skills and expertise,
- *organization*: is often organized decentrally using new organizational metaphors, such as communities of specialized knowledge workers, has strong communication, coordination and cooperation needs and is highly mobile, flexible and distributed,
- *ICT*: requires a strong yet flexible personalized support by information and communication technologies.

Information work

Knowledge work can be defined as work that creates, translates or applies new knowledge. This definition is a rather narrow one so that only a small percentage of actual work being done in organizations would qualify as knowledge work. The broader term, information work, takes into account that not all work with information necessarily generates, translates or applies new knowledge and comprises knowledge work, management work and data (service) work (Drucker 1993, Schultze 2003).

Data/service work

Data or service work relies on established procedures, is well defined and does not require equally high levels of education than in the case of knowledge work.

Management work

Management work is performed by business owners, executives, legislators, senior officials and supervisors whose daily work practices comprise the processing, communication and translation of (abundant) information and the preparation, taking and execution of decisions.

Knowledge work ...

In this narrow view, knowledge work is restricted to (re-)producing new knowledge whereas data (service) work transforms information, but does not produce new knowledge. However, in a wider view knowledge work comprises the creation, acquisition, distribution and application of knowledge. Therefore, it might be difficult to separate knowledge work from "mere" data or service work in actual work practices so that EKI might be most useful when supporting information work in general, not "just" knowledge work.

... as professions, personal characteristics or activities

The term knowledge work refers to (Kelloway/Barling 2000):

- *professions*: occupations or job positions are classified into knowledge workers and non-knowledge workers or routine, manual etc. workers. This distinction is not without trouble because on the one hand all human work requires some kind of knowledge and on the other hand even within one profession actual workers might differ widely according to the share of their work that qualifies as knowledge work.
- *characteristics of a defined group of individuals*: education, training and years of work experience are a necessity for a worker to be called an expert. In this case, knowledge work refers to experts' work. However, on the one hand experts might not always be engaged in knowledge work and on the other hand less experienced employees might be engaged in just the same type of work than experts are requiring just the same organizational and ICT design.
- *activities/behavior in organizations*: Thus, knowledge work should not be restricted to a certain class or group of employees. It should rather be used as a concept that allows a focus on commonalities across professions and positions for the application of KM instruments, KM-oriented organizational design and ICT support. As a higher and higher percentage of employees is engaged in this type of work, the corresponding design of an ICT environment throughout an organization gains importance: an enterprise-wide infrastructure for knowledge work or, shortly, enterprise knowledge infrastructure.

1.2.2 Traditional Work versus Knowledge Work

Environment for knowledge work

Table 1-4 compares the traditional, routine work environment of an office employee with the work environment of a knowledge worker. It shows the changed requirements for the organizational design and the ICT support for knowledge work that have to be considered by a KM initiative and some aspects of economics that affect the management of knowledge work.

Organizational design

When compared to traditional work, knowledge work can be characterized by stronger communication needs, weakly structured and less foreseeable processes, the assignment of multiple roles to one person rather than a single job position per person and the increasing importance of teamwork

in the form of project teams, networks and communities in addition to work groups and departments. These changes are reflected by a decentral organizational design that strengthens the position of decentral units.

The boundaries of an organization are blurry and knowledge workers are engaged in a large number of communication, coordination and cooperation processes and practices that cross the organizational boundaries. Alliances, joint ventures, (virtual) networks and professional communities are some examples for types of institutional settings that have been developed to organize these exchanges.

Table 1-4. Traditional office work versus knowledge work

criterion	**traditional office work**	**knowledge work**
organizational design		
orientation	data-oriented	communication-oriented
boundaries	organization-internal focus	focus across organizational boundaries, alliances, coopetition, (virtual) networks
centralization	central organizational design	decentral organizational design
structure	hierarchy	network, hypertext
process	highly structured, deterministic processes (pre-structured workflows)	ill-structured, less foreseeable processes (ad-hoc workflows)
group	work group, department	project team, network, community
role	one job position per person	multiple roles per person
ICT support		
type of contents	structured data (e.g., tables, quantitative data)	semi-structured data, e.g., links, hypertext documents, container, messaging/learning objects, workflows, skill directories
storage	(relational) data base management system, data warehouse	document/content management systems, experience data bases, newsgroups, mail folders etc.
data handling	coordination of accesses, integrity, control of redundancy	synchronization, information sharing, distribution of messages, search and retrieval
coordination	workflow management system	messaging system, Groupware
modeling	data, business process, workflow	ontology, user profile, communication, activity/work practice

Table 1-4. Traditional office work versus knowledge work

criterion	**traditional office work**	**knowledge work**
workspace	fixed workspace	mobile office (virtual office), multiple workspaces
equipment	personal desktop computer; poor resources	laptop, personal digital assistant, mobile phone; rich resources
applications	small range of applications	wide range of applications, including Web applications (browser, email, explorer, etc.)
connectivity	stand-alone	permanent, fast network connections, mobile devices
economics		
management focus	finance, past orientation, periodic reporting	balanced set, future orientation, instant access
location of value	things	flows
tangibility	tangible	intangible
metrics	production statistics, metrics for reporting	innovation statistics, metrics for managing
standardization	standards; standard products and services	common, yet customized products and services

ICT support

From an ICT perspective, the main changes in the requirements occur due to the considerably higher complexity of data and the focus on organization-wide and inter-organizational communication and mobility of knowledge workers. Storage and handling of semi-structured data require additional ICT systems, e.g., document management systems, experience data bases. Consequently, the challenges in the handling of data are no longer restricted to the provision of integrity, control of redundancy and coordination of accesses as in the relational data base world. New challenges are complex synchronization needs of mobile workspaces, information sharing within and across organizational boundaries as well as search and retrieval in documents and messaging objects that are encoded in a large number of heterogeneous formats for semi-structured data and reside in a variety of data and document sources spread throughout the organization.

Coordination in traditional office work is provided by workflow management systems that implement operative business processes. The lesser structured knowledge work can be coordinated by messaging systems and Groupware. Consequently, modeling used to focus largely on data (entity relationship modeling), objects and classes (object-oriented modeling) and business processes (business process modeling). Knowledge work requires

content- and communication-oriented modeling techniques that define meta-data and provide taxonomies, ontologies, user models, communication diagrams, knowledge maps and diagrams that show what objects, persons, instruments, roles, communities, rules and outcomes are involved in the main knowledge-related activities. Finally, the increased mobility of knowledge workers requires multiple, virtual workspaces that can be personalized according to the demands and practices of their users.

This change in ICT support is backed by a corresponding major shift in the ICT infrastructure. PCs are no longer equipped with weak resources and used in an offline, stand-alone mode. Computers have rich resources, provide information-rich modes of interaction with users, permanent, fast network connections as well as highly flexible wireless and mobile connections and comprehensive communication features. Mobile appliances, such as notebooks, PDAs and mobile phones are equipped with a wide range of applications.

To sum up, this calls for (1) the systematic, flexible handling of context, (2) intelligent functions to handle the vast amounts of substantially extended types of contents, i.e. semi-structured data in the organizational "knowledge base", and (3) extended functionality for collaboration. These functions have to be realized in or seamlessly integrated with the knowledge workers' personal workspaces (section 5.1.2, 316ff).

Economics

Correspondingly, management focus has shifted from a mere periodical financial focus with its past orientation to a flexible and balanced set of criteria that show the current status of the organization's resources, processes, innovation and performance. The interest thus has shifted from tangible to intangible assets, from things to flows, from standards and standard products and services to common yet customized products and services (Skyrme 2000). Metrics are required not simply for reporting the production statistics of goods and services, but to manage the innovation process(es) in the organization. Knowledge management in this realm provides for more visibility of organizational resources, skills and knowledge processes and allows for a more systematic strategic management of (core) competencies in an organization.

Conclusion

Consequently, KM initiatives primarily aim at fostering an organizational and ICT environment that is suited for knowledge work. Knowledge work is the primary target of knowledge management, but corresponding organizational instruments and knowledge infrastructures might also aim at improving information work. This includes management and data or service work. The substantially changed work practices of their largely increased main target group, knowledge workers, together with recent innovations in ICT infrastructure demand a strategic KM initiative. KM not only improves organizational effectiveness, but systematically realizes the potentials of a learning- or a knowledge-intensive organization for creating and sustaining superior competitive positions.

1.3 Knowledge Management

The field of knowledge management has drawn insights, ideas, theories, metaphors and approaches from diverse disciplines. This section briefly reviews the history of knowledge management. The tracing of the roots helps to understand the perspective which knowledge management has or can have on organizations.

1.3.1 Roots of Knowledge Management

Roots of the term KM

The roots of the term knowledge management can be traced back to the late 60s and early 70s in the Anglo-American literature. However, it almost took another 20 years until the term appeared again in the mid 80s in the context as it is still used today (e.g., Sveiby/Lloyd 1987, Wiig 1988). This time it got a tremendous amount of attention.

The underlying concepts used and applied in KM, though, have been around for quite some time. There have been a large number of fields and disciplines dealing with the handling of e.g., knowledge, intelligence, innovation, change, learning or memory in organizations. Figure 1-7 shows sociology, business and ICT as the main lines of development which approaches of knowledge management draw from.

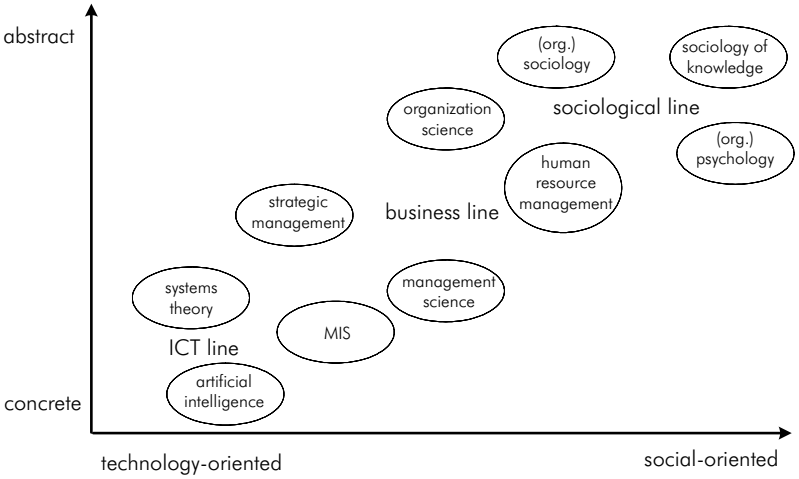

Figure 1-7. Lines of development of knowledge management

Various approaches have played a role in the development of the theories of organizational learning, organizational memory and, ultimately, of knowledge management. Table 1-5 characterizes the most important research fields and disciplines that fuel developments in KM.

Table 1-5. Research fields that form the roots of knowledge management

research field	characterization
organizational change	supports changes within and changes of organizations with development, selection and learning models and thus represents an umbrella term for fields such as organizational development or organizational learning.
organization development (OD)	is a methodical strategy for intervention, initiated through consulting and planned by management with the assistance of a change agent, concerning personal, interpersonal, structural, cultural and technological aspects.
organizational learning (OL)	claims that observable phenomena of change in organizations are connected with unobservable inter-personal processes of learning on a micro-social (group) as well as a macro-social level (organization).
organizational memory (OM)	is capable of storing things perceived, experienced or self-constructed beyond the duration of actual occurrence, and then retrieving them at a later point in time in analogy to an individual's memory.
organizational intelligence	provides a slightly different focus on organizational information processing than OL with an emphasis on collective processing of information and decision making.
organizational culture	is largely an implicit phenomenon only indirectly observable through concepts such as trust, norms, standards, unwritten rules, symbols, artifacts which are the results of learning processes, provide orientation and are shared by the organization's members in a process of socialization.
theories of the evolution of organizations	apply evolution theories originally developed in philosophy, biology and social sciences to organizations, e.g., population-ecology approach, self-organizing systems, organized chaos and evolutionary management.
human resource management	in an institutional sense denotes an organizational subsystem that prepares, makes and implements personnel decisions to secure availability and effectiveness of personnel.
information processing approach	explains individual behavior, e.g., problem solving, decision making, with concepts from cognitive psychology such as attitude, personality, definition of the situation as well as short and long term memory.
systems theory	aims at the formulation of general laws and rules about states and behaviors of systems and provides the basis for many investigations, theories and concepts developed within organization science and MIS.
artificial intelligence (AI)	tries to establish the analogy between human and computer problem solving and applies a common set of methods, e.g., mathematical logics, pattern recognition or search heuristics, to a variety of problem domains.
strategic management	determines long-term goals and positioning of organizations and encompasses the entire process of formulation, implementation and evaluation of strategies to link strategic and operational decision-making.
other management approaches	focus on certain aspects of management, such as innovation management, or provide an alternative view on management, such as systemic or system-oriented management, and evolutionary management.

Table 1-5. Research fields that form the roots of knowledge management

research field	characterization
organizational psychology	studies human behavior and experience in organizations and was later extended to explicitly consider the system characteristics of organizations on the individual, group and organizational levels of abstraction.
organizational sociology	analyzes structural similarities of organizations which are seen as social systems of activity and offers a variety of perspectives and approaches to describe and interpret events and processes in organizations.
sociology of knowledge	views knowledge as socially constructed on the basis of a world view (German: Weltbild) and influenced OL and KM in terminology and conceptualization with theories of social construction of reality.

Organizational knowledge base, learning and memory

However, it is literature and tradition of *organizational learning* and its more recent structural counterpart — *organizational memory* or *organizational knowledge base* — that influenced knowledge management most in the business line of development as well as artificial intelligence in the ICT line of development. Those are briefly reviewed in the following.

Organizational learning (OL). Even though OL has emerged as a field only in the 70s and 80s itself, it soon became a recognized way of looking at change processes in organizations. OL theories and approaches share the common hypothesis that phenomena of change in organizations are connected with collective or inter-personal processes of learning. The definitions of OL differ with respect to the question whether behavioral change is required for learning or whether new ways of thinking and, thus, new possibilities for action, are enough.

Differences to traditional OD

There are differences between traditional organization development (OD) and OL. Change is considered the rule in OL, not the exception as in OD. OL views change as endogenous, as part of the organization's processes, and the (indirect) management of change is considered an organizational competence rather than an (external) expert's competence.

Micro- and macro-structure

OL processes aim at weaving individual knowledge into organizational knowledge and can be classified into *micro-organizational learning* in groups and *macro-organizational learning* on the organizational level. Individual experiences and learning potentials are organizationally connected in groups (micro-structure). The groups' learning results are then turned into organizational learning success (macro-structure). From a management perspective, OL provides concepts, methods and instruments to support organized *collective learning (processes) in organizations*.

Learning organization

The term *learning organization* underlines an organization's skills in performing organizational learning, in more detail, its skills at creating, acquiring, and transferring knowledge, and at modifying its behavior to reflect new knowledge and insights.

Organizational memory (OM). The basic idea of this approach is that learning, whether individual or organizational, is not possible without memory. Memory generally denotes a system capable of storing things perceived, experienced or self-constructed beyond the duration of actual occurrence, and of retrieving them at a later point in time. Using this metaphor, organizational memory is seen as a prerequisite for organizational learning just as individual memory is a prerequisite for individual learning.

Misleading metaphor

As with many metaphors, the analogy between organizational and individual memory is weak and processes on the individual level are entirely different from those on the organizational level. Thus, intuitive understanding of the term is often misleading, e.g., regarding OM as a "brain" to which organizations have access or the more technical interpretation which uses the often cited, but unsuited analogy between computers and brains. OM simply means that employees, written records, or data bases "contain" knowledge that is readily accessible.

Active memory

However, emphasis has shifted from a static to an active interpretation of memory — that parts of the OM that define what an organization pays attention to, how it chooses to act, and what it chooses to remember from its experience. The results are individual and shared mental models as well as the complex phenomena taking place when groups or organizations jointly "process" knowledge. Many approaches have been developed which claim to guide organizations to use their common or shared memory in a more efficient way.

Artificial intelligence (AI). Information and communication technology represents a key enabler for KM initiatives. Consequently, many researchers and practitioners in the field of AI have changed their research focus from expert and knowledge-based systems to knowledge management systems. Together with its psychological sibling, the *cognitive sciences*, the field of artificial intelligence has tried to establish the analogy between human and computer problem solving.

AI technologies for KM

Most AI research institutes nowadays apply AI methods, tools and techniques, e.g., mathematical logics, pattern recognition and search heuristics, to a wide variety of problem domains including KM. Advanced AI technologies, such as formal logics, machine learning, neural networks, genetic algorithms and intelligent agents, are readily available to provide "intelligent" tools for KM, e.g., for semantic text analysis, text mining, user profiling and pattern matching.

KM compared to OL and OM

Knowledge management renews an old promise of a great part of the organization science literature, namely to provide concepts to improve the systematic handling of knowledge in organizations.

Focus on management

KM can be viewed as a *translation* of OL and OM approaches to management terms and an integration with management concepts, such as strategic management, business process management, human resource management and last, but not least information management. The management

focus encourages the *goal-oriented design of interventions into the handling of knowledge, capabilities or (core) competencies* on a strategic, organization-wide level, the redesign of knowledge-intensive business processes, the improvement of personal knowledge management as well as the implementation of enterprise-wide ICT infrastructures for KM.

ICT as enabler

Central to KM is the *use of modern information and communication technologies* as an enabler or a catalyst for organizational instruments. This implies that especially practitioners expect that KM produces expectable, manageable improvements in productivity of knowledge work. The growing management attention about productivity of knowledge work is necessary to implement a profound change in the organizations' organizational and ICT infrastructures: enterprise knowledge infrastructures.

1.3.2 From Data to Knowledge Management

In addition to the interdisciplinary perspective on KM as presented in the last section, there is yet another quite popular conceptualization which compares knowledge management to data management and information (resource) management. In the following, this perspective will be applied to briefly survey the development from the management of data to the management of knowledge.

Many authors seem to agree on some form of a *hierarchical relationship between data, information and knowledge* (section 1.1, 3ff). Each higher level is based on or extends the preceding one. This conceptualization is used to postulate different demands for management (goals, approach, organizational roles, methods, instruments) and different resulting systems (data base systems, data warehouses, information and communication systems, knowledge management systems) on each of these levels.

Historically, in the seventies and the beginning of the eighties the focus certainly was on data management (Figure 1-8). In the following, the steps will be discussed subsequently.

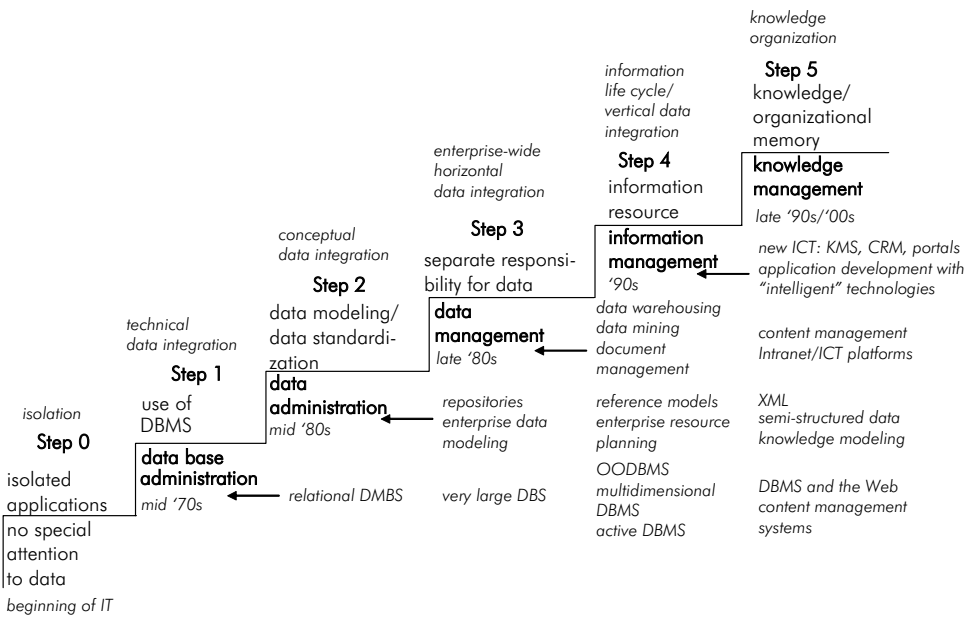

Figure 1-8. Historical development of information processing with focus on data (based on Ortner 1991)

Step 0: isolated applications

The starting point for the historical development of information processing can be described by a joint consideration of program logic and data. There is no special attention being paid to data. Application systems hold their own data storages leading to redundancies and inconsistencies between different application systems.

Step 1: data base administration

In the first step, technical issues therefore mattered most. Data base administration is concerned with the technical integration of previously isolated data storage units. Examples for tasks are to guarantee efficient data storage avoiding or controlling redundancies, to implement and administer the data base management systems (DBMS) that provide instruments for technical integration between application systems or to tune the performance of data base systems.

Step 2: data administration

As DBMS penetrated organizations, semantic or conceptual data integration, data modeling and data handling were the most important questions to be resolved. These tasks together provide semantic data integration which is the primary goal of step 2.

Step 3: data management

Separate organizational units were institutionalized, which were responsible for the co-ordination of data management tasks throughout an organization. Often, this coincided with the development of enterprise data models which were seen as an instrument for the integration of project or departmental data models on an organization-wide level.

With the advent of an organization on a certain step, tasks introduced at a previous step still play a role. For example data base administration on step 1 covers not only hierarchical and network DBMS, but also relational DBMS (step 2), very large DBS (step 3), object-oriented, active and multi-dimensional DBMS in step 4 as well as content management systems and the access of DBMS from the Web (both Internet and Intranet) in step 5 (Figure 1-8).

Step 4: information management

Information was understood as a production factor which had to be managed like other production factors (capital, labor). Thus, the scope of the *information resource management* was a much broader one as compared to data management. The most important aspects were the extension from the management of syntactic and semantic to pragmatic aspects of information understood as an instrument for preparing decisions and actions, information logistics, the contingency approach to information — the different interpretation of information in different situations — and the perspective-based approach to information which means that different groups of users might interpret the same data differently.

Step 5: knowledge management

An organization's ability to learn or handle knowledge processes (process view) or its ability to handle knowledge (product view) have been considered the new key success factor. This has required new organizational design alternatives and also new information and communication systems to support the smooth flow of knowledge which consequently have been called knowledge management systems.

Extended tasks on lower steps

Already existing tasks on lower steps have been once again extended. With the advent of advanced data base and network technologies as well as the availability of sophisticated AI technologies, knowledge management extended the focus of *information management* to the handling of new information and communication technologies as well as to enrich *application development with intelligent technologies* (Figure 1-8).

With respect to data, knowledge management needs to handle networks of semi-structured, context-rich data, experts, participants and their combination. *Data management* has been once again extended to cover *meta-data and content management* for semi-structured data on an enterprise-wide level. This includes design and handling of meta-data for the corresponding new tools and systems. Certainly, EKI cannot be reduced to their data and meta-data structures, but offer a new variety of ways to support the handling of knowledge in organizations.

KM next step in information processing

To sum up, in many organizational contexts and several approaches in the literature, knowledge management is simply viewed as the next consequent step in the development of organizational information processing. Indeed, from a data-oriented perspective, this view can be justified and has its advantages. It explains, for instance, what data management tools and methods, what information logistics and ICT infrastructures are required in order to effectively build EKI.

However, the concepts of knowledge management also require a much broader view which includes organizational functions and processes traditionally not viewed as part of information management (section 1.3.1, 30ff). Both historical roots of KM — the interdisciplinary field of organizational learning and the step model tracing management of knowledge back to management of data and information — have to be considered for a definition of KM.

1.3.3 Definition

Knowledge management is still a young field with multidisciplinary roots. Thus, it is not surprising that there seem to be almost as many definitions to the term than there are approaches or "schools" of authors contributing to the field. Basically, there are two groups of KM approaches, *human* and *technology oriented KM approaches* which basically reflect the origin of the approaches, either in a human/process-oriented organizational learning, organization science background, or in a technological/structural organization science, a MIS or computer science/artificial intelligence background.

There is also agreement that there are more holistic KM conceptualizations which encompass both directions. Figure 1-9 shows the two sides of knowledge management and some examples for concepts developed in holistic approaches aimed at their integration.

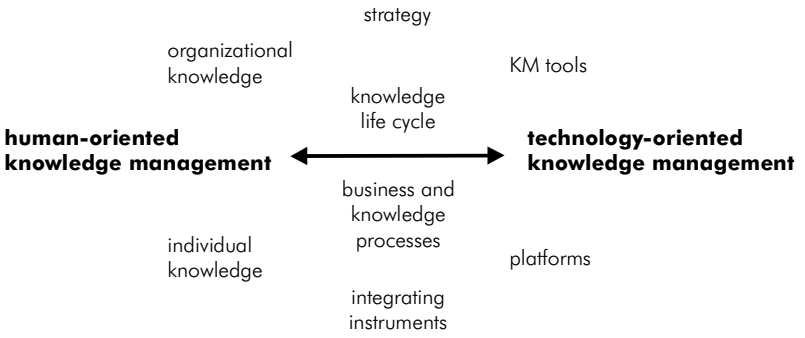

Figure 1-9. Human versus technology-oriented KM and approaches to their integration

The term management is used here in a *functional sense* (managerial functions approach) in order to describe the processes and functions, such as planning, organization, leadership and control in organizations as opposed to the institutional sense (managerial roles approach) which describes the persons or groups that are responsible for management tasks and roles. This leads us to the following definition.

Definition of knowledge management

Knowledge management is defined as the management function responsible for regular (1) selection, implementation and evaluation of knowledge strategies (2) that aim at creating an environment to support work with knowledge (3) internal and external to the organization (4) in order to improve organizational performance.

The implementation of knowledge strategies comprises all (5) person-oriented, product-oriented, organizational and technological instruments (6) suitable to improve the organization-wide level of competencies, education and ability to learn.

Detailed analysis of the definition

(1) Systematic interventions into an organization's knowledge base have to be tied to business strategy. So-called knowledge strategies guide the implementation of a KM initiative and tie it to business strategy. (2) KM creates an infrastructure, both organizational and technological, to improve knowledge work. Enterprise knowledge infrastructures are part of this infrastructure. (3) Knowledge processes are not restricted to the organization's boundaries, but involve cooperation with partners, suppliers and customers. (4) KM mainly aims at improving organizational effectiveness. However, creating, maintaining or distributing intellectual capital results in a higher valuation of an organization.

(5) Depending on the perspective on KM, objects of the implementation of knowledge strategies can be objectified knowledge resources (documented knowledge), people, organizational or social structures and knowledge-related technology (especially ICT). Consequently, the environment of knowledge work that is created or fostered by KM consists of (a) person-oriented, (b) semantic or product oriented, (c) organizational and (d) ICT measures. (6) KM is not exclusively about individual learning. Collective learning is of differing types (e.g., single loop, double loop, deutero learning), takes place on different levels of the organization (e.g., work group or project, community or network, organization, network of organizations) and in different phases (e.g., identification or creation, diffusion, integration, application, feedback). KM aims at improving the organizational competence base as well as the ability to learn.

None of these areas explicitly focuses on the contents, that is the actual subjects, topics or knowledge area(s) around which a KM initiative and finally an EKI builds a supportive environment. The reason for this is that the definition of KM should support all kinds of knowledge areas.

Summing up, there are substantial benefits to be gained from a KM initiative. An implementation of ICT to support a strategically relevant KM initiative not only has to select a KM perspective and often a combination of KM tools and systems, but also integrate KM instruments with the supporting technology.

1.4 Knowledge Management Instruments

As explained in the definition of knowledge management in section 1.3.3, 37ff, the implementation of knowledge strategies requires systematic interventions with the help of instruments, either person-oriented, product-oriented, organizational or ICT instruments. Section 1.4.1 reviews a number of case studies of KM measures to give examples of what actual KM initiatives in organizations aim at and finally gives a definition of the term KM instrument. Section 1.4.2 details the classification of KM instruments already presented as part of the definition of knowledge management (section 1.3.3, 37ff) and describes some central KM instruments according to this classification.

1.4.1 Definition

Even though the terms KM instrument, KM project, KM initiative and KM measure are widely used, there is hardly any concrete definition of any of these terms. A large number of measures has been proposed as part of case studies on KM which also comprise more traditional person-oriented measures well-known in human resource management, e.g., programs for personnel development, content-oriented measures well-known in data base theory that revolve around the use of (simple) meta-data, organizational measures well-known in organization science, e.g., job rotation, job enrichment or ICT measures well-known in MIS, e.g., the use of data bases, email or Groupware. Several case studies deal with the introduction of KM in organizations and describe what instruments were used. Table 1-6 lists some examples of case studies.

Table 1-6. Proposed instruments and supporting measures

instrument	measures
case debriefings	several information systems including yellow pages and a case data base; new roles like knowledge stewards, coordinators and advocates and organizational rules
best practice sharing	a new organizational structure with several centers of excellence, an information system containing best practices and the adoption of benchmarking and models
externalization of knowledge	career plans, incentive systems, 360° evaluation, an electronic document management system and yellow pages, the introduction of so-called Intellectual Capital Teams that review new documents

Table 1-6. Proposed instruments and supporting measures

instrument	measures
documentation/ evaluation of customer feedback	establishing a new team and regular meetings, creating templates and organizational rules
lessons learned	method for systematic harvesting of lessons learned in projects at defined project steps; consists of organizational rules, document templates and an IT system
community of experts, interest, practice, purpose	foster networking between experts (community of experts), employees working on (community of practice) or interested in a topic (community of interest) or working towards a common goal (community of purpose)
knowledge maps	consistent access to customer, product and process knowledge with the help of organizational rules and visualization tools
corporate and team culture management	corporate culture: off-shore meetings, expert meetings and debriefings; team culture: new team structures, informal interviews and an education program
documenting tacit knowledge, identifying and integrating external knowledge	a new organizational unit, document management system, access to an online encyclopedia, lessons learned enforced through a workflow management system and "in-a-nutshell" learning videos

KM instruments target different goals and consist of several measures that have to be aligned and supplement each other. Most of the instruments described in Table 1-6 comprise organizational as well as technological measures. Thus, it is useful to review a human-oriented and a technology-oriented perspective on KM instruments before aiming at a comprehensive definition of KM instrument.

Human-oriented definition

Instruments for knowledge organization are intervention tools that are describable, get deployed purposefully in a way that is traceable for an observer, have a clear knowledge orientation and are still relatively independent of the respectively organized knowledge (Roehl 2000). This definition has its roots in organizational psychology and sociology. The implementation of knowledge strategies is seen as a purposeful intervention into the way an organization handles knowledge. Having a clear knowledge orientation distinguishes KM instruments from other tools that help in an intervention into an organization, but remains unspecific about what exactly knowledge orientation is. In the case of ICT, knowledge orientation can be expressed by specific "intelligent" functions and specific content, with content being the most important part. Knowledge refers to con-

textualized information in an ICT context. Thus, KM instruments have to provide context in order to show knowledge orientation. Finally, a KM instrument in this view has to be general, spanning knowledge domains rather than being domain-specific.

Technology-oriented definition

Knowledge management tools are technologies, broadly defined, which enhance and enable tasks along the knowledge life-cycle, e.g., knowledge generation, codification and transfer. As with any tools, they are designed to ease the burden of work and to allow resources to be applied efficiently to those tasks for which they are most suited. It is important to note that not all knowledge tools are computer-based. Pulling these two perspectives together leads to the following definition.

Definition of knowledge management instrument

KM instruments (1) are part of an ICT-supported intervention (2) into an organizational knowledge base and consist of (3) a collection of organizational, human resources and ICT measures that are aligned, (4) clearly defined, (5) can be deployed purposefully in order to achieve knowledge-related goals, (6) target contextualized information as object of intervention and (7) are independent of a particular knowledge domain.

Detailed analysis

(1) ICT-supported intervention stresses the importance of ICT tools as enabling KM in organizations. (2) Organizational knowledge base reflects people's skills, the contents as well as (ICT) tools and systems in an organization that support handling of knowledge. (3) Only parts of the valuable knowledge exist in explicit form as documented, electronically accessible knowledge. Therefore, KM instruments have to consider person-oriented measures. Organizational measures are implemented e.g., as rules, roles and procedures that describe how to deal with ICT systems. (4) Clearly defined means that any proposed instrument has to clarify what measures and tools are involved so that it is possible to decide if an observed phenomenon in an organization matches this definition. (5) KM instruments have to be purposefully deployed within an organization, usually within the frame of a KM initiative. That includes defining knowledge-related goals and respective measurement.

(6) Knowledge orientation of the KM instrument can only be accomplished if the contents of the ICT systems are “knowledge-prone”, thus being contextualized information instead of only data. An example is a data base containing experiences, lessons learned or best practices together with links to people who have made these experiences and/or experts in the domains that are described (knowledge) as opposed to a data base holding telephone numbers of employees (data). Embedding information into context is crucial. In ICT systems, it can be achieved by assigning appropriate meta-data to help users to integrate information into their personal knowledge bases. (7) Finally, a KM instrument should be indepen-

dent of a special knowledge domain and can be targeted at any topic or (core) competence of an organization.

The next section organizes some important KM instruments that have been proposed in the literature and are applied widely in organizations.

1.4.2 Classification

Figure 1-10 gives an overview of KM instruments.

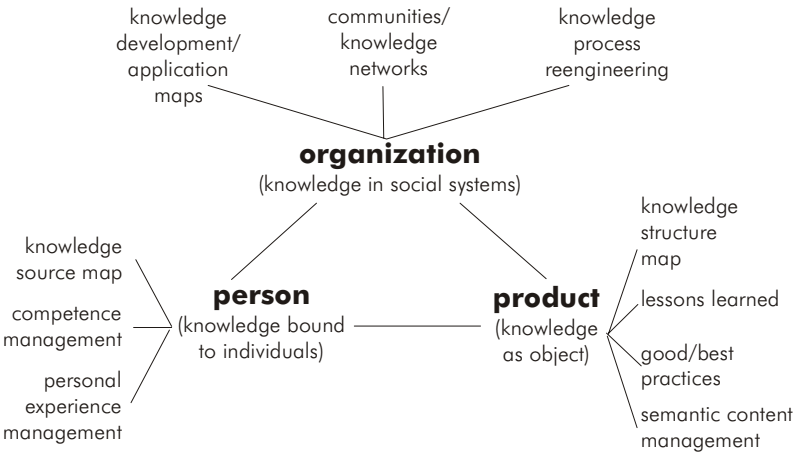

Figure 1-10. Knowledge management instruments

Medium of knowledge

Even though KM instruments have been defined as comprising person-oriented, product-oriented, organizational and ICT measures, actual KM instruments usually target one of the media of knowledge (see Figure 1-5 on page 21). All KM instruments, however, are supported by ICT. Basically, KM instruments thus are distinguished according to the main medium of knowledge into person-oriented, product-oriented and organizational instruments.

Knowledge life-cycle

This distinction also reflects a portion of the knowledge life-cycle that shows knowledge in terms of increasing organizational commitment and legitimation and thus extends from personal experiences until redesigned organizational processes (Figure 1-11).

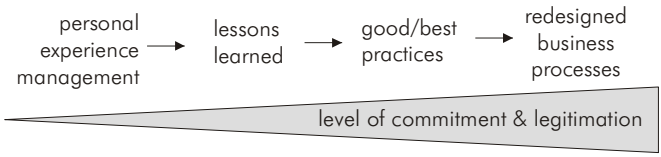

Figure 1-11. Portion of the knowledge life-cycle

The knowledge life-cycle starts with individual experiences which have the least level of organizational commitment. Individual experiences are discussed, filtered and further explored in a team. If the team commits to certain experiences, they are called lessons learned. This process can be aided by a lessons learned coach that helps the team to structure the process of group reflection on team experiences. Further commitment and legitimation is needed in order to turn lessons learned into good practices. Practices can be seen as guidelines how to act in certain situations. Sharing good practices throughout the organization and agreeing that this is the best way to deal with a specific situation turns them into (organizational or local) best practices. Knowledge process reengineering is finally one method for redesigning business processes taking good and best practices into account. Knowledge bound to an individual is disseminated in the form of knowledge products that ultimately reside in social systems, changed business practices and processes.

Knowledge mapping

Different types of knowledge maps are suggested for all categories of KM instruments. Central goal in this instrument is the creation of corporate knowledge directories which visualize existing knowledge in organizations and support a more efficient access to and handling of knowledge. The main objects of mapping are experts, project teams, networks, white papers or articles, patents, lessons learned, meeting protocols or generally document stores. In the following, the individual KM instruments are discussed in detail.

Person. Person-oriented KM instruments primarily aim at knowledge that is bound to individuals, e.g., personal experiences, ideas, proposals etc. These are knowledge mapping, competence management and personal experience management.

Knowledge source maps

Knowledge source maps visualize the location of knowledge, either people (sometimes also called knowledge carrier maps) or information systems and their relation to knowledge domains or topics. They can be further classified into knowledge topographies to identify gaps, competence maps to find experts and pointer systems that directly link from challenges within a process to a contact that can assist. Knowledge asset maps visualize also the amount and complexity of knowledge that a person or system holds.

Competence management

Competence management supports systematic analysis, visualization, evaluation, improvement and usage of competencies held by individuals in organizations. Competence management comprises expertise locators, yellow and blue pages as well as skill management systems, also called people-finder systems. Skill management comprises an information system that makes skill profiles accessible, learning paths that have to be defined for each employee and that have to be updated together with skill profiles. A central skill ontology has to be defined that provides context for all existing, required and wanted skills in the organization. Training measures

have to be offered. Skill management systems are often not limited to information about skills, their holders and their skill levels, but also contain information about job positions, projects and training measures in which employees learned, used and improved their skills. Yellow and blue pages are directories of organization-internal and -external experts respectively. Profiles of the experts together with contact details are listed according to a number of knowledge domains for which they might be approached. Information about employees' skill levels and degrees of expertise can be used e.g., to connect people, to staff projects, to filter and personalize contents and functions in enterprise knowledge infrastructures.

Personal experience management

The implementation of experience management systems eases documentation, sharing and application of personal experiences in organizations. These systems have to be integrated into the daily work practices of employees in order to be accepted. Several approaches exist that support capturing of experiences, e.g., information mapping, learning histories or micro articles that help employees to document and structure experiences. On organizational level, systematic management of personal experiences enables a company to solve recurring problems more effectively. However, there are some barriers which prevent the documentation of experiences or reuse of already documented experiences. Foremost, time required for documenting experiences is a critical factor because it imposes additional efforts on employees. Therefore, organizational measures are required that provide time tolerances and keep the effort as low as possible. Simultaneously, sufficient context of the experience has to be provided. ICT solutions help to automatically detect context. Personal barriers, e.g., insufficient willingness to share knowledge or to apply knowledge created by other employees (not-invented-here-syndrome) have to be considered by measures like trust management and incentive systems.

Organization. Organizational KM instruments target knowledge that resides in social systems. Social systems in organizations are described with the help of the formal organization design, especially business and knowledge processes supported by knowledge process reengineering and process warehouses, projects and work groups supported by knowledge application and development maps as well as the informal organization, reflected by communities and knowledge networks.

Knowledge development/application maps

Knowledge mapping is used here in order to highlight knowledge processes, especially processes of knowledge development and application. These maps are combinations of process models and knowledge carrier maps. Knowledge development maps visualize processes or learning paths that can or have to be performed by individuals or teams in order to acquire certain skills. Knowledge application maps describe what process steps have to be performed in what situation at what step in a business process, e.g., who should be contacted for a second opinion.

Communities/knowledge networks

Community management targets creation and fostering of communities or knowledge networks. Communities differ from knowledge networks with respect to who initiated their foundation. Communities are founded by like-minded people (bottom-up) and can at most be fostered by the organization. Knowledge networks are established and legitimated by management (top-down). However, organizational and ICT measures to foster communities are the same as the ones used to support knowledge networks. Communities per definition can not be controlled or externally induced. But organizations can provide employees with time and space to share thoughts, establish IT tools, e.g., community builder or home spaces, blackboards or wikis that support exchange of thoughts (section 4.3, 267ff) and create new roles like community managers that help keeping discussions going and look for important topics that should gain management attention.

Knowledge process reengineering

Knowledge process reengineering (KPR) aims at redesigning business processes from a knowledge perspective. The term references the field of business process reengineering (BPR) that aims at fundamental (process innovation) or evolutionary (process redesign) changes of business processes in organizations with the goal to increase organizational effectiveness. In addition to traditional BPR instruments, knowledge-intensive business processes are partially improved by KPR. The focus is on designing knowledge processes that connect business processes, defining cooperation scenarios, improving communication patterns between employees, as well as on "soft" skills or an organizational culture supportive of knowledge sharing (Davenport et al., 1996). Business processes are modeled with the help of modeling techniques. The models are stored in model bases. The model base can be expanded so that it handles not only knowledge about the process, but also knowledge created and applied in the process. This is termed process warehouse which can be used as a foundation for systematic knowledge process reengineering. Examples for contents in process warehouses are exceptional cases, case-based experiences, reasons for decisions, checklists, hints, frequently asked questions and answers, potential cooperation partners or suggestions for improvements.

Product. Documented knowledge certainly is of primary interest with respect to the design of enterprise knowledge infrastructures. Documented knowledge can be spread across multiple sources and requires integration which is supported by ontologies or knowledge structure maps. Ontologies also aid the management of semantic content. While these two instruments target (electronically available) content as potential knowledge sources throughout the organization, there are two instruments that specifically establish the systematic handling of inter-subjective knowledge with commitment, i.e. lessons learned and good or best practices.

Knowledge structure maps

Knowledge structure maps show the relationship between different knowledge domains or topics and should not only visualize that there is a

relationship, but also explain the type of relationship. Formal definition of knowledge structures results in ontologies and is an important instrument for the integration of diverse knowledge sources (section 3.2.2, 175ff).

Lessons learned

Lessons learned are the essence of experiences jointly made and systematically documented by members of the organization in e.g., projects or learning experiments. In a process of self-reflection, e.g., at the end of a project milestone (also called after-action reviews, project debriefings) the project members jointly review and document critical experiences made in this project. Lessons learned should aid individual self-reflection about one's own experiences, joint reflection that explicates know-how gathered in a team and learning from the experiences of others. This process can be moderated by a lessons learned coach. Templates can be created that support a structured documentation of experiences and help the team to include important context information. An information system can aid this process and store and provide access to all documents containing lessons learned. A subject matter expert could review the documents and further enhance them by referencing other documents, projects or people. Rules support integration of the lessons learned instrument into standard project processes and can also enforce that project managers study lessons learned documents before starting a new project.

Good/best practices

Lessons learned target project experiences and their reasons, but ideally make no statement about how processes should be adapted considering these experiences. The sharing of (good or) best practice is an approach to capture, create and share experiences in a process-oriented form as e.g., procedures or workflows. This term in a wide meaning denotes "any practice, knowledge, know-how or experience that has proven to be valuable or effective within one organization that may have applicability to other organizations" (O'Dell/Grayson 1998, 167). As managers might argue about what exactly is "best" in a practice, several organizations use different levels of best practice, e.g., (1) good (unproven) idea, (2) good practice, (3) local best practice, (4) company best practice, (5) industry best practice. So-called best practice teams are permanent institutions within an organization's networking infrastructure. They provide guidelines about what constitutes good or best practices and support identification, transfer, implementation, evaluation and improvement of practices. Goal is continuous process improvement, so employees have to be encouraged to make suggestions for improvement.

Semantic content management

Semantic content management refers to managing meaningfully organized content, i.e. documented knowledge embedded in a context. "Semantic" in this case means that content is well-described with the help of meta-data that assigns meaning and structure to the content and that these descriptions are machine-interpretable and can be used for inferencing (see also section 3.1.1, 149ff and section 3.2.1, 173ff). Semantic content management extends document management and enterprise content

management into integrated document and content management. The instrument is certainly tightly related to an IT solution, but there have to be rules that guide definition and use of semantics, monitoring external knowledge sources for interesting content that should be integrated, developing an appropriate content structure as well as publishing of semantically enriched documents in the system. Semantic content management also allows for "smart" searching, collaborative filtering and can be integrated with competence management in order to handle interests used to connect people with the help of the joint analysis of semantic content and skills.

Infrastructures foster KM instruments

Enterprise knowledge infrastructures aim in general at providing a platform for knowledge management and in particular foster the implementation of knowledge strategies with the help of a defined set of KM instruments. We will now turn to information system architectures before the final section of this chapter will sum up what we have learned so far and give a first comprehensive picture of what an enterprise knowledge infrastructure looks like.

1.5 Information System Architectures

Types of architectures

Architectures play an important role in the discipline *Management Information Systems* as blueprints or reference models for corresponding implementations of the organization-wide landscape of information systems (IS). The term architecture as used in MIS origins in the scientific discipline *Architecture*. An architecture represents the structure of a system by means of the main components of the system and the essential relationships between them. Table 1-7 gives examples for types of architectures used in organizations to document or implement information systems.

Table 1-7. Types of architectures

type	description
application	functions, methods, modules, programs or entire application systems with relations between them
data	entity types and relationship types as an, ideally enterprise-wide, abstraction of multiple data models describing individual data base systems
information system	components, e.g., application, communication, data, technology, of an enterprise-wide information infrastructure and rules that help to implement IS strategy
network	computers, routing devices and network media connecting them
technology	hardware, software and network architecture showing how to implement the logical architectures with information and communication technologies

Plan and documentation function

Architectures are the basis for communication about a system, e.g., between stakeholders during the design process of a system or between members of a steering committee who decide on essential interfaces and interactions between a number of information systems or projects that design information systems. Architectures document fundamental (and often irreversible) design decisions. They can be used as a plan or draft for systems and thus prescribe what these systems should look like (plan function). Architectures can also describe (parts of) information systems as they are, for example showing the main components or modules of a standard software package together with potential interactions with systems provided by other vendors (documentation function).

Conceptual architectures

Architectures can be classified into conceptual and physical architectures. Conceptual architectures guide the implementation of physical architectures. Examples for conceptual architectures are application, data or information system architectures. The latter describes the entire information infrastructure of an organization, also called information landscape, from a logical perspective limited to the most important components, relationships and characteristics of the system.

Physical architectures

Examples for physical architectures are hardware, software or network architectures. These architectures describe the main components that are actually implemented in an organization together with the relationships between them. Physical architectures have been important instruments for planning an organization's computing resources, especially computer servers, personal computers and the network connections between them.

Architectures for enterprise knowledge infrastructures

So far, conceptual architectures primarily aim at a documentation or a plan of the entire information infrastructure supporting business processes in organizations, e.g., the architecture of integrated information systems consisting of data, function, organization, control and output perspectives (Scheer 2001). Architectures for enterprise knowledge infrastructures are seen as complements to existing transaction-oriented architectures. Both types of architectures together describe or prescribe an enterprise-wide information and communication landscape. The construction of a separate architecture for enterprise knowledge infrastructures sheds light on the currently underdeveloped handling of semi-structured data, communication and cooperation that is required in knowledge work.

Section overview

Goal of this section is to discuss some architectural alternatives and related concepts that are important to understand the architecture of enterprise knowledge infrastructures which is presented in section 1.6. Section 1.5.1 presents centralistic, distributed or peer-to-peer as well as agent architectures. Section 1.5.2 discusses approaches to classify systems in an enterprise architecture, namely the IS pyramid, the value chain and an example for an integrated, service-oriented architecture of an enterprise resource planning system.

1.5.1 System Architectures

Architectures

System architectures represent the structure of software applications on an abstract level. Components are delimited parts of software systems. Examples are functions, procedures, abstract data objects, abstract data types or object classes. There are a number of views on what constitutes a system or software architecture. For instance, there are different views on what the major components of a system are, how the structure looks like and which decisions are hard to change or irreversible.

Views on software architecture

Four major views on software architecture can be distinguished (Clemens/Northorp 1996): a) the *conceptual view* represents the user's view and the functional requirements and tasks of the system, b) the *module view* depicts the structure of software modules for software developers, c) the *process view* describes control and communication flows and allows to analyze performance, throughput and synchronization of processes or functional groups, and d) the *physical view* renders the mapping of process view and module view to the underlying hardware.

Section overview

There is a large body of literature on software architecture which cannot be discussed here. In the following, only some basic concepts and architectural alternatives are briefly described which are needed in order to understand the enterprise knowledge infrastructure presented in section 1.6.4. These are structured into centralized, peer-to-peer and agent architectures.

Layers

Centralized architecture. A common principle to structure architectures is to form layers. The goal is to minimize dependencies between layers so that changes on one part of the system do not require changes in other parts. Another goal is to design systems with interchangeable layers. Thus, a general rule of a layered architecture is that components in one a layer can access each other without restrictions, whereas interaction across layers is restricted by (standard) interfaces between the layers (Figure 1-12). Advantages of a layered architecture are that the software or parts of it can be reused, maintained, standardized, ported and tested. Disadvantages are that extra layers can harm performance because data has to be passed through multiple layers which may lead to substantial overhead. However, encapsulation of functionality within layers increases reusability and reduces maintenance efforts that often compensate the additional effort induced by communication between layers.

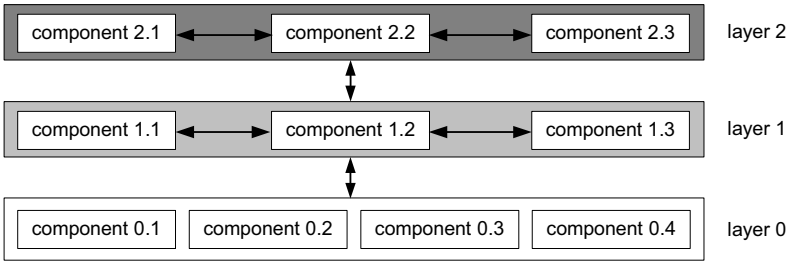

Figure 1-12. Generalized version of a layered architecture

The ISO/OSI-layer model (section 2.1.3, 88ff) is an example for a layered architecture. Changes in lower layers, e.g., using Ethernet instead of Token Ring on the network layer do not affect higher layers. Thus, it is possible to operate TCP/IP on different lower-level network protocols.

2-tier vs. 3-tier architecture. The use of layers as a principle to structure software became more and more apparent with client-server systems during the late 80s and 90s of the last century.

2-tier architecture

Those regularly had 2-tier architectures. A client contained the application and presentation logic and the server usually was a (relational) data base (Figure 1-13). This architecture is well-suited for data-intensive applications for displaying and updating relational data. But when it is necessary to apply more complex calculations or validations to data, the logic for that has either to be implemented on the data source layer or on the presentation layer. On the one hand, choosing the data layer leads to the danger of the application logic becoming dependent on the data structure. If the data structure has to be changed, the application logic also needs to be updated. On the other hand, implementing business logic on the presentation layer means that it has to be duplicated when alternative ways of presenting data are required.

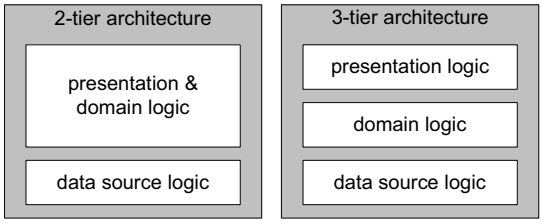

Figure 1-13. 2-tier architecture vs. 3-tier architecture

3-tier architecture

A good alternative thus is to encapsulate business logic into a separate layer, which leads to a 3-tier architecture (Figure 1-13). Today, systems with a 3-tier architecture are fairly common. The need to access data with

Web interfaces, e.g., using standard Web browsers, and propagation of object-oriented languages like Java may have accelerated this development. The characteristics and tasks of each layer are as follows:

Data source logic

Data source logic is responsible for communication with other systems that manage data on behalf of the application. Examples are transaction monitors, other applications or messaging systems. The most important data sources for enterprise applications certainly are data base management systems responsible for storing persistent data.

Domain logic

Domain logic actually performs transformations on data which are required to represent transactions in the user's business world (=domain). Thus, it is also referred to as business logic. It comprises e.g., calculations on the basis of inputs and stored data or choices of data sources, depending on commands received from presentation logic.

Presentation logic

Presentation logic handles interaction between user and application. Its task is to display information to the user, interpret user's commands and translate them into actions performed by the domain and data source layers. Graphical user interfaces are most common these days, text-based presentation and command lines are alternatives that have more and more receded during the last years. Today, they are mainly applied for specialized tasks like maintenance and administration of computer systems. Principally, users can be humans or other systems.

Example

The general principle that all layers should be independent from each other as much as possible. For instance, when sales data needs to be summarized by a defined set of regions, it has to be done in the domain logic layer. Short-term, it would not be a problem if the task is fulfilled as part of the presentation logic. But if the functionality is needed later e.g., to print out the values, the algorithm has to be duplicated. This might lead to inconsistencies and increase efforts required for maintenance.

Centralized client/server architecture. A variety of ICT services in an organization needs to be offered for and shared by all or at least a considerable number of users of the infrastructure. Reasons are efficient utilization and thus shared use of resources and the necessity to exchange and share data, information and knowledge.

Clients and servers

Thus, it is beneficial to offer some services centrally by so-called *servers*. The units that request and use these services are called *clients*. In the context of ICT infrastructures, these terms are usually used for logical (software) units like applications, software components as well as single software objects. Regularly, they are also applied to denote hardware devices, i.e. a server is a (dedicated) machine which solely provides services to (users on) other machines, especially personal computers or work stations which are called client machines.

Thin clients vs. rich clients

Different types of clients can be distinguished depending on how work is split between client and server, i.e. which layers are performed on the client and which ones are on the server. Clients that only contain presenta-

tion logic are called *thin clients*. Clients that additionally contain (substantial parts of) domain logic are termed *rich* or *fat clients* (section 5.2, 320ff). In some settings it may make sense to include all three layers on a client, e.g., to ensure fast responses and offline operation. Client and server then usually need to offer synchronization functions to secure data consistency.

Terminals

So-called terminals consist of a computer screen and keyboard, have practically no processing ability and were used widely in the early days of IT. These terminals forward user requests to be performed on a remote server, minicomputer or mainframe. Today, connections to mainframe computers are emulated by virtual terminals, e.g., the telnet client.

Example: X windows

An example for a thin client is the X windows system which is the standard graphical user interface on machines with the operating system Unix and Unix derivatives. Confusingly, the terms client and server are reversed with respect to the hardware devices on which they run. Display servers (presentation, user interface) run on thin client machines and communicate with client programs on server machines. The display server accepts requests for graphical output (screen windows) and sends back user input (keyboard, mouse). Other examples for thin clients are Web browsers that request from Web servers, display Web resources to users and provide an interface for user interaction.

Rise of thin clients

From the view of administration and maintenance, it is regularly most desirable to utilize thin clients as this reduces efforts for software and hardware maintenance. This is one reason for the rise of Web-based systems within organizations. However, thin clients restrain usability and interaction with applications, e.g., through less responsiveness and restricted possibilities for user interactions. Their applicability is fundamentally determined by network bandwidth. Table 1-8 lists typical services centrally provided by servers.

Table 1-8. Examples for typical server services

application area	service	examples
identification	addressing	domain name system - DNS, dynamic host configuration - DHCP
	authentication	LDAP directory, issuing/checking of certificates
input / output	scanning	network scanner
	printing	network printer
	communication	mail, tele-conferencing, remote access
processing	application	application server

Table 1-8. Examples for typical server services

application area	service	examples
	computing	computation of chemical structures or flow models
	Web provisioning	Web server
security	filtering	virus scanner, SPAM filter
	privacy, restricted usage	digital rights management, encryption
storage	caching	Web proxy
	data storage	file server, relational DBS, document management systems
	backup	tape library
synchronization	time	time service via network time protocol

Implementation

Technical issues that constrain decisions about how centralized services are implemented (e.g., type and scope of services, number of served clients) are network bandwidth and performance of server hardware. Large server systems usually consist of more than one hardware machine that each may fulfill different tasks such as data base services, processing services and Web services.

Example: Google

For example, the so-called Googleplex of the internet search engine Google (www.google.com) is supposed to be composed of in total more than 30,000 machines in different locations. In Dublin, one of Google's European locations, the company hosts a minimum of 100 racks each with 88 CPUs (dual 2 GHz Intel Xeon, 2 GB of RAM, 80 GB hard disk; all figures from April 2004). This hardware equipment is necessary to ensure quick responses to user search requests and to crawl and index Web sites, so that search results are always up to date.

The servers may have individual operating systems or a shared operating system installed. If one server fails, another can act as backup. Usually, this is not visible for users, the configuration of servers appears as to be one system to them. Load balancing is used to distribute requests from clients to server resources. They are especially applied in settings where the number of requests can not easily be forecasted.

Peer-to-peer architecture. The term peer-to-peer denotes the idea of a network of equals (peers) that provide resources such as CPU time, storage area, bandwidth or information to each other without a central coordinating instance. In the terminology of client/server-systems, a peer acts both as client and as server: Every peer uses services of and provides services to other peers. Together, the peers shape a dynamic, continually changing

network with each node always being able to participate in or leave the network and to choose autonomously which other nodes to communicate with.

Peer-to-peer, a new idea

The peer-to-peer idea is not new, some argue that it is in fact one of the oldest architectures in the ICT and telecommunication field with the telephone system, the Usenet and the Internet as major examples that employ this metaphor. However, due to the tremendous growth of peer-to-peer file sharing networks, the idea has risen again on individual personal computers as opposed to backbone networks.

Characteristics of peer-to-peer networks

Ideally, peer-to-peer networks can be described by the following characteristics:

- *mutual client-server-functionality*: every peer can act as a client and as a server, thus rendering all nodes functionally equal,
- *no central authority*: there is no central node which coordinates communication between peers,
- *autonomy*: peers are solely responsible for their activities, especially for determining what resources they share when and with whom,
- *distributed data*: every peer stores a fraction of the whole system's data and there is no peer that manages the whole data,
- *local interactions*: every peer only sees its direct neighborhood, i.e. the peers he is directly connected to,
- *dynamic changes*: peers and connections between peers are not necessarily reliable which means that they may go off-line or yet cause damage to the network by spreading viruses or worthless contents.

Differences to client/server-architecture

A main difference to client/server systems is that the number of peers strongly determines the overall amount of resources available. Every participating peer adds resources like storage capacity and processing power to the whole network. This is the reason why peer-to-peer systems are often characterized as to be highly scalable. However, network bandwidth may limit the size of a peer-to-peer network since a considerable share of network traffic is necessary to discover other peers and their resources. Another important difference especially interesting for KM is that peer-to-peer networks are designed to easily form ad-hoc networks, which is a good foundation for spontaneous collaboration and knowledge sharing.

Application areas

Examples for application areas of existing peer-to-peer networks are:

- instant messaging, e.g., the well-known ICQ network,
- file sharing, with prominent examples, e.g., Gnutella, Kazaa, Napster or Overnet (Edonkey 2000), i.e. peer-to-peer software that supports the sharing of files in networks of users, especially audio and video data as well as computer games,
- distributed and grid computing which aims at a coordinated usage of distributed computing power, with the prominent example of the world-

wide network that jointly processes data on the search for extraterrestrial life (SETI@HOME),

- collaboration and Groupware, with Groove (section 4.3.2, 276ff) being the most cited distributed Groupware platform that employs the peer-to-peer-metaphor.

Challenges

However, there are still serious technical challenges that have to be overcome in peer-to-peer computing. These challenges concern:

- *connectivity*, e.g., locating peers that do not have public IP addresses and mechanisms for communicating through firewalls,
- *security and privacy*, especially the risk of spreading viruses, unauthorized access to confidential and private information and installation of unwanted applications,
- *fault-tolerance and availability*, e.g., finding the required resources available when they are needed,
- *scalability*, especially concerning the naming scheme and searches in the flat structure of the distributed search domain,
- *self-managed systems* that are administered by individual users with limited experience and tools who provide services to others and
- *interoperability*, i.e., current peer-to-peer installations cannot connect to each other due to e.g., a variety of computing models, a variety of network settings and a wide range of application types.

Ideal types of peer-to-peer architectures

In the context of organizations, peer-to-peer systems should be rather seen as complementing existing client-server systems than as competing approach. Many tasks are immanently centralized. Examples are processing and analyzing large amounts of business transactions within management support systems and maintenance tasks like software license management, software updates or bandwidth management. Even peer-to-peer systems might deploy centralized devices for authentication, indexing and search of contents, message relaying and discovery of other peers. Contrary to the client-server architecture, peers in the peer-to-peer architecture communicate and offer services directly to other peers. Three ideal architectural types of peer-to-peer systems can be distinguished with respect to whether there is any central assistance at all or the role that central assisting devices play in the network respectively (Figure 1-14):

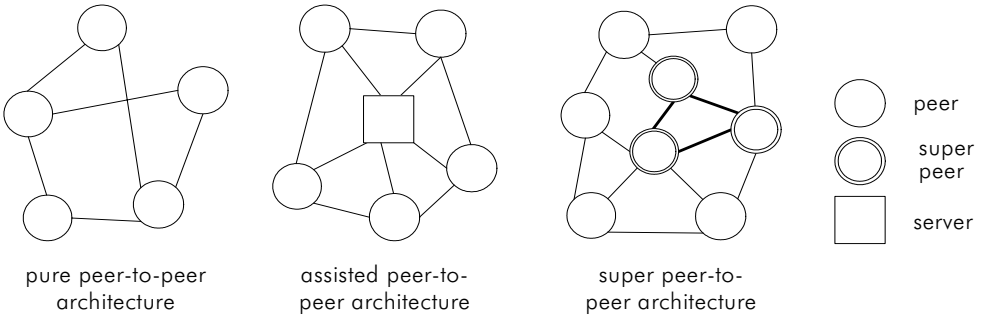

Figure 1-14. Ideal architectural types of peer-to-peer systems

- The *pure peer-to-peer architecture* does not have any central authentication or coordination mechanism. Every peer provides complete client and server functionality.
- The *assisted peer-to-peer architecture* requires a central server, e.g., to authenticate users, to act as global search index or to provide interfaces to other centralized systems.
- The *super peer-to-peer architecture* is in between assisted and pure architectures. A peer is connected to one single super peer, thus forming clusters of peers in the network. Super peers are also connected to each other, thus forming a hierarchically superordinated peer-to-peer network. Requests from peers are always handled by the connected super peer and eventually forwarded to other super peers. As in the assisted architecture, a direct connection between peers is established, once a peer with the desired resource is found.

Agent architecture. In general, software agents should not be seen as alternatives to centralized or distributed systems, but as complements. But what are software agents? In the literature there is no single accepted definition due to heterogeneous research domains of the community of agent researchers. But there is a widely accepted set of attributes that a piece of software must have in order to be considered an agent. The most important attributes are autonomy, rationality and pro-activeness.

Definition of software agent

A software agent is an autonomous software entity that can act reactively and proactively, can communicate with other agents and uses a rational calculus to develop and carry out plans in order to reach its goals (Wooldridge/Jennings 1995, Kirn 2002).

Further characteristics of agents

Many other attributes like intelligence, sociality, intention, deliberation, cognition, mobility and persistence are often used in conjunction with agents, but most of them are either not essential (mobility and persistence),

difficult to define (intelligence, sociality) or are just other words for aspects already covered by autonomy, rationality and pro-activeness.

The definition above is general enough to comprise a large number of concrete software systems, starting from Unix-processes (daemons) that run in an own thread, are monitoring their environment and communicate with other processes by passing messages or invoking functions. Processes of this kind are regarded as representing the weak notion of agency, i.e. a wide interpretation of the term. A narrow interpretation, called the strong notion of agency, is mainly represented by research in the artificial intelligence community that demands that agents have to have sophisticated mental models, use machine learning algorithms and symbolic reasoning. In both cases, agents must not be mixed up with objects (in the sense of object-oriented programming). Where objects encapsulate data and methods that operate on this data, agents also encapsulate their behavior. That leads to the fact that often an agent exposes only a very limited set of generic methods to its environment that hide its real purposes. Examples for methods are `perceive()`, `reason()` and `act()`.

Agents and environment

In general, agents exist in an environment of some kind. which is monitored by them. They have an internal representation of the environment (world model) and interact with it. The main distinction between agent architectures is how they are doing this. There are a large number of suggestions for agent architectures for general or for specific purposes. They can be classified as reactive, cognitive and deliberative architectures.

Reactive architecture

Reactive agents only wait for certain stimuli and react in one of certain pre-defined ways based on their internal state. Therefore they do not have to make plans for the future.

Cognitive architecture

Cognitive agents are influenced by psychology so that their architecture is often an implementation of a model of human mind. Examples are Soar and ACT-R.

Deliberative architecture

Deliberative agents finally have to contain a symbolic world model and to make decisions based on logical reasoning based on pattern matching and symbolic reasoning.

Challenges

This leads to great challenges for creators of agents including the translation of real-world phenomena into adequate symbolic representations and the design of reasoning processes that are somewhat problem independent. The reasoning process can be goal-based or utility-based. The latter has strong similarities to micro economic models of a rational decision maker.

Hybrid architecture

There are also hybrid architectures that try to integrate reactive and deliberative behavior as humans also have both: reflexes for quick reactions in critical situations and reasoning for optimal decisions with long reaching consequences.

Example: InterRAP

One example for a hybrid architecture is InterRAP (Müller 1995). It is a layered architecture with three layers (Figure 1-15):

- the behavior-based layer that is responsible for reactions,

- the local planning layer that is responsible for plans the agent can carry out on its own and
- the cooperative planning layer that is used for plans that involve the activity of other agents.

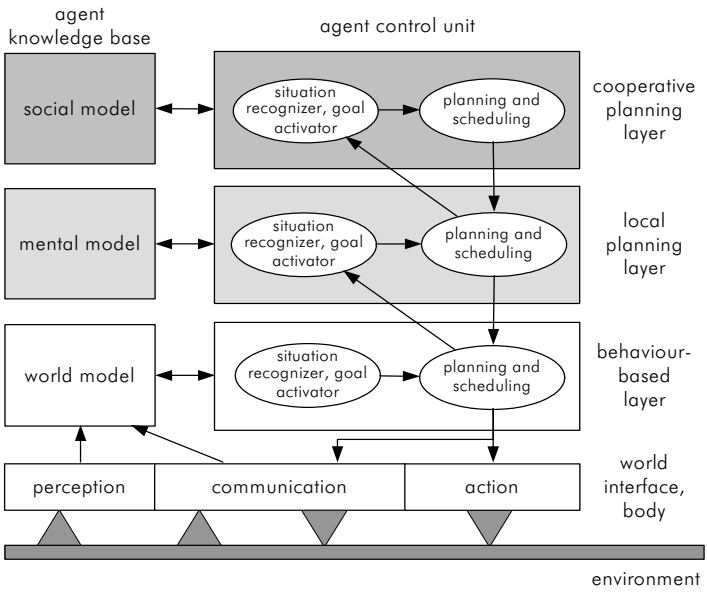

Figure 1-15. InterRAP agent architecture

BDI model

Accordingly, the knowledge base of an InterRAP agent is also divided into three parts: (1) the world model for reacting, (2) the mental model for local planning and (3) the social model for cooperative planning. Layers are activated bottom-up. If the lower layer cannot cope with a given situation, then a higher layer is activated.

For planning, InterRAP builds on the BDI model (Belief, Desire, Intention, Rao/Georgeff 1991). Beliefs are statements about the environment, the agents think are true. The sum of all beliefs represents the world model of an agent. Desires are states of the environment and the agent itself that the agent wants to achieve. Desires often are contradicting (like having money on the bank account and spending money to buy a car) so that goals have to be set, that are consistent and reachable. Intentions are concrete actions that an agent wants to take in the near future to reach a goal.

Application in knowledge infrastructures

There are a lot of application areas for software agents like distributed problem solving and controlling complex machines. For knowledge infrastructures, two types of applications are of special interest: information agents and interface agents.

Information agents

Information agents monitor information sources and collect information they consider useful for their user, often based on keywords that represent the user's interests. They help a user to stay up to date without requiring him or her to select relevant information by him- or herself. Typically, the agent uses extensive user profiles that are permanently updated to get more relevant results than usual search engines. It typically also acts as a meta-search engine and Web crawler by retrieving information from several search engines and additionally crawling pages that are not indexed by search engines. Examples of such agents are Copernic Agent and WebMate developed at the School of Computer Sciences of the Carnegie Mellon University.

Interface agents

Interface agents monitor user activities and suggest improvements, provide help for functions or automate short tasks. They often have graphical representations that help users to develop a sort of "social relationship" with the machine. An example for a simple interface agent is the MS Office assistant. It acts as a shortcut to the applications' help functions and provides hints thought helpful in the current situation. In the context of knowledge infrastructures, an interface agent could monitor the user's activities, deduce the mode and context a user is working in and suggest a number of meaningful actions the user could take (section 6.1, 352ff).

1.5.2 Enterprise Architectures

Knowledge work was described as being creative work that requires creation, acquisition, application and distribution of knowledge and bases inputs and outputs primarily on data and information (section 1.2, 24ff). Individuals concerned with knowledge and information work deal with structured and semi- or unstructured data. EKI is primarily targeted at providing flexible support for dealing with semi- or unstructured, contextualized information, but also needs to integrate organizational systems that manage structured data.

Business information systems (BIS)

These systems have a long tradition in representing business transactions and the flow of materials, goods and services through an organization in information systems. These information systems have been termed operational or transaction processing systems. The data collected in these systems is stored in relational data base systems and filtered, aggregated as well as integrated into data warehouses. The data is then analyzed and presented with the help of management information systems, data mining and decision support systems as well as executive information systems. Altogether, these systems primarily focus on structured data and are called business information systems.

Classification of BIS

From the perspective of an EKI, these systems primarily play the role of data sources. They can be integrated at different levels (data, domain or presentation layer, section 1.5.2, 59ff) depending on the requirements of the actual application scenario. This section gives an overview of business

information systems for managing structured data with the help of two schemes that are suited to form classes of IS: the IS pyramid and the value chain.

IS pyramid. This scheme classifies systems along two dimensions: (1) the time horizon of the decisions they are designed to support (and thus the time horizon of the corresponding data) and (2) the business or functional area in which the system is applied (Figure 1-16).

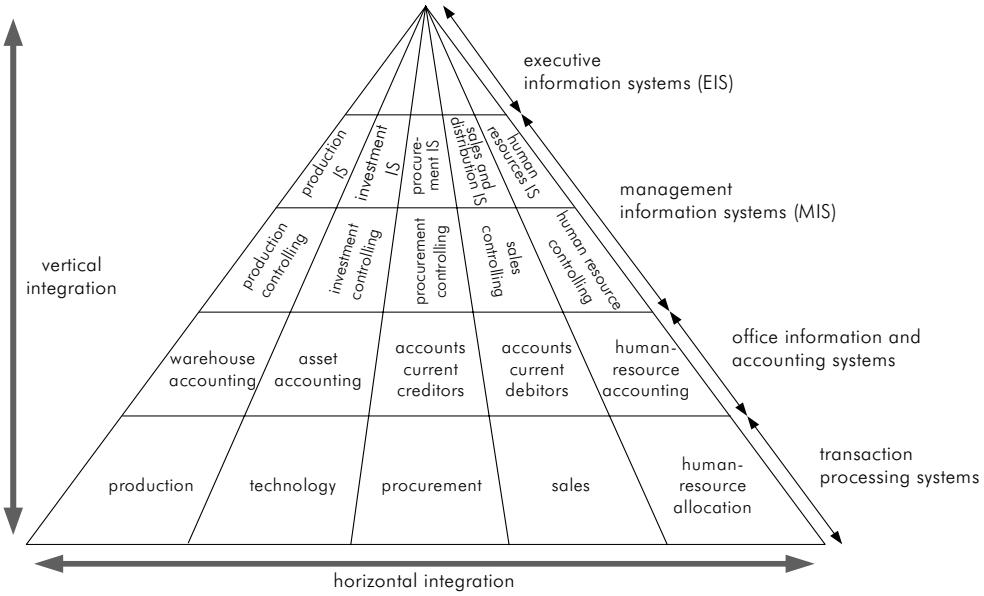

Figure 1-16. Information systems pyramid (based on Scheer 1994)

Time horizon

The decision horizons can be

- *operative*, i.e. with a short time horizon (hours, days or a few weeks) which means they are less important for the long-term development of the organization, e.g., reordering of consumable materials,
- *tactical*, i.e. with a medium time horizon (several weeks up to two years) and are placed to steer the operative business, e.g., choice of supplier for the next year,
- *strategic*, i.e. with a long time horizon (two to five years) constituting the development of the organization and its goals, e.g., definition of product lines based on the selection of product-market combinations.

Functional area

The functional areas are classified according to the main business functions, e.g., purchasing, materials management, production, marketing or human resource management (HRM). The business functions are supported by specific software functions, modules or entire systems, e.g., e-

procurement system, material management module of an enterprise resource planning (ERP) system, production planning system, computer-integrated manufacturing system, sales management systems customer relationship management system, HRM module of an ERP system.

Horizontal and vertical integration

The IS pyramid shows two directions for integration between systems: horizontal integration connects systems of different organizational functions on the same level. Vertical integration connects systems within the same organizational function on different levels. In this case, integration means connecting data and coordinating tasks, processes, methods and programs (chapter 3, 147ff).

The following classes of information systems can be distinguished according to the different levels of the pyramid:

Transaction processing systems

Transaction processing systems (TPS, also called operative IS) support the day-to-day operations of the organization with the aim to make operative processes more efficient, e.g., by automating and supporting them with periodical reports and statements. They manage data about quantities of parts, products, service units, etc. on a fine-grained level. Data is handled by (relational) data base management systems for flexible, application-independent and efficient storage and retrieval of data.

Accounting systems

Accounting systems are oriented towards financial value. These are systems to support business transactions that handle data about payments, billing and accounts. Data about single operative tasks is aggregated to business transactions. Nevertheless, accounting systems still handle data on a low level of granularity which means they are disaggregated.

Office information systems

Office information systems (OIS) support administrative tasks by helping employees to collect, transform, store and exchange data and information. Examples are text processing software, communication and collaboration applications such as e-mail clients, Groupware systems (section 4.3, 267ff) and project management software. These end-user tools are sources for semi-structured and unstructured data such as text documents (e.g., proposal, meeting protocol), spreadsheets (e.g., budget plan, work time record) and presentations (e.g., product information).

Management information systems

Management information systems (MIS) assist middle and upper management in planning and control tasks, e.g., by analyzing business data, comparison of targets and actual business results, guiding and structuring decisions as well as giving hints and suggestions for corrective measures. Examples are decision support systems, query and reporting systems, online analytical processing (OLAP) for analysis and exploration of business data. Many of these tools build on data warehouses that manage a data base of theme-oriented, integrated, time-variant and non-volatile data in support of management's decisions.

Executive information systems

Executive information systems (EIS) finally are information systems that target senior and top management of an organization. They highly aggregate data up to only a few measures that provide orientation and often provide advanced visualization techniques which use well-known

metaphors to direct attention to crucial developments or exceptional data constellations, e.g., charts, traffic lights, cockpits, maps in geographical information systems or warning thresholds.

IS along the value chain. The value chain structures an organizations' activities to create value and is an instrument to analyze the sources for competitive advantage (Porter 1985). Value is defined as output of the organization that customers are willing to pay money for and created along the value chain through the contribution of single value-creating activities.

Primary and support activities

Value-creating activities can be classified into primary and support activities. *Primary activities* are directly related to value creation, e.g. manufacturing of parts and distribution of products. Every organization has primary activities for inbound logistics, operations, outbound logistics, marketing and sales, and customer service. *Support activities* facilitate primary activities and other support activities by supplying resources such as factors of production, technologies, and labor. Value creating activities are the foundation for an organization to add value and create a margin that results in profit.

Business processes

The value chain arranges value-creating activities on a general, abstract level. On a more detailed level, activities can be described by business processes that sequence actions, functions or tasks as parts of activities. In analogy to the classification of activities in primary and supporting activities, they can be classified into core processes and supporting processes.

Classes of IS can be formed according to the activities and business processes along the value chain they support, which determines the type and domain of data and information they handle. Figure 1-17 depicts the value chain and relates examples for system classes to the value chain. In the following, the discussion concentrates on those classes of systems that have not been presented yet, starting with system that facilitate support activities of an organization's value chain.

Facility management systems

Facility management systems ease the planning and control of the organization's basic infrastructure, i.e. buildings, rooms and related basic services such as electricity, illumination, air conditioning, network access etc. This comprises tasks such as managing usage requests (e.g., booking of meeting rooms), accounting usage expenses, tracking maintenance cycles and controlling basic services.

E-procurement systems

Generally, e-procurement comprises handling of all kinds of procurement processes by means of electronic systems. The term is often used in a narrower sense to denote the ordering of maintenance, repair and operations (MRO) goods directly by the consumer, i.e. an employee at his workplace. Those systems are usually implemented based on Internet technologies. Important motivation for the introduction of e-procurement systems is that a centralized procurement of cheap goods can cause disproportional high costs.

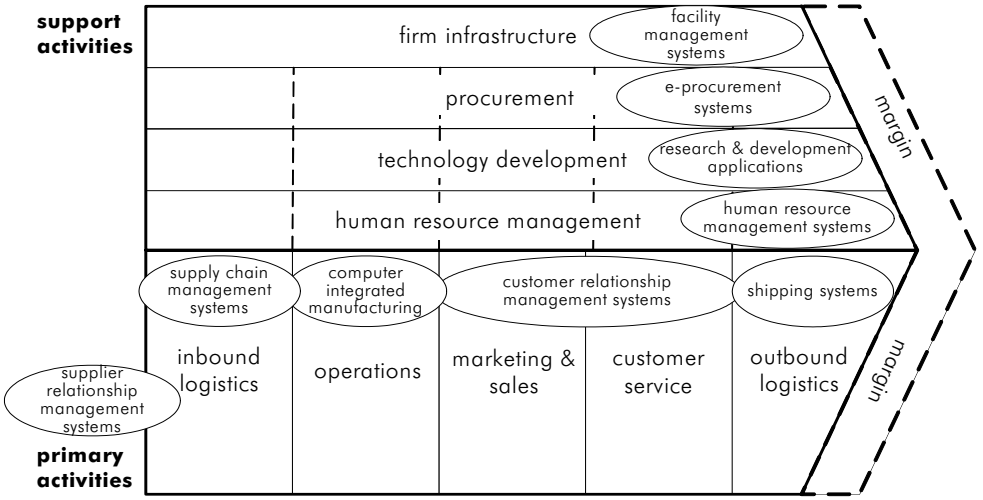

Figure 1-17. Examples for information systems along the value chain

Research & development applications

Systems supporting research and development in organizations are regularly highly specialized according to specific domains of research or industry sector. This comprises systems that support planning, development and simulation of e.g., new services, products and technologies such as computer aided design (CAD) applications, expert systems, visualization tools (e.g., for molecules) and simulation systems.

Human resource management systems

Human resource management (HRM) systems support and automate selected tasks related to managing the personnel of an organization such as calculating and paying salaries, planning career paths and skills as well as managing an employee's or a group's workload. Data security and protection are important issues as the stored data is very sensitive.

Supply chain management systems

Supply chain management (SCM) aims at increasing efficiency of processes associated with acquiring goods and services, managing inventory, and supplying and processing of materials. Examples for systems that facilitate SCM are EDI (electronic data interchange) systems for standardized data exchange between customer and supplier and collaborative planning systems for supporting a joint solution of planning problems by members of different organizations. SCM manages structured data about products, services and processes of suppliers and data of inventory management.

Computer integrated manufacturing

Main aim of computer integrated manufacturing (CIM) is integrated support of production operations and of disposition and control tasks to facilitate efficient operations. Goals are e.g., optimal utilization of resources, optimized stock of inventory, high product quality, adherence to delivery dates and short cycle times. A typical example for this class are

production planning and control systems (PPS) that support materials and capacity management as well as production control.

Customer relationship management systems

Customer relationship management (CRM) comprises techniques and software to manage data describing customers, business transactions and communication acts with customers in an organized and efficient way. CRM systems integrate systems and tools that contain detailed customer information (e.g., offering systems, help desk systems, call center, and marketing information systems) and support all tasks for marketing, sales and customer service. They enable sales and service people e.g., to match customer needs with product offerings and for individualized customer services.

Shipping systems

Shipping systems support all aspects related to product or service orders, delivery and billing. This includes management of freight carriers, auditing of orders and bills, freight tracking during the delivery process and accounting tasks.

Supplier relationship management

Supplier relationship management (SRM) is similar to CRM with the difference that it is directed to the supplier side. It deals with financial information, technologies, research and development as well as future developments of suppliers. Supporting systems thus aim at effective selection and management of as well as communication with suppliers.

Enterprise resource planning

Many of these system classes are incorporated by enterprise resource planning (ERP) software. The term is commonly used for integrated standard software that integrates and automates tasks associated with all operations or production aspects of an organization as well as supporting tasks. Traditionally, ERP systems were seen as an extension of manufacturing resource planning (MRP II) systems. ERP systems support the areas manufacturing, logistics, distribution, inventory, shipping, invoicing, accounting, human resources, and marketing. They are sources for fine-granular operative data representing all areas of operative business activity. Well-known product vendors are J.D. Edwards, Oracle, Peoplesoft and SAP.

Integrated enterprise architecture. Integration denotes (re)construction of a systematic whole. A reason for the need for integration is specialization. Though it is necessary to efficiently accomplish tasks and utilize resources, it often results in unnecessary and impeding boundaries between functional areas, processes or organizational units.

Enterprise application integration (EAI) comprises methods and technologies for unrestricted, integrated and automatic support of business processes within and between organizations by enterprise systems. It is a foundation for inter-organizational activities like e-procurement, electronic marketplaces and supply chain management (SCM). One reason for the strong interest in EAI is that many monolithic, single-purpose systems (legacy or “stove-pipe” systems) are not replaced because they are often the backbone of the organization’s operative business transactions and it is

considered not feasible or too risky to replace them. Thus, they have to be deployed and maintained together with newer systems.

Needs for integration

Additional reasons for integration can be seen in the following areas:

- *Mergers and acquisitions*: Mergers or acquisitions can be horizontal (companies of the same production level, similar branch, e.g., acquisition of a competitor), vertical (companies of successive production levels, e.g., acquisition of a supplier) or lateral (companies of different branches or production levels). They usually cause the need to integrate on different levels, e.g., in terms of strategy, personnel and IT infrastructure. On the level of IT, depending on the characteristics of IT systems and actual context, possibilities are to completely supplant the information systems of one partner, to work independently with different IT infrastructures or to connect both infrastructures on data, application or user interface level with the help of enterprise application integration.
- *Cooperations*: Different types of electronic communication and collaboration mechanisms are employed to realize, support or improve inter-organizational business processes. It usually makes it necessary to connect an organization's systems to external processes or data sources, because services regularly are realized in cooperation with business partners.
- *Business process improvements*: The integration of information systems often provides potentials for business process improvements. Integrated enterprise architectures built e.g., with the help of ERP systems and automation are possible ways to significantly improve internal business processes. The success of ERP systems depends on the joint support of internal business processes, either by providing comprehensive functions or by connecting to other internal systems. Automation of processes regularly makes it necessary to integrate tasks or process steps on a technical level.

Advantages of integration

The potential advantages of integration are:

- *Higher flexibility.* An organization can adapt more easily to changing business environments, unforeseen changes and different customer needs by faster reconfiguring its business processes and the software that supports these processes.
- *Cost reduction*. Processes are automated and unnecessary activities are avoided, e.g., the generation of information already stored in other systems of the company.
- *Better quality.* The quality of services and products increases in terms of lower response times, less errors through manual data input or better decisions through updated information.
- *Time savings*. Response or production times are shorter because internal processes can be automated and suppliers or customers are integrated

into the company's processes more tightly. It simplifies the development of new software or extensions of existing software and reduces the complexity of the development process.

Integration can be realized on all three layers of software architecture: data, domain and presentation layer. This depends on the actual challenges, characteristics of the infrastructure and capabilities of developers and IT-personnel.

Data layer

The goal of integration on the data layer is to build one common data base for different applications which formerly stored their data separately. Separate data storage often leads to data inconsistencies due to the danger that data entered or changed in one system is not updated simultaneously in all relevant applications. Another result of integration on this level can be a better understanding of an organization's data which can lead e.g., to the identification of unused cross-selling-potentials.

Domain layer

Integration on the domain layer targets the integrated support of business processes that involve multiple applications. Coordination of the applications and exchange of data between them should be automatic as much as possible and not mediated by humans, though human actors can fulfill tasks within the business process. Another goal of application level integration is to reuse software components (section 2.4, 136ff). One approach to implement domain-level integration is to insert an additional software layer to enable communication between applications, which is called middleware.

Presentation layer

Integration on the presentation layer is the most basic type of integration and often the last alternative if integration on lower layers is not possible, i.e. if no interfaces exist to access an application's data or its functions. It targets at presenting different applications or systems in a more or less consistent and standardized way to the user. The most prevalent approaches are screen scraping and Web integration using Frames or a Portal server. Screen scraping is a common way to integrate text-based interfaces of legacy systems. These terminals either are displayed directly with the aid of emulators or indirectly by interpreting data streams from mainframe to terminal to present it in a more user-friendly way. Web integration organizes the output of multiple systems in HTML pages displayed with Web browsers. This could be done rather simply, e.g., by displaying each system within a separate browser frame, or more advanced by using dedicated portal servers (section 5.1.1, 312ff).

SAP R/3, mySAP

Example: SAP Enterprise Services Architecture. By the end of 2003, the German software company SAP decided to undertake steps for reducing software maintenance and development costs of their products SAP R/3 and mySAP. SAP R/3 is a widely-used enterprise resource planning (ERP) software product based on the client/server model. MySAP is an e-business software integration tool for role-based delivery of contents

and application functions through a browser-accessible Web portal, e.g., for customer relationship management, supply chain management or human resource management (section 1.6.4).

Enterprise Services Architecture (ESA)

SAP decided to break their products up into smaller software components, each of them being accessible through Web service interfaces. The services together form a comprehensive Enterprise Services Architecture (ESA) based on the principles of a service-oriented architectures (SOA). SOAs promise more flexible and adaptable information systems and recently are discussed intensively in connection with Web services.

Definition of service-oriented architecture

A service-oriented architecture (SOA) is a collection of interacting independent services. A service is an activity that accomplishes a defined unit of work and is conducted by humans and/or machines. In the context of SOA, software services have to a) be *interoperable* in terms of platform independence, b) own a *network addressable interface*, and c) be *dynamically discovered* and *accessed*.

Challenges

However, SOAs reflect a vision and their implementation will induce major changes on products currently available from SAP and from other vendors. For SAP, the creation of an ESA is a very challenging task since SAP's ERP system R/3 was continuously extended over the years, which led to complex relations between system functions. Most likely, this demands re-development of large parts of the software in Java. It currently is implemented in ABAP (Advanced Business Application Programming), a proprietary programming language of SAP. It is not guaranteed that SAP will be successful with the concept of ESA. Skeptics warn that, even if one single Enterprise Service may be rather simple, the combination of many services for business processes and tasks may lead to an extremely complex and thus hardly manageable enterprise architecture.

SAP NetWeaver

SAP NetWeaver is an integrated application platform from SAP and foundation for the development of an ESA by providing a basic infrastructure for interaction between software components. Its architecture reflects the importance of integration issues (Figure 1-18). According to the architecture, integration happens at four levels

- *people*[3], i.e. integration of internal and external persons with the help of multi-channel interfaces and by e.g., portals (section 5.1, 312ff), as well as by tools for collaboration,
- *information*, i.e. integration of all kinds of structured (e.g., master data from operative systems) as well as unstructured information (e.g., documents) that are handled by a master data management and analyzed, used or created by means of business intelligence functions as well as

[3] Please note that people are not really integrated into the architecture. It is more an allegory that fits well into the marketing picture.

functions to manage unstructured data ("knowledge management" functions according to SAP),

- *processes*, i.e. integrated support of internal processes and connection with external processes and workflows, e.g., of suppliers (section 3.4, 203ff) and
- *applications*, i.e. an integrated application platform supporting multiple programming languages with the capability to expose functions as Web services.

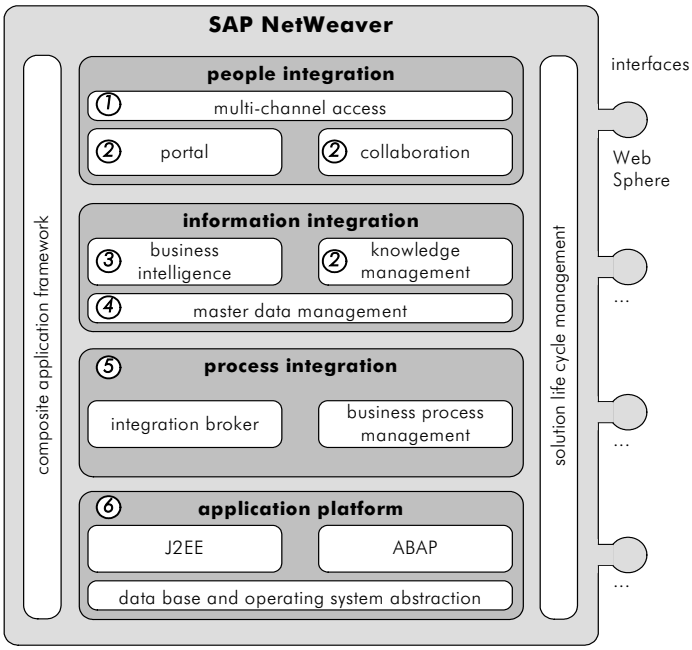

Figure 1-18. Architecture of SAP NetWeaver (SAP 2003)

Composite application framework, xApps

The composite application framework comprises tools, methods, rules as well as patterns and supports developers (of SAP or of solution partners) to create so-called packaged composite applications that use bundles of Enterprise Services for individual business scenarios and business processes. SAP is going to offer xApps, pre-configured applications for business processes.

Solution life-cycle management

The solution life-cycle management provides functions to manage software that is part of the NetWeaver architecture, i.e. installation, upgrade, monitoring, customizing and test of software. NetWeaver provides interfaces to systems implemented in other programming languages and from other product vendors.

Critique

This architecture reflects a view on integration that is strongly focused on products currently provided (or planned to be provided) by SAP. It should be noted that it does not reflect the perspective on integration of EKI, which will be discussed thoroughly in chapter 3, 147ff.

Currently, the following products provided by SAP can be mapped to the NetWeaver architecture (see numbers in Figure 1-18):

1. *SAP Mobile Infrastructure* for multi-channel online and offline access, e.g., with Web browsers or mobile devices like PDAs and Smartphones,
2. *SAP Enterprise Portal* for the handling of unstructured information and collaboration in teams,
3. *SAP Business Intelligence* to analyze structured data,
4. *SAP Master Data Management* for integrated data management,
5. *SAP Exchange Infrastructure* to integrate external and internal processes and applications and
6. *SAP Web Application Server*, a runtime environment for J2EE compliant Java and ABAP.

1.6 Knowledge Infrastructures

Enterprise knowledge infrastructures are implemented as part of a KM initiative that comprises a number of KM instruments. After our discussion of the terms knowledge, knowledge work, management and instrument, this section relates the concepts to EKI with the help of five perspectives on KM (section 1.6.1) as well as a number of distinctive characteristics of EKI (section 1.6.2) that finally lead to their definition (section 1.6.3).

1.6.1 Perspectives on Knowledge Management

Main modeling concepts

The design of KM initiatives requires joint consideration of the most important KM modeling concepts. These are (1) KM instruments, (2) organizational design, i.e. knowledge tasks and processes, roles and responsibilities, (3) persons, i.e. skills, communication and cooperation in networks and communities, (4) knowledge products and structures, i.e. type of knowledge, ontologies and meta-data and (5) ICT tools and systems, i.e. functions, architecture, structure and interaction of EKI (Figure 1-19).

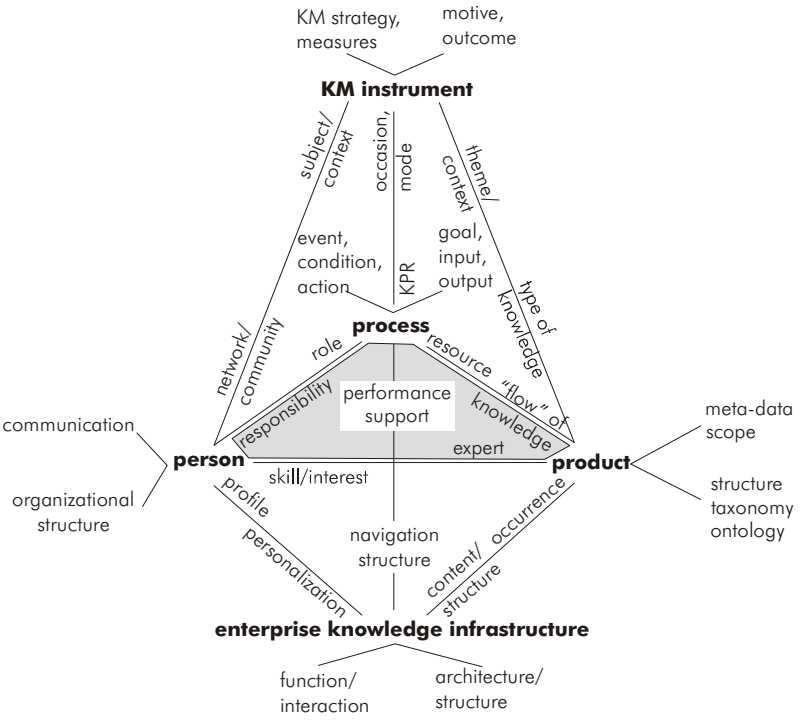

Figure 1-19. Perspectives on knowledge management

Connecting concepts

The five main perspectives KM instruments, person, process, topic and enterprise knowledge infrastructures, are connected by a number of concepts. Persons are involved in processes by responsibilities for activities and roles that are assigned to actions. Processes, especially activities and actions, are supported by EKI in order to improve organizational performance. Also, processes can be used to aid navigation in EKI. Topics are mapped to occurrences, e.g., documents or other resources that are stored in EKI. Structures, taxonomies and ontologies can be used as the primary structure of contents of EKI. Persons hold skills that are structured as topics. Persons and networks have interest in topics. Experts, e.g., subject matter specialists, take care of certain topics in organizations. Processes and topics are connected by the knowledge resources, both in the form of skills and in the form of documents, that are required in activities and actions and by the so-called "flow" of knowledge. This "flow" refers to an analysis of the knowledge life-cycle, i.e. in which actions or activities knowledge is created, distributed and applied. Profiles and personalization techniques are used to model persons to support their use of EKI. Finally, KM instruments link the context of persons, topics and EKI to certain steps in processes which provide occasions for knowledge-oriented tasks.

1.6.2 Characteristics

Knowledge management tools and systems

Generally, there are a number of approaches to define ICT that supports KM. This is reflected by a large number of different terms in use, such as knowledge infrastructure, knowledge management system, knowledge-based information system, knowledge management software, suite or support system, knowledge-oriented software, knowledge warehouse or organizational memory (information) system.

In addition to these terms meaning a comprehensive platform in support of KM, many authors provide more or less extensive lists of individual KM tools or technologies that can be used to support entire KM initiatives or certain processes, life-cycle phases or tasks.

Learning management systems

Another group of software systems supports these approaches called e-learning suite, learning management platform, portal, suite or system. These platforms not only support presentation, administration and organization of teaching material, but also interaction between and among teachers and students. KMS with roots in document management, collaboration or Groupware and learning management systems with roots in computer-based training already share a substantial portion of functionality and seem to converge or at least be integrated with each other.

Enterprise knowledge infrastructures

Enterprise knowledge infrastructures are organization-wide platforms that offer a joint workspace for collaboration, information, knowledge and learning to support knowledge work. In the following, we will summarize the most important characteristics of EKI (Figure 1-20).

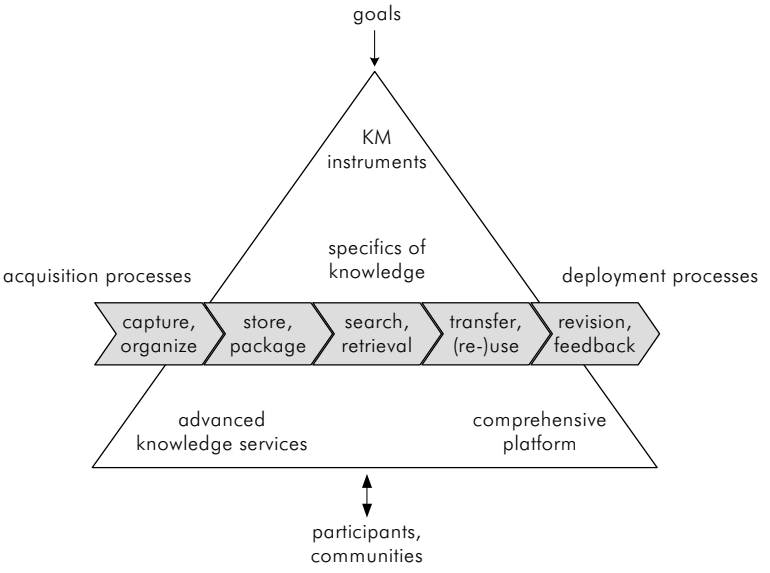

Figure 1-20. Characteristics of enterprise knowledge infrastructures

Goals

The use of this kind of systems aims at increased levels of effectiveness for the organization. Primary goal of EKI thus is to increase organizational effectiveness by a systematic management of knowledge. Thus, EKI are the ICT environment for effective knowledge work, the technological part of a KM initiative that also comprises person-oriented and organizational instruments targeted at improving productivity of knowledge work. The type of initiative determines the type of infrastructure for its support.

Processes

EKI are installed to support and enhance knowledge-intensive processes, tasks or projects of e.g., knowledge creation, organization, storage, retrieval, transfer, refinement and packaging, (re-)use, revision and feedback, also called the knowledge life-cycle, ultimately to support knowledge work. In this view, EKI provide a seamless pipeline for the flow of explicit knowledge through a refinement process, or a thinking forum containing interpretations, half-formed judgements, ideas and other perishable insights that aims at sparking collaborative thinking.

Comprehensive platform

Whereas the focus on processes can be seen as a user-centric approach, an IT-centric approach provides a base system to capture and distribute knowledge. This platform is then used throughout the organization. In this view, EKI are not application systems targeted at a single KM initiative, but a platform that can either be used as it is (if bought from a vendor, "out-of-the-box") to support knowledge processes or that is used as the integrating base system and repository on which specific KM application systems are built. Comprehensive means that the platform offers extensive functionality for user administration, messaging, conferencing and sharing of (documented) knowledge, i.e. publication, search, retrieval and presentation.

Advanced knowledge services

EKI are described as ICT platforms on which a number of integrated services are built. The processes that have to be supported give a first indication of the types of services that are needed. Examples are rather basic services, e.g., for collaboration, workflow management, document and content management, visualization, search and retrieval or more advanced services, e.g., profiling, personalization, text analysis, clustering and categorization to increase the relevance of retrieved and pushed information, advanced graphical techniques for navigation, awareness services, shared workspaces, (distributed) learning services as well as integration of and reasoning about various (document) sources on the basis of a shared ontology.

KM instruments

EKI are applied in a large number of application areas, e.g., in product development, process improvement, project management, post-merger integration or human resource management. More specifically, EKI support KM instruments, e.g., (1) the capturing, creation and sharing of best practices, (2) the implementation of experience management systems, (3) the creation of corporate knowledge directories, taxonomies or ontologies, (4) expertise locators, yellow and blue pages as well as skill management systems, also called people-finder systems, (5) collaborative filtering and

handling of interests used to connect people, (6) the creation and fostering of communities or knowledge networks, (7) the facilitation of intelligent problem solving. EKI in this case offer a targeted combination and integration of knowledge services that together foster selected KM instrument(s).

Specifics of knowledge

EKI are applied to managing knowledge which is personalized information related to facts, procedures, concepts, interpretations, ideas, observations, and judgements. Here, knowledge means information that is meaningfully organized, accumulated and embedded in a context of creation and application. EKI primarily leverage codified knowledge, but also aid communication or inference used to interpret situations and to generate activities, behavior and solutions. Thus, on the one hand EKI might not appear radically different from existing IS, but help to assimilate contextualized information. On the other hand, the role of ICT is to provide access to sources of knowledge and, with the help of shared context, to increase the breadth of knowledge sharing between persons rather than storing knowledge itself.

The internal context of knowledge describes the circumstances of its creation, e.g., author(s), creation date, assumptions or purpose of creation. The external context relates to retrieval and application of knowledge. It categorizes knowledge, relates it to other knowledge, describes access rights, usage restrictions and circumstances as well as feedback from its re-use. Contextualization is one of the key characteristics of EKI which provide a semantic link between explicit, codified knowledge and the persons that hold or seek knowledge in certain subject areas. Context enhances the simple "container" metaphor of organizational knowledge by a network of artefacts and people, of memory and of processing.

Participants

Communities or networks of knowledge workers that "own the knowledge" and decide what and how to share can provide important context. Meta-knowledge, also sometimes in the form of a set of expert profiles or the content of a skill management system, is sometimes as important as the original knowledge itself. Therefore, users play the roles of active, involved participants in knowledge networks fostered by EKI.

1.6.3 Definition

Enterprise knowledge infrastructures can thus be defined as follows.

Definition of enterprise knowledge infrastructure

An enterprise knowledge infrastructure is (1) a comprehensive ICT platform (2) for collaboration and knowledge sharing (3) with advanced knowledge services built on top that are (4) contextualized, integrated on the basis of a shared ontology and (5)personalized for participants networked in communities (6) that fosters the implementation of KM instruments (7) in support of knowledge processes (8) targeted at increasing productivity of knowledge work.

Detailed analysis of the definition

A definition of the term EKI has to focus on (1) a comprehensive platform rather than individual tools. As mentioned above, this platform targets (2) collaboration and knowledge sharing, contains (3) knowledge services based on (4) integration services that contain an enterprise-wide base ontology and/or standardized ways to map ontologies. Contents and services are (5) personalized for the users playing active roles in knowledge networks. The combination and integration of services realizes ICT-supported (6) KM instruments around which content and context of these systems revolve. The entire life-cycle of (7) processes of acquisition and deployment of knowledge has to be established. Also, an EKI has to be aligned with the specifics of its application environment, goals and type of KM initiative and ultimately aims at (8) improving the productivity of knowledge work.

Minimal requirements for EKI

Goals and processes describe the application environment of an EKI and therefore can only be used to judge an EKI that is applied and evaluated together with its organizational environment. In order to determine whether or not a tool or a system offered on the market or installed in an organization qualifies as an EKI tool or system from a system-centered view, the following minimal requirements have to be fulfilled:

Platform

The system has to provide an integrated set of basic functions for collaboration, document management, classification, visualization, search and retrieval. The system ideally has to support the entire knowledge life-cycle or at least a complete subset, e.g., capture, store, transfer, search and retrieval. The system has to be a multi-user system, scalable and theoretically be usable for a whole organization, i.e., several (overlapping) communities of users. This requirement excludes tools that focus a limited phase in the knowledge life-cycle, e.g., text analysis and knowledge mapping tools and excludes tools that help to organize an individual's knowledge workspace or a single group of knowledge workers.

Advanced knowledge services

The system has to provide more advanced services, at least for contextualization, integration of various (document) sources, personalization and workspace management. More specifically, EKI have to offer functions for the handling of meta-data, a shared taxonomy, text analysis, profiling, flexible navigation, shared workspaces, analysis and viewing of multiple (document) formats. This requirement excludes basic Groupware, document or content management systems.

KM instruments

EKI ideally offer default solutions for the implementation of a number of KM instruments, e.g., for skill management, experience management, best practice management, the support of communities or the development of corporate knowledge directories. Minimal requirement is that the EKI offers support for one document-oriented KM instrument and one person-oriented KM instrument. This requirement distinguishes EKI from systems offering advanced functions for purposes other than KM.

Specifics of knowledge

The system has to provide means to handle various types of knowledge including stable, documented knowledge, ad-hoc and/or co-authored

"live" experiences, the internal and external context of knowledge as well as information about skills and expertise of participants. EKI therefore reflect that knowledge is developed collectively and continuously reconstructed, revised and reused in different contexts. EKI ideally support all stages of knowledge from ideas, experiences, lessons learned, best practices to manuals, rules, procedures and patents. This requirement again contrasts EKI from more traditional document management or collaboration software.

Vision and available technologies

Apart from these minimal requirements, actual implementations of ICT systems certainly fulfill the characteristics of an ideal EKI only to a certain degree. Thus, a continuum between traditional IS and advanced EKI might be imagined with the minimal requirements providing some orientation. Still, a large number of ICT technologies is available on the market that together can be integrated to readily fulfill the vision of an EKI.

EKI and enterprise resource planning

EKI is a concept that can be seen as complementary to enterprise resource planning (ERP). Both concepts represent comprehensive ICT infrastructures. ERP focusses on the informational representation of business transactions as part of well-structured business processes or workflows on the administrative, dispositive as well as managerial level. EKIs provide a number of services to support any kind of knowledge work besides traditional business transactions and well-organized and well-supported business processes, e.g., in innovation projects, knowledge processes, knowledge-intensive business processes and the personal handling of experiences and semi-structured data. In contrast to traditional ERP systems, and similar to recent approaches there, EKIs are highly modular and consist of a number of building blocks that are partly common multi-purpose tools like Web or email servers and partly highly specialized knowledge-oriented tools like ontology inference or skill management systems. As in the case of ERP systems, not all parts of EKI are equally important for a single organization, so that EKIs might have to be customized to fit perfectly.

1.6.4 Architecture of Knowledge Infrastructure

This book thus lays out a comprehensive picture of what tools and technologies are already available to fulfill the vision of an enterprise knowledge infrastructure. Goal of this section is the description of an ideal, conceptual layered architecture that structures the ICT services to be provided by an EKI. This architecture is then used throughout the book as the structure guiding the presentation of numerous concepts, standards, technologies and tools that are available to implement parts of this architecture.

Ideal architecture

The ideal architecture can be seen as a vision: It poses requirements and asks for services to support knowledge work that an EKI *should* provide from the perspective of KM, rather than focusing on what current technologies are able to perform. They represent an amalgamation of theory-

driven, market-oriented and several vendor-specific architectures (for an analysis of these architectures and the process of amalgamation see Maier, 2004).

Nevertheless, appropriately combined existent approaches and technologies are able to provide a good foundation to support effectiveness of information and knowledge work. This book discusses current technologies needed to implement an EKI. This section also relates the vision incorporated by the ideal architecture to what is achievable with today's technologies and thus gives an overview of the structure of the book.

Layers

In the architecture, services provided by an EKI are offered by a central server that is accessed by the participants by means of (thin) clients. The system integrates all knowledge shared in an organization and includes a variety of services that are either aimed at supporting the participant directly or indirectly by supporting upward layers (Figure 1-21).

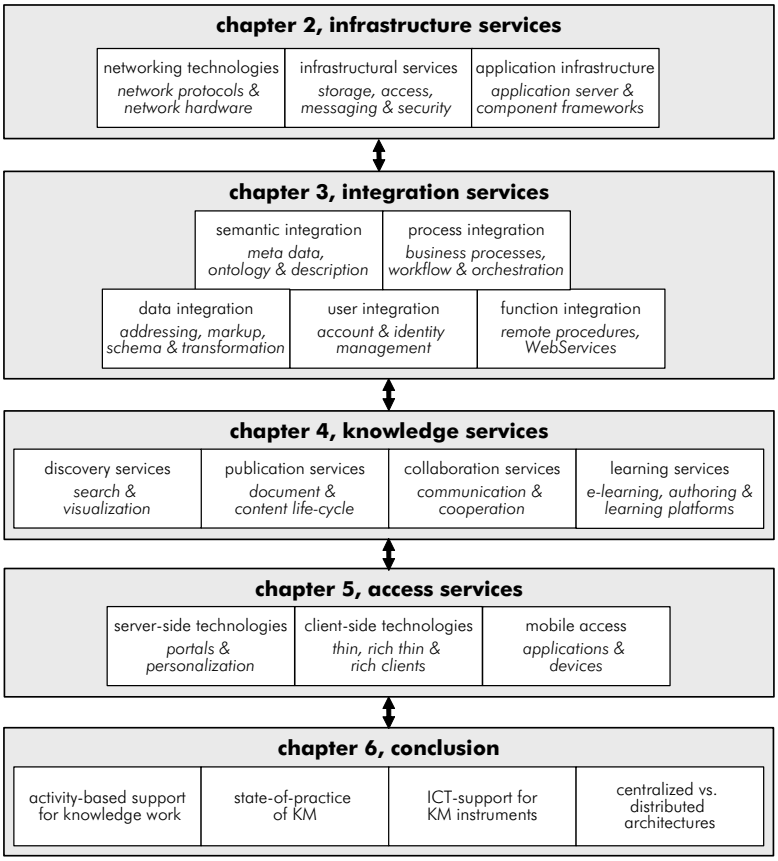

Figure 1-21. EKI layers and contents of following chapters

Infrastructure services

The Intranet infrastructure provides basic functionality for synchronous and asynchronous communication, sharing of data and documents as well as management of electronic assets in general and of Web content in particular. In analogy to data warehousing, extract, transformation and loading tools provide access to data and knowledge sources. Inspection services (viewer) are required for heterogeneous data and document formats.

Infrastructure services are discussed in chapter 2. *Networking technologies* enable communication between distant persons and technical devices and are discussed according to the network layers of the ISO/OSI-model up to application layer where basic network services and protocols are implemented. *Infrastructural services* comprise services for storage, messaging, security, access and encryption that EKI services of higher layers build upon. *Application services* offer common functionality needed by many, if not every application, regularly with the help of component frameworks.

Integration services

A taxonomy or an ontology help to meaningfully organize and link knowledge elements that come from a variety of sources and are used to analyze the semantics of the organizational knowledge base. Integration services are needed to manage meta-data about knowledge elements and the users that work with the system. Synchronization services export a portion of the knowledge workspace for work offline and (re-)integrate the results of work on knowledge elements that has been done offline.

Chapter 3 discusses how integration of data and semantics, functions and processes as well as data about users can be implemented. *Data integration* is strongly based on the implementation of standards that define character sets, addressing, markup, scopes and schema definitions. Examples are Unicode, XML and XML schema. The same is true for *semantic integration* via ontologies. Examples for important standards described in this chapter are RDF, RDF schema and OWL. A first step to *integration of users* is the centralized management of their accounts. Ultimately, EKI should manage user identities in a holistic approach, which is coined identity management. *Function integration* is necessary to enable remote execution of software and communication between (parts of) applications, e.g., by remote procedure calls or Web Services. *Process integration* through workflows and orchestration of services builds on function integration and aims at coordinating and automating tasks within business processes and of knowledge work.

Knowledge services

The core knowledge processes search and retrieval, publication, collaboration and learning are each supported by knowledge services. These are key components of the architecture and provide intelligent functions for:

- *discovery*, i.e. services for search, retrieval and presentation of knowledge elements and experts with the help of e.g., mining, visualization, mapping and navigation tools,

- *publication*, i.e. functions to support joint authoring, structuring, contextualization and release of knowledge elements supported by workflows,
- *collaboration*, i.e. support of joint creation, sharing and application of knowledge by knowledge providers and seekers with the help of e.g., contextualized communication and coordination tools, location and awareness management tools, community homespaces and experience management tools, and
- *learning* by authoring tools and tools for managing courses, tutoring, learning paths and examinations.

Knowledge services are the core of an EKI and are thought to offer complex services for the realization of KM instruments. Chapter 4 discusses approaches and technologies to implement knowledge services. Today, *discovery services* can either support querying of contents with keywords or exploring them with the help of appropriate visualization techniques like knowledge maps. *Publication services* aim at supporting the whole life-cycle of documents and contents, e.g, their creation, filing, (re-) use, publication, change and archiving. Predominant system classes described here are document and content management systems. The section about *collaboration services* discusses system classes and technologies commonly subsumed under the topic computer supported collaborative work (CSCW), especially support of synchronous and asynchronous communication as well as collaboration in groups. The section about *learning services* gives a short overview of some foundations of learning such as differing views on learning or learner types and then discusses tools and systems to create and use learning contents effectively.

Access and personalization services

Main aim of personalization services is to provide a more effective access to the large amounts of knowledge elements. A portion of the contents and services of an EKI can be organized for specific roles, e.g., with the help of role-oriented portals. Also, contents and services can be personalized with the help of e.g., interest profiles, personal category nets and personalizable portals. Automated profiling can aid personalization of functions, contents and services. The participant accesses the organization's EKI either with a thin client, e.g., a Web browser installed on his desktop or on mobile devices such as PDAs or smartphones, or indirectly by different applications and appliances where the EKI services are integrated into. On the server-side, a variety of services help translating and transforming the contents and communication to and from the system. Functions for authentication and authorization are necessary to protect EKI against eavesdropping and unauthorized use.

Chapter 5 shows how and by which communication channels participants can access knowledge services. They can either be provided in a *single point of access*, i.e. a portal that enables integrated and personalized access, or by mobile devices and applications for location-independent and

thus *mobile access*. Other services enable platform-independent, manageable access through Web browsers and rich clients or rich thin clients as well as by means of technologies that enable desktop integration and thus are a foundation for personal information management (PIM) and for enabling access according to new metaphors as an alternative to the "traditional" desktop metaphor.

EKI - vision and reality

This book gives a detailed introduction into the concept of EKI, its layers and building blocks. An ideal enterprise knowledge infrastructure that substantially improves knowledge work throughout and beyond the organization is partly a vision that is not yet reality in organizations, although there are some innovative companies that have implemented individual productivity environments that come close to this vision. The vision, mainly based on a conceptual view of knowledge work and knowledge management, is reflected in the main structure of the book (levels one or two of the table of contents). But there are many existing classes of technologies and software systems that cover one or more of the layers or building blocks of EKIs and appear in the detailed structure of the book structure (level three). It is important to understand the relation between the functionality typically provided by a certain class of software and the requirements for an EKI building block. Chapter 6 reflects on the potentials of already existing technologies to fulfil the vision and gives an outlook to future developments. So after completion of this book you will be able to describe the most important technologies used to support knowledge work today, assess their typical functionality and classify them in relation to the conceptual architecture of EKIs.

Questions and Exercises

1. What are the most important dimensions of knowledge? What opposites can be distinguished in each of the dimensions? Find examples for knowledge that reflects each of the opposites!
2. Which fundamental perspectives on knowledge can be distinguished? What consequences has that for knowledge management? Describe examples of concrete goals and measures for each of the perspectives!
3. Discuss the relationships between the terms capability, expertise, intellectual capital, knowledge and skill. Find examples for each of these terms!
4. Sketch Nonaka's processes of knowledge conversion and find supporting KM instruments for each of the processes!
5. Discuss the definition of the term knowledge management! What lines of development can be distinguished?
6. Discuss whether and to what extent the following positions fulfill the characteristics of knowledge work: air traffic controller, hair dresser, medical doctor, nurse, sales person responsible for selling PCs in a local PC store, service technician for household appliances (e.g., dish washer, microwave). Which types of knowledge are focussed in each of these positions?
7. What information and communication technologies could support the implementation of knowledge management instruments? Find examples for one instrument out of each class!
8. Discuss how the term architecture can be used with respect to an organization's information and communication infrastructure!
9. What are the main advantages of layered architectures?
10. What is an enterprise knowledge infrastructure and how could it support knowledge work?

Further Reading

There are a large number of books and articles on knowledge management and related topics. The following list provides in no way an exhaustive list, but only attracts the reader to some works that provide an overview of many concepts.

Alavi, M., Leidner, D. E.: *Review: Knowledge Management and Knowledge Management Systems: Conceptual Foundations and Research Issues*. In: Management Information Systems Quarterly - MISQ 25(1) 2001. 107-136

A comprehensive overview and discussion of the definitions of knowledge management and knowledge management systems that also offers some important implications for further research.

Holsapple, C. W. (ed.): *Handbook on Knowledge Management*. Vol. 1+2. Berlin et al. 2003

The book draws together a large number of authors and perspectives on knowledge management spanning strategic, organizational, and technological issues and also provides some case studies that show knowledge management in action.

Maier, R.: *Knowledge Management Systems. Information and Communication Technologies for Knowledge Management*. 2nd edition. Berlin et al. 2004

An almost encyclopedic treatise of the facets, concepts and theories that have influenced KM is presented together with the state-of-practice on the basis of a comprehensive empirical study. The book furthermore integrates the concepts, theories and practical findings into a general KM framework consisting of strategy, organization, systems and economics.

Mertins, K., Heisig, P., Vorbeck, J. (eds.): *Knowledge Management. Best Practices in Europe*. Berlin et al. 2001

The book comprises practical instruments, techniques and tools that were developed in the course of a number of projects in organizations implementing knowledge management.

Nonaka, I., Takeuchi, H.: *The Knowledge Creating Company.* New York 1995

A classic in knowledge management and still an interesting read for those that are interested in the characteristics of the approaches implemented in Japanese organizations that helped to create the hype on knowledge management that spread in the 90s.

2 Infrastructure

Overview

Infrastructure is a term that is used for a variety of things. The most common contexts for the term in everyday life are roads, water and wastewater, electricity and telecommunication. The basis for an EKI is the network infrastructure. This starts with laying cables into the floors and/or the walls of organization's buildings or setting up base stations for a wireless network. Today, a medium-sized company building needs several kilometers of cable in order to provide a good network infrastructure for every workplace. Besides the computer network for data communication today there is also still a telephone network for voice communication that forms a separate infrastructure. Although both forms seem to amalgamate in the near future, as current IP telephony and data communication over telephone lines indicate, we will concentrate our considerations on computer networks. Theoretical foundations for computer networks including a layered architecture are presented in section 2.1. Section 2.2 then discusses standard network protocols that are implementations of the layers. Based on the basic network infrastructure, services for data storage, access, messaging and security are offered (section 2.3). More recently, many organizations have also installed an application infrastructure (section 2.4) that provides services needed in most enterprise applications (e.g., transaction support).

Learning objectives

On completion of this chapter you should be able to

- describe the most important network standards for PANs, LANs and WANs,
- use a conceptual model that distinguishes several layers within a network that provide different services and build upon each other,
- identify the most important network protocols for the Internet and categorize them according to layers in the model,
- distinguish several network devices and explain their usage,
- describe the most important technologies for storage, access, messaging and security,
- explain the functionality of application servers and explain how they support enterprise applications.

2.1 Network Infrastructure

A classification of computer networks helps to understand the features networks can provide. We will first present a list of several classification criteria before we discuss two criteria, topology and geographical expansion, in detail. The section concludes with illustrating the ISO/OSI reference model, a layered architecture for computer networks.

Classification of computer networks

Computer networks can be classified according to a variety of criteria (Table 2-1). The criteria focus either on technical and physical aspects, e.g., the way of transmission, type of media, topology and scale of a network, or on organizational aspects, i.e. who owns and operates the network or who is allowed to access and use the network.

Table 2-1. Criteria for classifying networks

criterion	description	example
transmission	mode of transmission	broadcast, point-to-point
topology	basic structure of the physical network	ring, bus, star
scope	geographical expansion of the network	PAN, LAN, WAN
network owner	institution that operates the network	company network, Internet, Value Added Network
user group	group of users that has access to the network	Internet, Intranet, Extranet
function	purpose of the network	front-end network, back-end server network, backbone
media	type of physical media	fiber optics, copper cable, infrared light, radio
transmission procedure	structure of messages, message passing, etc.	Ethernet, Token Ring, ATM
performance	network bandwidth	low (< 1 MBit/s), medium, high (> 1 GBit/s)

Other criteria shown in Table 2-1 will be discussed in conjunction with network standards in section 2.2 and infrastructure services in section 2.3.

2.1.1 Topologies

Machines linked in a network communicate by sending messages through a communication medium. In analogy to the traditional mail system these are called packets. The actual content of the message is wrapped in transport data about sender and receiver and data to control the transport process. In this section, alternatives to structure communication networks are explained with regard to physical connections between machines. A topology represents the physical communication connections (edges) between machines (nodes) in a network.

Broadcast networks

The architecture of a network is fundamentally determined by the mode of transmission which can be broadcasting of messages or point-to-point communication. In broadcast networks, all nodes are connected to the same physical medium. Messages sent by one machine are received by all others, although they can either be addressed to one single communication partner (*unicast*) or to all nodes in the network (*broadcast*). Some networks allow addressing of messages to a subset of the network which is termed *multicasting*. A main challenge in broadcast networks is how to utilize the network medium most efficiently which largely depends on management of media access.

Point-to-point networks

In point-to-point networks (sometimes also called *peer-to-peer* networks), only pairs of machines are connected through a physical medium. Distant communication partners thus regularly will not be connected directly to each other. In this case, a message needs to be transmitted via intermediate stations that receive and forward the message to its destination. A challenge in this class of networks is how to find the best out of multiple alternative routes that connect source and destination of the message. In contrast to broadcast networks, point-to-point networks are only capable of *unicasting*.

Topologies of broadcast networks. Bus and ring networks are two basic types of topologies of broadcast networks which are often combined in practice, e.g., a ring network connects multiple bus networks.

Bus network

The left hand side of Figure 2-1 shows a bus network. All nodes are connected to a shared communication medium, e.g., a copper cable, that passively transmits messages in both directions. No routing or forwarding of messages is necessary because every connected station receives all messages.

Ring network

In a ring network, every node is connected to exactly two other nodes. A message is passed only in one direction from one node to the other until its destination is reached. If one station breaks down, the whole communication is blocked. To enhance the reliability of the network, redundant communication channels can be established that allow bypassing defect machines. The nodes usually regenerate the physical network signal with the effect that network size is not limited with respect to the number of

participating machines, but by length of the connections between two nodes.

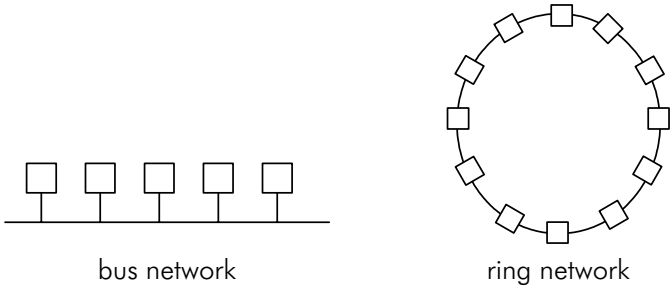

Figure 2-1. Topologies of broadcast networks

Topologies of point-to-point networks. Four basic architectures of point-to-point networks can be distinguished (Figure 2-2):

Star network In a star network, all nodes are physically connected to a central node that handles the entire communication between all nodes. The advantage of this topology is that maintenance and control of the network are simple, because it can concentrate on the central node. A major disadvantage is the dependence of the whole network on performance and availability of the central node.

Tree network In a tree network, communication between two nodes always runs over hierarchically superordinated nodes. This architecture thus can be seen as to be composed out of interconnected star networks. Network control is performed by superordinated nodes. Thus, the tree network has the same advantages and disadvantages as the star network: easy administration, but dependence on central nodes.

Mesh network In a mesh network, every node is connected to two or more other nodes. If all nodes are directly connected to each other, we speak of completely intermeshed networks. Mesh networks are very reliable and can grow without central control. A disadvantage of this architecture is that it can be complex to find the best route between source and destination.

Loop network In a loop network, every node is connected to exactly two other nodes. Every single node controls network traffic. Thus, network control is more complex than in the case of centralized network architectures such as star or tree networks. In contrast to a ring network, communication between nodes can happen in both directions and communication media not necessarily have to be of the same type. The nodes in a loop network play a more active role than in a ring network as they forward messages and not just refresh physical signals.

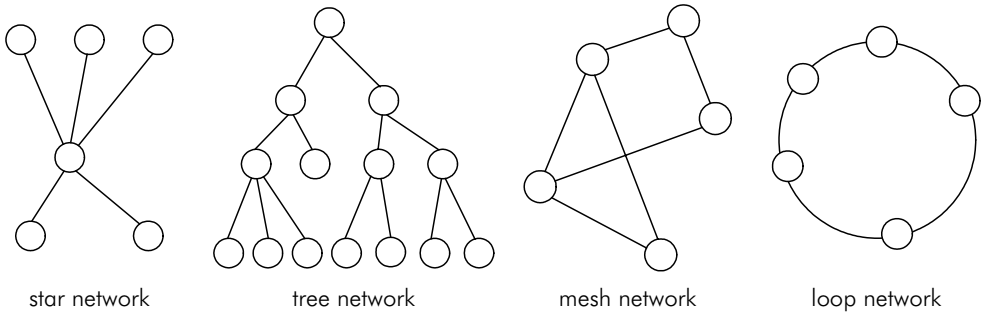

Figure 2-2. Topologies of point-to-point networks

2.1.2 Geographical Expansion

Networks can also be classified according to their scope, i.e. the geographical area they cover. Table 2-2 gives an overview of commonly distinguished classes of networks with respect to their geographical scope.

Table 2-2. Network classification according to scale

interprocessor distance	location examples, network for ...	network class
1 m	workplace	personal area network (PAN)
10 m	conference room	
100 m	company building	local area network (LAN)
1 km	university campus	
10 km	city	metropolitan area network (MAN)
100 km	country	wide area network (WAN)
1000 km	continent	
10.000 km	planet	the Internet

Personal area network (PAN)

Personal area networks (PANs) connect devices of a single person. Often, the term implies the use of some form of wireless technology to connect e.g., personal digital assistants (PDA), notebooks and cellular phones, or to connect a PC to a printer and a scanner.

Body area network (BAN)

The term body area network (BAN) is sometimes used to denote very short ranging personal area networks between components of a wearable computer, e.g., computer system that is integrated into a jacket.

Local area network (LAN)

Local area networks (LANs) are used to share hardware and software resources within a workgroup or an organization. They are restricted in

size and usually span a single building, site or campus. Although geographic expansion of small LANs and wide reaching PANs may overlap, there is a clear distinction as LANs connect several computers belonging to different people, e.g., in a work group whereas PANs connect several devices belonging to one person.

Metropolitan area network (MAN)

A metropolitan area network (MAN) covers a city or region and either interconnects separate LANs of organizations or provides centralized services like Internet access or cable television to private homes. A recent development is high-speed wireless Internet access via MANs (IEEE 802.16 standard, UMTS).

Wide area network (WAN)

A wide area network (WAN) enables communication with very high bandwidth (e.g., 10 GBit/s) over large distances, e.g., between states, countries or continents. It basically consists of hosts, e.g., servers, switches, routers, and a communication subnetwork that connects them. The network usually is owned and operated by a telephone company, an Internet service provider or by public authorities. Challenges in WANs are bandwidth management, cost accounting, scalability and high reliability.

Internet

Last, but not least the Internet is the network of interconnected networks spanning all continents of our planet (section 2.3.4, 124ff).

2.1.3 Layered Network Architecture

ISO/OSI reference model

Computer networks can be seen as hierarchical systems with several layers. Each layer provides certain services to the higher layers which leads to an increasing degree of abstraction for higher layers. The higher layer can therefore be recognized as a service consumer whereas the layer below can be seen as a service provider. The specification of the service is the interface between both layers. There exist a number of layered architectures of which the ISO/OSI reference model is the most important one from a theoretical perspective. The Open Systems Interconnection (OSI) model has been defined by the International Standards Organization (ISO). It specifies seven layers that help to understand network systems (Figure 2-3).

The physical communication in this model is top-down from the highest to the lowest layer on the sender side, then horizontal over the network media and then bottom-up from the lowest to the highest layer on the receiver side. From a conceptual point of view, each layer on sender side communicates with the layer on the same level on receiver side.

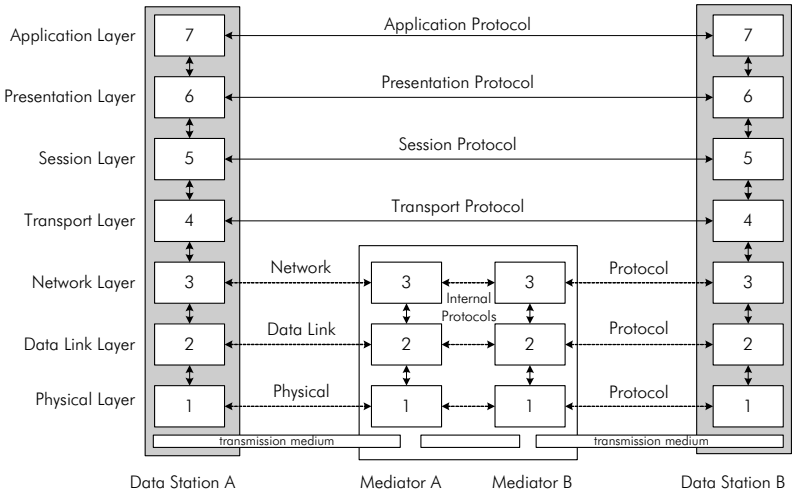

Figure 2-3. ISO/OSI reference model

Example

Imagine the owner of a small company in Wellington, New Zealand, who recently met a business man from Munich, Germany, on an industry event in the USA. When she comes home from the event, she wants to send him a message, but she does not know his address details. All she knows is the name of the man and the company he works for. (a) So she gives her secretary the message she wants to send and the instruction to transfer it as soon as possible. (b) The secretary looks up the address details, puts the message into an envelope and puts the address on the envelope. She is not sure about the correct value of stamps she has to put on the envelope, so she gives the letter to the post office of the company. (c) In the post office, they look up the rate and put stamps on the envelope. Then, they hand it over to a logistics provider that handles all the letters and parcels for the company. They do not know on which way the letter will travel to Germany, nor do they care. (d) The logistics provider handles that on its own. The letter goes by truck to the central post office and from there to Auckland airport. The plane with the letter leaves to Singapore, which is a central trade center for passengers as well as goods and flies to Frankfurt/Main airport in Germany. The message moves on to Munich by train until it reaches the post office of the company at its destination. (e) There, the clerk reads the address on the envelope and brings it to the secretary of the receiver (f) who removes the envelope and finally (g) hands it over to the business man.

The company owner back in New Zealand thinks that she has communicated directly with the business man in Germany. In fact, she has only communicated directly with her own secretary.

ISO/OSI layers

The same system applies to the ISO/OSI reference model. Each layer has a distinct purpose that will briefly be described in the following section. The application layer (7) gets data that has to be communicated from the application. The presentation layer (6) is concerned with syntax and semantics of transmitted data. It uses high order data structures (e.g., bank account records). The session layer (5) handles session information and time-outs, so that user-specific information is only saved as long as applications need it. The transport layer (4) provides a virtual channel for communication that can either be connection-oriented or connection-less. The former means data can rely on data passed on earlier. The latter means data is transmitted according to fire-and-forget mechanisms. Connection-oriented transportation can be thought of as having the telephone system as role model. A user picks up the phone, dials a number, communicates with the remote person and hangs up. Connectionless transportation on the other hand can be thought of as having the postal system as role model. Each message/letter carries the full address and is routed through the system independently. The advantage of connectionless transportation is fault tolerance, whereas connection-oriented transportation allows for billing and a guaranteed level of quality (quality of service, QoS). The network layer (3) is responsible for identification of communication partners which is implemented via unambiguous identification addresses. The data link layer (2) takes care of reliable reconstruction of the transmitted signal by adding checksums to the data. Finally, the physical layer (1) provides access to the transmission medium by modulating the signal.

Mediators

In between two data stations, there can be several mediators that refresh the signal and route the network packets. They may use different protocols to communicate with each other. Real end-to-end communication is only established on layers four and above.

2.2 Network Standards

The following sections discuss concrete implementations for the different layers of the ISO/OSI model. These implementations are standardized network protocols. Afterwards, we also present network hardware which can be categorized according to the ISO/OSI layers they use.

2.2.1 Physical and Data Link Layer

Media, cable, fiber, copper

Starting from the bottom, transmission media which can be wired or wireless have to be examined. For wired media, fibre channel and copper cables can be distinguished with copper cables being further divided into twisted pair, coaxial and power cable. The different types of copper cables

vary in cross-section and shielding, which are the main characteristics that influence attenuation and liability to interference. Today, copper is mainly used for end-user connectivity. Most long range cables are fiber optical because of better transmission characteristics. With more and more fiber optical cables produced, costs decrease so that this material becomes affordable and is also used for connecting servers in order to satisfy increasing bandwidth demands. This leads to a further increase in production and decreasing costs so that it seems only a matter of time until fiber optical cables are used to connect end-user computers (classification according to function in Table 2-2 on page 87).

Wireless media, radio, optical

Wireless media can be divided into radio connections and optical connections that differ not only in wavelength of the signal (1-5 GHz for radio vs. 3-300 THz for optical connections), but also in their diffusion model. Where optical methods usually use point-to-point connections, radio frequency emitting devices usually send in all directions (broadcast). Both types have two important sub-types: terrestrial radio transmission and satellite transmission for radio connections as well as infrared light and directed laser for optical connections.

Table 2-3 gives an overview of the network standards discussed in the next section with respect to geographical expansion of networks explained in section 2.1.2, 87ff.

Table 2-3. Overview of network standards

	cable-bound	**wireless**
PAN	USB, Firewire	IrDA, Bluetooth
LAN	Ethernet, Token Ring	WLAN, DECT
WAN	ATM, FDDI, X.25, FrameRelay, Sonet/SDH	GSM, GPRS, EDGE, HSCSD, UMTS

USB

Protocols for personal area networks. There are two important standards each for wired and wireless connections between devices in personal area networks. The purpose of these standards is to provide easy connectivity, low implementation costs and small yet robust physical interfaces. For wired connections, USB (universal serial bus) is the dominating standard and can be seen as a successor of the serial RS/232 interface. It is used to connect peripheral devices, e.g., pointing devices and keyboards, as well as digital cameras, scanners and printers to computers. It can also be used to connect peripheral devices with each other (USB2go), although this is still rarely the case. USB version 1.1 allows transmission with up to around 1 MBit per second. Version 2.0 is downwards compatible and specifies transfer speeds up to 480 MBit per second.

Firewire

A competing standard for USB 2.0 is the IEEE specification 1394, commonly known as iLink (Sony) or Firewire (other companies). It is mainly

used for connecting external high-speed devices like DVD burners or hard disks to computers (especially by Apple Computer, Inc.) and in the video industry for connecting digital video camcorders to computers. The standard can be used for transfer speeds up to 400 MBit per second. A successor with doubled transfer speed has recently been specified.

IrDA

Wireless connections in PANs are still mainly realized via infrared light signals based on the IrDA standard (Infrared Data Association). Most devices available only support version 1.0 that provides transfer rates of 115 kBit per second. There are also two faster versions that provide 4 MBit per second (version 1.1, also called Fast IrDA) and up to 16 MBit/s (version 1.2, sometimes called Very Fast Infrared, VFIR). However, both are rarely used. All versions have a signal range of 1-2 meters. Mobile phones are the largest group of devices that heavily rely on IrDA for wireless connection to computers and especially notebooks. Optical connections rely on a direct line of sight between both communication partners. This leads to easy interruptions in the communication, e.g., by shakes that move or rotate the mobile phone just a little so that connection to the computer gets lost.

Bluetooth

Bluetooth, a radio standard for PANs, becomes more and more adopted, because radio transmissions are much easier to handle. It is designed to facilitate any kind of wireless connections, e.g., connect keyboards and mice to computers, connect headsets to mobile phones, or connect digital cameras to printers in order to directly print photos. Bluetooth was specified by the Bluetooth Special Interest Group (BSIG) initiated in 1998 by Ericsson, IBM, Intel, Nokia and Toshiba. It is designed to connect devices within a 10-100 meters range with 1 MBit/s transfer rate and operates at 800 mW transmission power in the 2.4 GHz frequency band. It is a technique for ad-hoc connection of devices where one device declares itself as master and up to seven other devices can connect as slaves and form together a so-called *Piconet*. Devices can also take part in more than one Piconet simultaneously. The resulting overlapping Piconets form a so-called *Scatternet*. To support such different deployment scenarios as described above, every device has to support a number of profiles. A profile specifies a concrete set of protocols that span one or more layers. The profiles are divided into basic and advanced profiles (Table 2-4).

Despite its increasing dissemination, there are still some interoperability problems between devices of different vendors that are a result of inaccurate specification or sloppy implementation. Therefore, testing whether two devices interoperate is principally advisable, even if both devices support the same profile. Bluetooth version 2.0 has recently been specified and provides faster transfer rates and a larger range.

Table 2-4. Examples for Bluetooth profiles

type	profile	typical purpose (connect ...)
basic	generic access	discovery of remote devices and services
	service discovery	discovery of supported profiles
	dial-up networking	notebook to mobile phone (Web access)
	LAN access	PDA to WLAN access point (PPP)
	generic object exchange	calendar items between PDAs (OBEX)
	synchronization	PDA to PC
advanced	basic imaging	scanner or digital camera to PC
	hands free	headset to mobile phone
	hardcopy cable replacement	PC to printer (formatted text and images)
	human interface device	mouse and keyboard to PC
	local positioning	GPS device to PDA

Ethernet, IEEE 802.3

Protocols for local area networks. The most important and most widely used technology for local area networks is Ethernet (IEEE 802.3). It is a cable-bound transmission standard developed by Xerox PARC in 1970, implements the bus topology (section 2.1.1) and supports different cable types. The Ethernet specification covers layer one and two of the ISO/OSI reference model (physical layer and data link layer). The transfer rate of the first Ethernet specification approved by the IEEE in 1985 is 10 MBit/s (10BaseT), which is not very comfortable for end user PCs any more. Servers that usually handle multiple connections to several end-users simultaneously need even more bandwidth. Thus, two other specifications with 100 MBit/s (100BaseT, Fast Ethernet) and 1 GBit/s (1000BaseX, Gigabit Ethernet) bandwidth have been developed. Recently, a 10 GBit Ethernet standard has been specified for usage in backbones. Modern company Intranets typically run 100BaseT with some servers already connected via Gigabit Ethernet. The advantage of Ethernet is that all components necessary to establish a network (like NICs, cable, switches, section 2.2.4, 106ff) are reasonable priced due to their wide-spread use.

CSMA/CD

Ethernet uses a CSMA/CD (carrier sense multiple access with collision detection) method to access the medium, which is a competitive access method and operates as follows. The cable is checked for network traffic. The adapter sends if no traffic can be sensed, otherwise it waits for a random time interval until it tries again. After a signal has been sent, the cable is further checked for collisions with other signals sent simultaneously by other data stations. A strong jamming signal is sent if a collision is detected so that all connected devices are informed about the collision.

Token Ring, IEEE 802.5

Token Ring (IEEE 802.5) is a network standard of theoretic interest due to its different mode, although it is not used that much any more. It has been developed by IBM and implements a ring topology (section 2.1.1). A token is passed on from one data station in the network to the next one. Only stations that currently have the token are allowed to send. The first computer in the network that goes online generates the token. This is an example for a coordinated access method. The ring topology is only a logical structure and describes the way of the token. Physically, all data stations can also be connected with each other in a star structure. Supported transfer rates are between 4 and 16 MBit per second. Token Ring was popular 20 years ago but is increasingly replaced by Ethernet networks.

WLAN, IEEE 802.11

Wireless LAN (WLAN, IEEE 802.11 family) is the wireless equivalent to Ethernet. It is radio-based and has a range of approximately 30 meters within buildings and up to 300 meters outside. WLAN is a group of standards that operate in the 2.4 GHz frequency band (802.11b and 802.11g) and in the 5 GHz band (802.11a) respectively. It supports transfer rates of 11 MBit/s (802.11b) and 54 MBit/s (802.11a and g). The most wide-spread standard is still 802.11b, but 802.11g is catching up quickly. Those three standards are supported by a set of extensions, adaptations and corrections (802.11d, e and f) as well as security standards (802.11i).

WEP, WPA

With good reason, security is still one of the main concerns for WLAN implementations. In contrast to cable-bound networks, WLAN communication signals can easily be received by anybody in reach of the radio transmission. The first security protocol designed for WLAN that should prevent network intrusion was *WEP (wired equivalent privacy)*. However, it is considered insecure, since default configuration of most WLAN access points has WEP turned off, WEP passwords have to be manually configured for every single device and default key length is only 56 Bit. Therefore, an IEEE working group was established (802.11i) that presented a new security protocol named *WPA (WiFi protected access)*, with 128 Bit keys (or higher) and AES encryption (section 2.3.4, 124ff).

Roaming

Another challenge in WLANs is hand over from one access point to the next one when users move. This endeavor is called roaming (not to be confused with roaming in cellular phone networks) and should be as smooth as possible without disturbing the network connection.

CSMA/CA

The only obligatory method for medium access that every WLAN implementation has to have is CSMA/CA (carrier sense multiple access with collision avoidance). As in Ethernet networks, the number of collisions rapidly grows with an increasing number of users on the same access point. Therefore, advanced methods like *RTS/CTS* (request-to-send/clear-to-send) and *PCF* (point coordination function) have been developed to overcome these problems. Since they are not mandatory for compliance to the WLAN specification, only a few manufacturers implement them in their devices.

DECT

DECT (digital enhanced cordless telecommunications) is a telephone standard for wireless telephones within company buildings, exhibitions or at home. It must not be confused with cellular phone standards like GSM (see "Protocols for cellular mobile networks" on page 96) that have a much larger range. DECT is designed for private use, with every household operating its own base station (about 100 €) whereas GSM base stations are operated by infrastructure providers (a GSM base station is about 10,000 €). DECT uses the frequency band between 1880 and 1990 MHz.

FDDI

Protocols for wide area networks. FDDI (fiber distributed data interface) is a network protocol mainly used for medium distances, e.g., in the backbone of a university spread across a city. As the name suggests, it uses fiber optical cable as medium. It employs a ring topology similar to Token Ring with a second redundant ring to guarantee availability of the network. Compared to data rates that are supported with Ethernet these days, FDDI seems to be relatively slow with data rates of 100 MBit/s for normal FDDI or 200 MBit/s for the full duplex version that uses the second ring simultaneously to the first. A successor called FDDI-2 better supports audio and video transmissions by providing reserved capacity and faster latency times (125 µs). The ring can be up to 100 km long with a maximum distance of 2 km between adjacent network stations.

X.25

X.25 is an example for a connection-oriented network. It was the first public data network (since the 1970s) and was widely used in the 1980s. Nowadays, use of X.25 for general networking is declining, but it remains important in point-of-sale credit card and debit card authorization. However, huge amounts of money have been invested in X.25 infrastructures throughout the world and in some regions, it continues to expand. X.25 permits a network station to communicate simultaneously with a number of remote stations.

Switched and permanent virtual circuits

Connections in X.25 can occur on logical channels of two types:

- *Switched virtual circuits* (SVCs) are very much like telephone calls: a connection is established, data is transferred and then the connection is released. Each station on the network is given a unique address which can be used much like a telephone number.
- *Permanent virtual circuits* (PVCs) are similar to leased lines in that connection is always present. Logical connection is established permanently by the packet-switched network administration. Therefore, data may always be sent, without any call setup.

X.25 is optimized for what today would be considered low speed lines of 100 kBit/s and below.

Frame relay

A connection-oriented network that is widely replacing X.25 is frame relay. It is mainly suited for use in WANs, but can also be used to interconnect LANs, e.g., offices of a company in different cities can be connected in a cost-effective way by providing a secure private IP-based network in

contrast to connections over the Internet. Frame relay has no error or flow control and thus is quite simple. It is packet-switched and mainly realized as PVC, although the specification also supports SVC scenarios. The data rates reach from originally 56 kBit/s (V.34) up to 1.544 MBit/s (T1). Some providers also use special frame relay variants with 34 MBit/s or 45 MBit/s (T3).

ATM

A wide-spread protocol is called ATM (asynchronous transfer mode). It was designed to merge voice, data and cable-television networks and stands out due to its high data rates ranging from 155 MBit/s (OC-3) and 622 MBit/s (OC-12) up to 2.5 GBit/s (OC-48) that are in operation. There are also some 10 GBit/s lines (OC-192) in limited use, spanning up to 40 GBit (e.g., OC-48 with Wave Division Multiplex) in trials. One reason for the possibility of such high speeds is that ATM uses small, fixed size data packages called cells (53 byte) that can be routed in hardware (5 byte header). ATM provides flow control, but no guaranteed delivery due to lacking error control.

SONET/SDH

The commonly used protocol within public switched telephone networks (PSTN) is called SONET (synchronous optical network). There is also a set of CCITT recommendations called SDH (synchronous digital hierarchy) which is similar to the SONET standard and will not be distinguished here for simplicity reasons. SONET has been designed to unify and replace the former PSTN protocols that were all based on 64 kBit PCM (pulse code modulation) channels (ISDN) and combines them in different ways to get high speed connections (T1 equals 24 ISDN channels). A SONET frame is 810 bytes long and is sent 8,000 times per second. Since SONET is synchronous, frames are sent whether there is useful data to be sent or not. Sampling frequency of PCM channels in all digital telephone systems is exactly 8,000 Hz, so that there is a perfect match. A basic SONET channel, called STS-1 (synchronous transport signal, called OC-1 on optical media) has a data rate of 51.84 MBit/s (8,000 frames/s * 810 bytes/frame * 8 bits/byte). SONET channels can be combined to yield higher data rates. There are specifications for a number of speeds up to STS-192 (9.953 GBit/s).

Protocols for cellular mobile networks. Cellular mobile networks initially designed for voice telephony are increasingly used for data transfer as well.

GSM

The most widely used standard for mobile telephony world wide is GSM (*global system for mobile communication*). In February 2004, only 12 years after the launch of the first networks, more than one billion people (almost one sixth of the world's population) were using GSM mobile phones. GSM networks are digital and represent the 2nd generation of mobile networks, with the first generation represented by analogue networks like the A, B and C networks since 1958. GSM is a circuit-switched data (CSD) network and therefore connection-oriented as usual for net-

works designed for voice telephony. Data rate is 13 kBit/s for voice and 9.6 kBit/s for data connections which sums up to 22.8 kBit/s over all. The frequency range in Europe originally was 890-915 MHz for upload and 935-965 MHz for download known as GSM 900. More recently there is also a frequency range around 1800 MHz in Europe whereas the 1900 MHz band is used mostly in the US. GSM is a cellular network with cell sizes between 300 m in densely populated urban areas and 35 km in rural areas. Adjacent cells use different frequency bands and overlap in their coverage, so that a endeavor from one cell to the other can happen without interference to open connections. Every GSM user needs a *SIM card* (*subscriber identification module*) which is used for identification and to store data. GSM was mainly designed for voice telephony, but today more and more services on mobile phones require data transmissions.

HSCSD

GSM offers insufficient bandwidth for data services and the connection-oriented protocol is not ideal for using IP-based Internet services. Therefore, several new protocols were developed that should supplement GSM to allow data communication with speeds that are equivalent to their wired counterparts. A direct enhancement of GSM is HSCSD (high speed circuit switched data). It uses channel bundling with a time division multiplex method so that a data rate of up to 57.6 kBit/s is achievable when four time slots are combined in upload and download direction. Due to connection-orientation, users have to reserve (and pay!) all bundled channels for the entire connection time, although bandwidth is used only for short time frames (e.g., to transmit a new Web page) and afterwards users require a comparably long time to read until the next data is requested.

GPRS

GPRS (general packet radio service) is a packet-switched protocol with higher bandwidth than GSM and the advantage that only transferred data is billed, not connection time. This reproduces a development from time-oriented towards volume-oriented billing methods that became apparent in the history of wired Internet connections only a few years before. GPRS works like HSCSD with combining time slots, but unlike HSCSD it combines them on demand and offers more efficient coding algorithms. Both methods together provide data rates of up to 170 kBit/s in theory, but in practice only few telecommunication providers have implemented the most efficient coding algorithms so that effective GPRS data rates are only 53.6 kBit/s in most settings. Another shortcoming of GPRS is that typical round trip times (RTT) are over one second, even for small network packets in networks with low load which is high compared to RTTs between 10 and 100 ms for wired connections. RTT is measured from the moment the client sends a request until the response from the server reaches the client. High round trip times lead to long response times and thus a loss in perceived quality and user satisfaction.

EDGE

With EDGE (enhanced data rates for global evolution), telecommunication providers made a last attempt to enhance the wide-spread GSM net-

work before they had to build a completely new (and expensive) infrastructure for so-called 3rd-generation networks (3G networks). EDGE exists in two versions that enhance either HSCSD (ECSD) or GPRS (EGPRS). It replaces the inefficient GSM modulation method (Gaussian Minimum Shift Keying, GMSK) with the highly optimized 8-phase shift keying (8-PSK). This leads to a theoretical maximum data rate of 384 kBit/s. However, since 3G networks are built up in many countries (or are already realized, e.g., in Austria), EDGE is not expected to be widely adopted in developed countries. A major drawback of EDGE is that the highest data rates can only be achieved in close proximity to the transmitting station. This drawback is even more obvious than in GPRS, which show the same symptoms.

UMTS

UMTS (universal mobile telecommunications system) is the European part of a family of 3G network standards specified in the IMT-2000 effort (IMT = international mobile telecommunications). It uses the two frequency bands 1920 to 1980 MHz and 2110 to 2170 MHz. Originally, IMT-2000 aimed at a global unification of mobile telephone systems, but it turned out that this was not feasible due to several reasons, e.g., political differences between countries. The new infrastructure necessary to enable full-blown UMTS is called UTRA (universal terrestrial radio access). It envisions two versions called UTRA/FDD with data rates up to 384 kBit/s and UTRA/TDD with data rates up to 2 MBit/s for asymmetric connections and 384 kBit/s for symmetric ones. Besides the infrastructure that depends again on geographic circumstances (rural vs. urban areas), traveling speed of users is another factor that determines the maximum data rates that can be expected. A minimum data rate of 144 kBit/s is specified for traveling speeds up to 500 km/h whereas the maximum of 2 MBit/s can only be reached with speeds below 10 km/h. UMTS was not only designed to provide higher data rates. Another main goal is to unify formerly separate standards like DECT, GSM, WLAN and satellite communication standards like Inmarsat. UMTS further specifies four different quality of service (QoS) classes:

- *conversational* for bidirectional services like voice and video telephony,
- *streaming* for unidirectional services like video on demand or radio,
- *interactive* for typical Internet applications with high data integrity demands and low latency,
- *background* for services like SMS or fax that require high data integrity, but do not require low latency.

Summary

Figure 2-4 sums up the discussion of cable-bound and wireless network protocols and shows their speed and range.

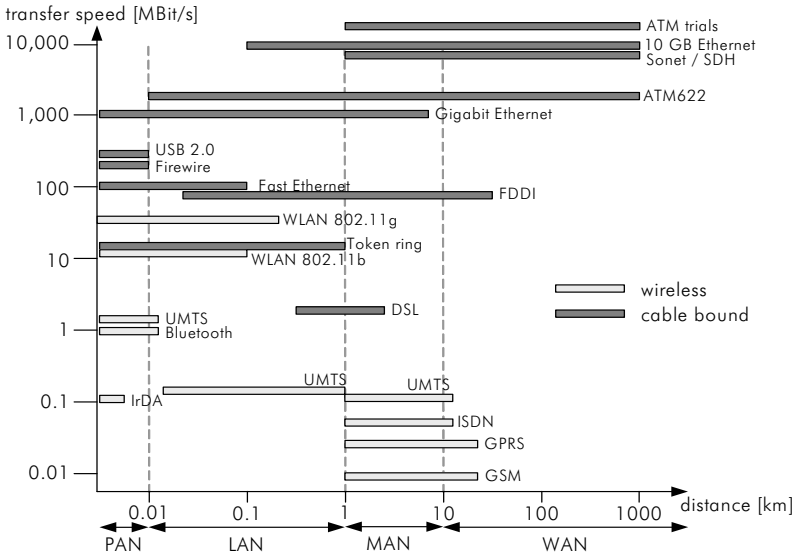

Figure 2-4. Protocols classified according to transfer rate and range

2.2.2 Network and Transport Layer

MAC address

The basic mechanisms for identification of network participants and establishing network connections can be found on layers three and four of the ISO/OSI reference model (section 2.1.3, 88ff). The identification on the lowest level uses a so-called MAC address (media access control). This is a 48 bit number usually displayed in hexadecimal system consisting of a 24 bit manufacturer identification number and a 24 bit serial number of the device, e.g., the network adapter. That guarantees a world wide unique number. An example of a MAC address is 08-00-20-AE-FD-7E.

IP address

However, a conceptual view of the network that enables logical partitioning into network segments is more suitable for network management. Therefore, a new identification method was established and the IP address (Internet Protocol) was introduced. IP address spaces can be assigned to organizations or organizational units independent of the hardware used. The address range 141.48.0.0 - 141.48.255.255 is assigned to the University of Halle-Wittenberg for example. When a server is replaced by a new one, the IP address stays the same although the MAC address of the server is different from that of its predecessor. IP addresses are 32 bit numbers consisting of four octets that build a kind of hierarchy (e.g., 141.48.204.242).

Network class

The four octets are divided into a network address and a computer address. Depending on the number of octets used for network address, the network is called class A (first octet only), class B (first and second octet)

or class C (first three octets) network. The number of octets used for computer address decides how many data stations can be part of the network (e.g., two octets equal two bytes, which allows for 2^{16}=65,536 data stations). To further clarify to which class a network belongs, the address space is also divided (Table 2-5).

Table 2-5. Network classes

network class	IP range	network mask	number of hosts
class A	1.0.0.0 - 127.255.255.255	255.0.0.0	16.7 million
class B	128.0.0.0 - 191.255.255.255	255.255.0.0	65,536
class C	192.0.0.0 - 223.255.255.255	255.255.255.0	256

Reserved address ranges

Within these address spaces, a part is reserved for private use respectively and may not be used in the public Internet (1.0.0.0 - 10.255.255.255, 172.16.0.0 - 172.31.255.255 and 192.168.0.0 - 192.168.255.255).

Mindful readers may have noticed that there are some addresses left in the IP address space that were not mentioned yet. These addresses are reserved for special purposes. The range from 224.0.0.0 to 239.255.255.255 is reserved for so-called multicasts and the range from 240.0.0.0 to 255.255.255.255 is reserved for research and further extensions of the Internet.

Multicast

Network participants usually communicate with each other one by one, so that there is one sender and one receiver. This way of communication is called *unicast*. However, in some cases it is useful to send network packets to more than one recipient at the same time which is called *multicasting*. The sender just sends a transmission to the multicast address and does not care how many participants get the message nor who gets it. A server on this address handles distribution of network packets to all data stations that have previously subscribed to the multicast. An application of multicasting is the transmission of a live video stream. As time is an important factor for live video transmissions, the most efficient way is to use multicasting, so that the sender sends the video data only once and does not have to wait for confirmation from every recipient.

Broadcast

Besides unicasts and multicasts, there is a third form of network communication called broadcast. A *broadcast* is similar to multicast in that multiple recipients get the transmission of one sender. The difference is that recipients do not have to subscribe to the transmission, but recipients are determined based on the network segment. The last address of every segment is reserved for broadcasts that reach every network participant within that segment. So a broadcast to the address 192.121.17.255 would reach each network device in the 192.121.17.xxx network and a broadcast to 255.255.255.255 would reach each network device in the whole world.

As the last case does not make sense because of immense network traffic usually broadcasts are not forwarded to participants outside of the network segment of the sender, so that both examples would lead to the same result. Limiting the range of broadcasts, which is a very popular mechanism, is an additional reason for administrators to further subdivide a network.

Subnet mask

This is accomplished by using subnet masks. They have the form of an IP address with 4 octets but use in the simplest form only zeros or 255s for every octet. 255.255.255.0 is an example of a subnet mask that specifies a subnet with three octets for the network address and one octet for the computer address, which would divide a class B network into 255 subnets. The subnet mask has to take the network class into account, e.g. 255.255.0.0 is no valid mask for a class C network as there are already three octets identifying the network. Subdivisions that do not divide networks at the border of an octet are also possible. The term *classless interdomain routing* is used to denote it. The mask 255.255.255.64 (64 decimal = 11000000 binary) uses two additional bits to address the network compared to a class C network, which leaves 6 bits for the computer address. This separates the class C network into four subnets (2^2=4) with a maximum of 63 hosts each (00111111 binary = 63 decimal).

Localhost

IP address 127.0.0.1 denotes the so-called localhost and always addresses the machine of the sender. Localhost is even more than a special address. It resides on a separate network interface that is different from the interface connecting to the outside and is called loopback interface. A computer could still reach itself on 127.0.0.1 if there is no network cable connected. In this way, proven mechanisms for communication can be used no matter if the receiving program resides on the same machine or a different one.

Hosts without IP address

The last IP address worth mentioning is 0.0.0.0 that can only be used as a sender address for network participants that do not have a valid IP address yet.

Internet protocol (IP)

The Internet protocol (IP) belongs to layer three of the ISO/OSI reference model (network layer) and its main purpose is identifying network participants and routing network packets from sender to receiver. It specifies addressing of network packets as described above and takes care of other issues like prioritization of packets, connection time-outs and preferred network routes. The version currently used in the Internet is version four (IPv4) which operates with 32 bit addresses (2^{32} = 4.3 billion). As the number of available addresses decreases dramatically with the number of Internet users continuously increasing, the need for a larger address space arose. This need will be satisfied with the implementation of IP version six (IPv6) that operates with 128 bit addresses which leads to an enormous amount of addresses (approximately 10^{24} addresses for every square meter on earth).

Routing

Routing means finding a way for packet from sender to receiver. This can involve several data stations that forward the network packet before it

reaches its destination. Data stations that function as mediators (section 2.1.3, 88ff) are called *router*. Usually there is at least one router for every network segment border a packet is passing. For wide area connections, packets pass more stations within short distances at the beginning and the end of the travel. In between, there are a few stations with great distances in between them. In our example about the message from New Zealand to Germany it is same: first the message is routed from secretary to post office (short distance), later from Singapore to Frankfurt (large distance) and at the end from the secretary to the business man. Making the way from one data station to the next one is called a hop. Network packets are allowed to make only a limited number of hops before they have to reach their destination. This number is called time-to-live (TTL), can be specified by an administrator and usually is around 30.

ARP

As on lower layers still the MAC address is used for identification, there is a need for translation from IP- to MAC addresses and vice versa. This is accomplished by a translation service called address resolution protocol (ARP) for IP to MAC translation, and by reverse ARP (RARP) for MAC to IP translation. Both ARP and RARP are network protocols.

ICMP, ping

Internet control and manipulation protocol (ICMP) is a protocol on the network layer and the basis for many small but useful tools that are usually part of the operating system. It specifies a number of messages and corresponding responses (e.g., echo). The most well-known application for ICMP is the `ping` command that exists on all operating systems providing network access. With the `ping` command, a user can check whether a host is reachable over the network by sending the ICMP `echo` message to it. If it is reachable, a response (`echo response`) is returned and time between sending and receiving the message is measured. If no response is received within a specified time frame (time-out), the partner is declared to be unreachable, even though it does not automatically mean the partner is really unreachable for any kind of network service (see firewalls in section 2.2.4, 106ff). A second useful application of ICMP is `tracert` (trace route) that helps to find out how many hops and over which routers a packet travels before it reaches its destination.

TCP and UDP

On layer four of the ISO/OSI-reference model, transmission control protocol (TCP) and user datagram protocol (UDP) fulfill the task of transportation. TCP is a connection-oriented protocol that establishes sessions for connecting to remote hosts. Data stored for the duration of the session (session data) is used for authorization and states.

Ports

Both, TCP and UDP specify 65535 ports. Every connection between two data stations has to specify one port each on sender and receiver side. On receiver side, usually a certain port specified for the service invoked is used. On sender side, any free port can be used. Port numbers from 1 to 1024 are reserved for well-known services on higher layers like HTTP on port 80 or FTP on ports 20 and 21 (section 2.3.2, 119ff). These port assignments are just a convention for user convenience and no obligation. It is

e.g., also possible to communicate over port 8000 or any other port using HTTP, but if the standard port 80 is used, the user does not have to specify the port explicitly. Port numbers above 1024 are free for use by any application. Despite this, there are some wide-spread applications that use certain default ports and it is good practice not to use one of those when developing an own networking application. Examples are the Oracle data base management system that uses port 1521, the Microsoft SQL Server that uses port 1433, HTTP proxies that use port 8080 and many Java application servers (section 2.4.2, 139ff) that use ports 8000 or 8001.

Differences between TCP and UDP

Difference between TCP and UDP is that TCP is a connection-oriented protocol whereas UDP is connectionless. IP networks are packet-switched networks, which means that two network messages that are sent by the same sender immediately one after another to the same destination do not necessarily have to take the same route to the destination, nor is there any guarantee that they arrive in the same order. For simple network communication, this is no problem as all messages take only one network message and there is no need for authentication or for keeping other session information. Those applications can use UDP as transport protocol that provides no error correction, confirmation for received messages or guaranteed order. TCP on the other hand, invests a substantial amount of protocol overhead (header data that has to be transmitted together with the payload) into establishing a virtual connection on top of IP in order to maintain data necessary to guarantee correct order of messages and exchange session ids.

TCP connections

Establishing connections takes place in three phases during which sequence numbers are exchanged to assure the connection. Every single message is confirmed by the receiver. If no confirmation is received within a specified time frame, the message is retransmitted. Another feature of TCP is that sending and receiving data at the same time is possible which is called *full duplex mode*. Additionally, it is possible to handle different higher-level protocols over a single TCP connection in one session which is called *multiplexing*. Finally, *flow control* enables TCP to automatically adjust its speed to different capacities at sender and receiver side.

Encapsulating and unwrapping data

Figure 2-5 shows how data is marked with headers on sender side as it passes down the layers (encapsulation) and how headers are interpreted and removed at receiver side (unwrapping). On the lowest layer there is also a checksum (CRC, cyclic redundancy check) that is used to check the integrity of the frame. The terms used to denote network packets also change from layer to layer. Whereas the term frame is used on lower layers (1+2), we speak of packets on the Internet layer (3) and of messages on the transport layer (4).

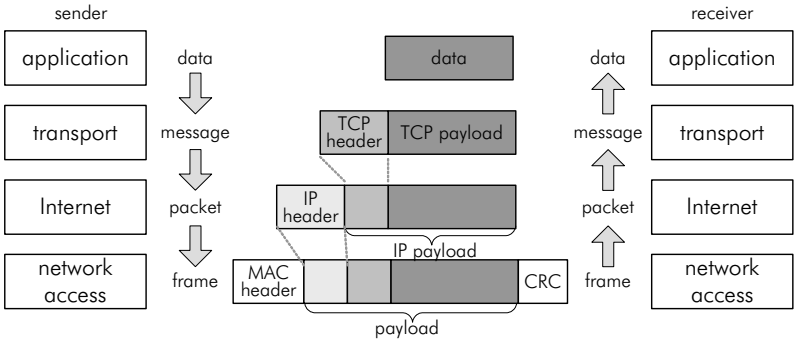

Figure 2-5. Encapsulating and unwrapping data in TCP/IP

2.2.3 Network Management

Management and supervision of networks must be as simple and efficient as possible. That applies firstly to the administration of logical structures on top of the physical network topology, namely IP addresses and DNS names. Secondly, it includes supervision of hardware components' states and easy remote control of network-related settings on clients.

DHCP

One mechanism to achieve this is the dynamic host configuration protocol (DHCP) that automatically assigns IP addresses to computers. It is quite simple and uses UDP as transport protocol. To exemplify general function of a network protocol, we explain the process of dynamically assigning IP addresses in detail.

DHCP example

A computer that goes online in a network and has no permanent IP address assigned needs to get a dynamic IP address before it can communicate normally. It thus generates a *DHCP discover* message that could be interpreted as the English sentence: "Can anybody give me an IP address?" It has no own IP address and it does not know which IP address the responsible DHCP server has, nor if there is one at all. Therefore, sending the message as a broadcast from 0.0.0.0 to 255.255.255.255 is the only option. If a DHCP server receives it and has free IP addresses in its pool, it takes one of those and sends it back as a *DHCP offer* message which could be interpreted as: "I can offer you this address! Do you want it?" This message is again sent as a broadcast, as the recipient has no address to send to yet. The computer that needs an IP address receives this message and sends a *DHCP request* message back stating: "Yes, I want this address! Please lease it to me!" The final message sent is from the DHCP server and is called *DHCP acknowledge* which means: "You can take the IP address. I have booked your MAC address on this IP and will give this address to no one else until the lease expires."

Advantages of DHCP

With four simple messages an IP address is assigned to a computer. Advantages of dynamic address assignment is that administrators do not have to enter addresses manually, IP addresses are better utilized if not all computers are online every day and a central register of all computers on the network together with their IP addresses is created automatically. DHCP servers can be configured so that certain addresses are only assigned to a certain computer (identified by its MAC address).

SNMP

A second protocol easing network management tasks is called simple network management protocol (SNMP, UDP ports 161 and 162). A central application acts as manager and collects status information via SNMP from network devices that act as SNMP agents. Managers can request information about status attributes with the `GET` and `GET-Next` commands that are answered by agents with `GET-Response` messages. Managers can also request a change of state for the network device with a `SET` command. Finally, agents can report critical states that have to get attention immediately with the `TRAP` message. A network administrator can use the manager to get a current picture of states from the whole network landscape.

DNS

The IP addresses are a first step from the physical identification via MAC addresses to a logical identification. Since people are usually better in remembering names than numbers, especially if the names contain information about the services a computer offers, a naming system has been established to identify computers by names instead of numbers. It is called domain name system (DNS) and is a network protocol. Its hierarchical structure is similar to the folder structure in the file system of a computer, but instead of forward '/' or backward slashes '\', dots '.' are used to separate the different levels. An example for a DNS name is:

```
www.google.com
```

FQDN

Such a name is called fully qualified DNS name (FQDN) as it includes not only the name of the computer itself (`www` in our example), but also the complete name of the hierarchy it is in. The so-called root domain, which is the top level element of all DNS names is '.', so that the complete FQDN would be "`www.google.com.`". However, the trailing dot is usually omitted.

Parts of a DNS name

A DNS name normally contains at least one top level domain, one sub domain (e.g., google, yahoo, freenet, …) and the host name (often `www` for computers offering World Wide Web services). Top level domains are either purpose-specific (e.g., `com` for commercial use, `tv` for television-related content, `edu` for educational institutions) or country-specific (e.g., `de` for Germany, `uk` for the United Kingdom, `nz` for New Zealand). In addition to DNS names, servers offering DNS name resolution services are also hierarchically structured. Name resolution means conversion from DNS names into IP addresses. If one DNS server can not resolve an IP address to a DNS name it asks the next DNS server in the hierarchy. For example, the DNS server for `wiwi.uni-halle.de` could ask the DNS server for `uni-halle.de` concerning the DNS name `www.urz.uni-`

`halle.de`. Besides data about the next higher DNS server, every DNS server holds data about the top-level DNS server in the so-called *root hints*. Table 2-6 shows an example for a DNS name with corresponding IP and MAC address.

Table 2-6. Example for DNS name, IP address and MAC address

address type	value
DNS name	www.wiwi.uni-halle.de
IP address	141.48.204.242
MAC address	00-90-27-7E-16-F8

Authority zone

Every DNS server has an area of responsibility called authority zone. It is the only server that has the authority to give qualified answers to questions concerning naming of computers in this zone. For example, if a client is asking for the IP address of `www.google.com` only the DNS server responsible for `google.com` can give a *qualified* answer to this question.

Caching

One server would soon be unable to answer all questions for a popular site. Thus, other servers are caching the name, so that the next question concerning the same DNS name can be answered based on the cached data. Such an answer is called *unqualified*, as the IP address could have changed meanwhile without the DNS server outside the authority zone being notified. This is no problem, since IP addresses of publicly accessible servers usually do not change often.

Dynamic DNS

For servers that do change their IP address regularly, it is possible to mark a name as cachable for a short time frame or not at all. This mechanism is called dynamic DNS.

Telnet

Telnet is a network protocol specified by the IEEE and provides a fairly general, bidirectional communications facility with seven-bit character set. It is typically used to provide command-line sessions to administrate hosts on the Internet. The idea behind was to create a virtual terminal or terminal emulation to control a host. The program implementing the client part of the protocol is also called `telnet` and can be used to invoke a Telnet session to a remote host. Telnet operates on the basis of TCP and uses port 23 by default. It provides a connection to interact with the remote operating system, e.g., using a UNIX shell. The program can also be used to connect to other remote services (e.g., SMTP on port 25, section 2.3.3, 123ff) and communicate with them using the native commands specified by the respective protocol.

2.2.4 Network Hardware

This section examines network hardware and builds on the explanation of ISO/OSI layers and their implementation in network protocols. Network

devices mainly differ with respect to their processing of header data at different layers.

Repeater

One of the simplest devices within the network is a repeater. It is limited to refreshing signals. Refreshing means, that a attenuated electric signal coming in is resent with full strength, so that longer distances can be bridged. Thus, repeaters operate at the physical layer (layer 1).

Hub

A second device type that has to be classified into the physical layer is a hub. It receives signals from one port and forwards it to all other ports. It can be seen as a repeater with more than two ports. A port in this context is a kind of socket where the network cable is plugged in.

Bridge

A network bridge can be seen as a smarter form of a repeater. It forwards network packets only if they are addressed to the respective network segment and therefore avoids collisions being spread into the whole network. It thus has to support physical and data link layer. A bridge is also able to connect networks with different topologies. In a way, a bridge creates a physical separation between two parts of a network that belong to the same or to different logical segment(s).

Switch

Another device class on layers one and two is called switch. A switch is like a hub because it connects multiple computers with each other. In contrast to hubs, it does not just forward a transmission to all other connected devices, but evaluates the target address (MAC address) in the network packet and forwards the transmission only to the port of the recipient or to the uplink (special port to "the rest of the network") if the recipient is not directly connected to the switch. Therefore, it provides kind of a direct communication between two computers which is significantly reducing collisions.

Access point

Access points manage connected wireless devices and provide an uplink to the wired part of the network. Sophisticated multiplexing methods (e.g., code division, CDM) are needed to directly communication with a single wireless device. Currently, these methods are rarely implemented so that access points can only provide direct communication in uplink direction and forward packets targeted at one of the wireless connected devices to all of them.

Router

Routers connect network segments. They have one connection to every segment and one IP address for each of them. They inspect packets coming in from one segment and process their headers up to network layer three to identify whether they are targeted to one of the other segments (or a segment not directly connected). They determine to which network interface the packets have to be sent, based on a routing table and IP addresses of network packets. Every computer has a simple form of a routing table built in that is being created based on IP address, network mask and *standard gateway*. Standard gateway in this context is the name of the router of the network segment. The routing table can be displayed with the Windows command `route print` (XP and 2000) or `route -p` on UNIX systems. Routers maintain much more complex routing tables. Since they would be hard to

maintain manually, most routers support one ore more of the automatic router configuration protocols, like RIP (router information protocol) or OSPF (open shortest path first).

Gateway

Devices on higher levels of the ISO/OSI stack are called gateways. We distinguish transport gateways and application gateways. Transport gateways connect two networks that use different connection-oriented transport protocols, e.g., TCP/IP and ATM. As the name already suggests, they examine headers of network messages up to the transport layer. Gateways on the application layer are called application gateways and must be able to process application-specific high-level data formats and translate data from one format into a different format. An email gateway for example could receive email messages with SMTP in the MIME format and forward them to a mobile phone in SMS format (section 2.3.3, 123ff).

Network interface card

Up to now we have only discussed standalone devices. But computers also need some kind of device or adapter (data transmission equipment) in order to connect to the network. Such an adapter is usually called network interface card (NIC), no matter if it really is an extension card (e.g., PCI or PCMCIA) or a chip on the mainboard of the computer as common today. It is an interface that maintains a permanent connection to the network.

Modem, DSL modem, ISDN adapter

Devices designed to create temporary connections to the network (dial-up networks) are called modem (modulator/demodulator) based on their initial purpose to translate between digital computer signals and the analogue signals used on phone lines. Nowadays, many telephone lines also operate on a digital basis (ISDN, integrated service digital network), but devices that create network connections over phone lines (e.g., DSL modems, digital subscriber line) are still called modems or sometimes adapter (e.g., ISDN adapter). Those dial-up connections provide much less bandwidth than permanent network connections starting from 14 to 56 kBit/s for analogue modems over 64 kBit/s for ISDN adapters up to 768 kBit/s to 3 MBit/s (seldom even more) for DSL and cable modems. Cable modems are used for Internet connections via TV-cable networks and are widely spread in the United States.

2.3 Infrastructural Services

On top of this basic network infrastructure, a set of higher-layer services establishes a more abstract and easy to use platform for applications. Depending on the structure of data, one of several storage services takes care of permanently and securely storing data on a medium. Specialized access protocols most of which are implemented in a Web server today can be used to access stored data from any place in the network. Messaging services are used to actively deliver data to one or more recipients. A set of

security services that covers the areas storage, access and messaging supplements the infrastructure at this level.

2.3.1 Storage

Data storage is organized in a layered system similar to the network layers in the ISO/OSI model. It starts with media that store single bits physically e.g., as optical or magnetic marks. A drive is needed to read and write from and to a medium. A controller provides access to the drive for the operating system in close conjunction with a device driver. The file system is a part of the operating system dealing with organization and access to data on a logical level using files and folders. On the application level finally, various application systems provide further abstractions that are suited either for structured or semi-structured data (section 3.1.1, 149ff). Table 3-1 gives an overview of the layered storage architecture.

Table 2-7. Overview of storage layers

domain	layer	examples	logical unit
information system	application	document and content management systems, hierarchical storage management systems, data base management systems	document, record
operating system	file system	FAT, NTFS, HFS, ext3, vxfs, reiserFS	file, folder
	driver	driver for controller, drive, volume	volume
hardware	controller	IDE, SATA, SCSI	device, drive
	drive	hard disk drive, tape drive, CD drive	block, sector
	medium	magnetic (tape), optical (CD), magneto-optical (MOD), electric (flash)	bit

Media. On the lowest level, all data is stored physically on a medium. Available media differ in terms of cost, capacity, speed, physical space needed and other characteristics. The following classification system helps to clarify the differences.

Online, offline, nearline

Availability of media can be separated into online, offline and nearline. Online storage means that the medium is directly available, e.g., main memory (RAM), hard disks. Offline storage denotes media stored separately from the drive in a cabinet that have to be manually put into a drive in order to access them, e.g., CDs, DVDs, tapes. Nearline storage is in between those two forms. Nearline media are actually offline, but can be automatically inserted into a drive by robots or other automated mechanisms, e.g., tape libraries, jukeboxes. A general heuristic is that the faster data has to be accessed, the smaller is the medium that can be used and the higher is the cost per capacity unit.

Volatility

Volatile media like RAM loose data after power has been switched off. Non-volatile media like hard disks or CDs keep data regardless of power supply. However, readability of data on non-volatile media is limited. Time varies from 10 to 50 years depending on medium and environmental influences which is still far less than durability of acid-free paper as a medium.

Cost

Cost of media reaches currently from roughly 200 € per GB for RAM to around 4 € per GB for hard disks and 0.2 € per GB for DVDs. Tapes are somewhere in between hard disks and DVDs depending on their type.

Table 2-8 shows examples for commonly used storage media ordered according to access time. Media on the top of the list are small, fast and costly (SRAM). Towards the bottom of the table, media get slower, larger and cheaper.

Table 2-8. Storage media ordered by access time

medium	typical capacity	typical access time	cost (in EUR)
SRAM	0.1 - 2 MB	5 ns	~ 100 per MB
DRAM	128 - 2000 MB	25 ns	~ 0.2 per MB
Flash memory	64 - 4000 MB	100 ns	~ 0.1 per MB
hard disk (SCSI)	18 - 300 GB	4 ms	~ 4 per GB
hard disk (IDE)	40 - 400 GB	10 ms	~ 0.5 per GB
magneto-optical disk	2.6 - 9.1 GB	50 ms	~ 6.5 per GB
CD	0,7 GB	100 ms	~ 0.7 per GB
DVD	4.7 or 8.5 GB	100 ms	~ 0.2 per GB
WORM	30 GB	100 ms	~ 18 per GB
tape	40 - 200 GB	60 sec	~ 0.4 per GB

Access method, access time

Access method and as a consequence access time are further characteristics of media. Methods are direct access, e.g., hard discs and optical media, and sequential access, e.g., tapes. Sequential access means that access to data saved on parts of the medium far away from each other takes a substantial amount of time, whereas data written directly one after another can be read much faster. For direct access media the difference is much lower. Access time is dependent on that and reaches from less than one millisecond for RAM to about 10 ms for hard disks, 40-100 ms for optical media up to several seconds or even a few minutes for tapes.

Physical characteristics

Media can further be classified according to the way they physically store data. Main types of storage systems are *magnetic* (hard disks, floppy disks and tapes), *optical* (optical disks, e.g., CD and DVD) and *electric* (SRAM, DRAM, flash). There are also some mixtures of the types like *magneto-optical* disks (MOD, e.g., Minidisk).

Magnetic media

Magnetic media uses very small magnetic particles that are evenly spread over a medium. The read/write head of the drive then uses induction to read and electromagnetic forces to change the magnetic orientation of the particle and therefore make it a binary zero or one.

Optical media

Optical media use refractional characteristics of a material (often silicon) to read. In the case of CDs and DVDs, a laser with a certain wavelength, e.g., 780nm (infrared) for CDs and 650nm (red) for DVDs, about 400nm (blue) for DVD successors, sends a light impulse to the disk. The light gets reflected and is detected by a sensor. The zeros and ones are realized with small holes (pits) in the surface (lands), so that light is reflected differently.

Electric media

Flash memory stores information in an array of transistors, called cells, each of which traditionally stores one bit of information and is usually made of silicon. There are two types of Flash memory: NAND memory is faster (especially for writing large blocks of data) and can be larger (currently up to 4 GB). NOR memory can be used directly for executing programs and thus is often used in PDAs to store the operating system.

Transfer rate

Transfer rate is depending on the density of data on the medium (the higher the density the more bits can be read at once) and the medium's speed of movement. The original CD speed (1x) is 210 rotations per minute (rpm) which equals a transfer rate of 150 kB/s. Current CD drives have 52x speed and therefore rotate with up to 11,000 rpm transferring 7.8 MB/s! DVDs operate with 1,400 rpm resulting in a transfer rate of 1.4 MB/s. Current 16x speed drives offer a transfer rate of 22 MB/s, but that speed is no longer measured against the angular velocity but against velocity at which a track bypasses the head in outer disk areas so that they reach only 8,800 rpm. As a comparison, current hard disks rotate between 5,400 and 15,000 times a minute.

Rewrite

The last characteristic is the ability to rewrite. Some media like hard disks can be rewritten nearly infinite times. Optical disks like CD-RW, DVD-RW or DVD-RAM are rewritable about a thousand times, whereas data can only be written once to other media like CD-R, DVD-R or DVD+R and WORM (write once read many).

Drive. Media need a drive so they can be read and written. In some cases, drive and medium are inseparable, e.g., hard disk, USB stick, but usually media can be removed from the drive, e.g., tape, CD, SD-card. The drive has to provide mechanisms to read and write data, to position the head to find data, and provides a first logical abstraction of the physical medium.

Tape library, jukebox

Mechanisms have been invented that automate insertion and removal procedures and increase available capacity dramatically, as several (usually 8-72) media can be used. Tape libraries deal with tapes and jukeboxes with optical discs. The principle is the same for both systems. Media are put into a special stockroom where a robot arm can pick the selected

medium up and insert it into a drive. Each medium is identified by a barcode in order to quickly find the right medium when searching for a file.

Controller and Driver. Controller and its driver implement the interface between operating system and drives. Program logic is partly stored in a controller's BIOS (basic input output system) and partly in the driver. Over the years, more and more program logic and functionality has been integrated into the controller's BIOS (see "Developments in abstraction from storage media" on page 116). Besides that, controllers differ mainly in the type of interface and the number of drives they can address.

IDE, SCSI, SATA

IDE (integrated drive electronics) is the current standard for consumer drives (hard disks as well as CD and DVD drives) and is able to simultaneously address two drives each on two channels. SCSI (small computer system interface) is the standard for drives used in enterprises and can address 7 to 15 drives. Serial ATA (advanced technology attachment) is the most recent standard and tries to fill the gap between IDE (also called ATA or parallel ATA) and SCSI. A number of enhancements in the different versions of the interface protocols and partly also the physical interfaces have increased the transfer rate of both SCSI (Ultra SCSI, Ultra Wide SCSI, Ultra160 and Ultra320) and IDE (ATA33, ATA66).

Partition, volume

Space on a drive can be separated into one or more partitions that form logical units. More important for enterprise infrastructures are volumes as a logical abstraction of disk drives. In the simplest case, a volume consists of one partition, but it can also span multiple equally big partitions on several disks (see also the section on RAID below). The abstraction layer that handles volumes is called *logical volume manager*.

File system. A file system builds on volumes and structures them logically. Folders are used to hierarchically structure files which are the smallest logical unit containing data. A file allocation table (FAT) stores information about positions files begin at and about how long they are.

Folder hierarchy

UNIX file systems use a root folder ("/") as top-level element whereas Windows systems work with drives representing a volume for the file system and each drive has a drive letter and its own root folder. Since Windows 2000, it is also possible to mount volumes into folders which is common practice in UNIX file systems. Besides that, file systems differ mainly in their capabilities to handle long filenames, to address large volumes, to handle file security (encryption and access control) and to deal with fragmentation and corrupt files (e.g., due to power failure).

FAT, NTFS, UFS, ext2

On Windows systems, FAT was the dominant file system for years, first in version FAT16 that could address volumes with up to 4 GB and later with FAT32. In Windows NT and its successors, NTFS (new technology file system) is used mainly due to its advantages in file security, i.e. encryption and access control using access control lists (ACLs). Nonetheless, FAT is still relevant as it is used on floppy discs and flash memory.

On UNIX systems, the traditional UNIX file system (UFS) has been replaced by several (often OS-specific) newer systems (e.g., Linux ext2, Irix EFS, HPUX HFS).

Journaling file systems

In the last years, most of them have been replaced by so-called journaling file systems. Journaling introduces the concept of transactions to file systems by recording each write transaction and committing it after success. Thus, corrupt files can easily be identified instead of having to scan the whole disk for any inconsistencies after a crash. In Linux, ext3 adds journaling to ext2. Reiser FS and JFS (Journaling File System) are native journaling file systems and are also used in other UNIX systems.

Database file systems

With an increasing number of files being stored, it becomes increasingly difficult to find them using the simple folder hierarchy provided by current file systems. A recent approach to overcome this limitation uses meta-data (section 3.2.1, 173ff) to dynamically generate views on files or find files according to attributes. It uses the query capabilities of relational data base management systems to efficiently access files by querying meta-data. With this approach, queries for e.g., files used last week, that are related to a project or are written by a certain author can easily be answered. The public discussion about data base file systems was initiated by Microsoft's presentation of WinFS as integral part of the forthcoming Windows version (codename Longhorn). With WinFS and Longhorn being more and more delayed, probably until the end of 2006, other organizations like Apple Computer, Inc. with its Spotlight technology that will be included in MacOS 10.4 (codename tiger) and the open source community with DBFS (data base file system) that will probably be included in KDE version 4 gain more interest.

Application. On the application layer, data storage is based on files and further abstractions are introduced. *Documents* are logical units for semi-structured data and often are stored in one file, but can also span multiple files, e.g., master and sub-documents in MS Word or files composing an OpenOffice document (for a detailed definition of document see section 4.2, 246ff). *Records* are logical units for structured data and several records are stored within one file.

Record, field, attribute

A record is a set of data treated as a logical unit, also called entity, and consisting of attribute value pairs, also called fields. An attribute describes what kind of data is being stored. It is therefore data about data or meta-data. It has at least a data type, a name and ideally also has a description that helps users to interpret values. Thus, meta-data captures part of the semantic. Data is divided into structured and unstructured data according to the amount of meta-data that is available to describe structure and semantics of the data (section 3.2.1, 173ff).

DBMS

Usually data base management systems (DBMS) are used to store structured data. A data base system consists of a collection of structured data (data base) and software, the DBMS, to describe, store, create, retrieve and

manipulate data. They further abstract from file-based data storage mechanisms offered by operating systems. DBMS use relational calculus to store data in relational tables consisting of typed fields (columns in a table) and data records (rows) where instance values for each field are stored. Tables usually have a primary key that uniquely identifies a single data record in a table. Such a primary key can be referenced in a second table to establish a relation between the two tables and is called foreign key there. Structured query language (SQL) is used for inserting, retrieving and manipulating data. The following SQL statement creates a new record in the table `customer` that consists of the primary key `CoID` and the fields `CoName`, `CoStreet` and `CoCity`. Values of the fields in this new record identified by the `CoID` 17 are given as Smith, Nelson Avenue and London.

SQL insert

```
INSERT INTO customer (CoID, CoName, CoStreet, CoCity)
VALUES (17, 'Smith', 'Nelson Avenue', 'London');
```

The newly created record could be retrieved from the data base using the following statement.

SQL select

```
SELECT * from customer WHERE CoID='17';
```

DBMS additionally provide support for e.g., transactions including roll-back of incomplete transactions, fast access to data via indexes and caching, backup and restoring data, verifying integrity, monitoring of performance and user management to control access. A DBMS can handle multiple data bases. A data base is usually stored in one file for data and a separate file for indexes.

Directory service

A directory service is designed to store structured data and optimized to provide fast read access and simple query capabilities. In contrast to regular data bases that store data mostly in relational tables, directories provide hierarchical storage (Figure 2-6). In addition, a directory is designed for fast retrieval instead of data manipulation. Another difference is the lack of sophisticated transaction or roll-back mechanisms. Like data bases, directory services principally can hold any kind of structured information. However, they are mainly designed for relatively small portions of data, e.g., for user account management.

Scheme, distinguished name

Each directory has a certain structure defined by a scheme. The scheme can be designed for any purpose like data base schemes. Schemes define object classes. Each directory entry is member of one object class. The main object class within a directory is `entry`, which is mostly composed around real-world concepts such as people or organizations. A unique distinguished name (DN) identifies each entry, similar to fully qualified domain names in DNS. An entry can have one or more attributes, defined by the scheme. Unlike data bases, directories typically offer a set of predefined object classes and attributes.

```
c=de;o=University Halle;ou=Wiwi;ou=MIS;cn=Rene Peinl;
```

Example of a distinguished name

The example uses the object classes `country`, `organization`, `organizational unit`, and `person`, which are all subclasses of `entry`. Further relevant object classes are `organizational role`, `organizational person` (a subclass of `person`) and `alias`.

Important attributes for these classes are country name (`c`), street address (`sa`), organization name (`o`), organizational unit name (`ou`), common name (`cn`), surname (`sb`), email address (`mail`), user ID (`uid`), and user password (`userPassword`).

Application areas

Directory services are commonly used for organization-wide address books and application-spanning login information (section 3.3.2). Most current email clients such as MS Outlook, Apple Mail, Eudora Mail or Mozilla Thunderbird support searching LDAP (lightweight directory access protocol) directories for addresses and many enterprise application support central management of user data in LDAP directories. Examples for directory service implementations are Microsoft Active Directory, Novell eDirectory and the open source system OpenLDAP. Figure 2-6 shows an example for two person objects in a hierarchical organizational structure.

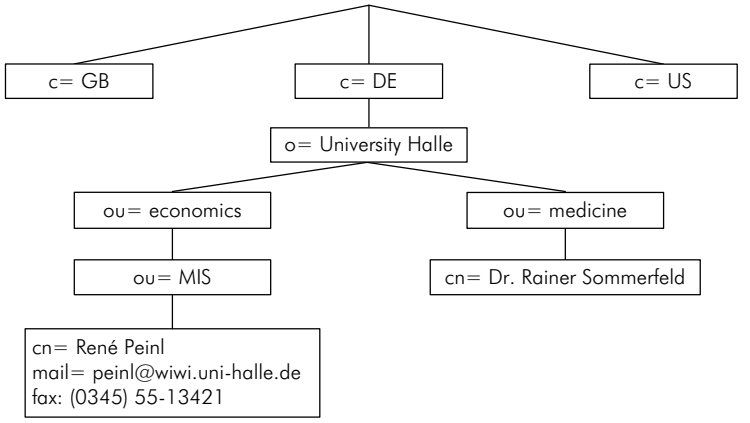

Figure 2-6. Hierarchical structure of an LDAP directory service

LDAP server can be operated redundantly in a master-slave configuration. Entries can only be edited on the master server. A synchronization mechanism called replication updates slave servers when data has changed on the master.

LDAP

LDAP (lightweight directory access protocol) is a client-server protocol for accessing and managing directory data. LDAP was developed at the University of Michigan in 1995 and is part of the X.500 specification. It was originally developed as a lightweight front end for the X.500 directory

system, as the original X.500 standard was too resource-intensive for Internet and desktop applications. Whereas the X.500 Directory Access Protocol (DAP) was implemented on a separate network protocol stack, LDAP is based on TCP/IP and uses only string formats for data transport.

File server, DMS, CMS

There are three alternatives for storing semi-structured data. File servers store files of all types in the file system of the operating system and exposes them to network users with a suitable protocol (section 2.3.2, 119ff). Document management systems (DMS) are designed to store documents together with meta-data. They excel file servers by e.g., providing versioning, advanced locking mechanisms and sophisticated search capabilities (section 4.2.1, 247ff). Content management systems (CMS) have been designed to manage Web pages or content for Web publishing, but are also used to manage digital assets of any kind (text, audio, image, video). DMS and CMS features overlap, though, and are more and more integrated in enterprise-wide document and content management systems (section 4.2.2, 258ff).

Storage systems. As there are many different media suitable for storage there is a need for unified access. Ideally, users do not need to deal with the question whether data is stored on hard disks, DVDs or tapes. Storage systems manage multiple (different) media, unify access to them, and optimize access times by caching data.

HSM

A system closely related to that is called hierarchical storage management (HSM) system. Not all data is accessed equally often. Therefore, it is a waste of money to store all data on fast media like hard disks. Data that are rarely used should be stored on cheaper, but slower media. HSM systems integrate access to several different types of storage sub-systems and provide automated, rule-based migration of data from expensive hard disks to cheaper tapes or optical disks. HSM systems are discussed in conjunction with *information life-cycle management* (ILM) which implements more advanced rules related to the data value to decide which data to store on cheaper media.

Developments in abstraction from storage media. Development in the storage area (as in many other areas) is characterized by increasing abstraction. Programmers do not have to care about how to position the head of the disk drive, but can work with records and documents instead. The first abstraction is that logical blocks of data on the disk can be addressed (LBA, logical block addressing) instead of having to deal with head movement. Blocks are numbered from 0 to 65535 and can be read and written using their numbers.

RAID

In the next step, volume abstraction went down from driver to controller. Controllers no longer provided access to disks, but rather to volumes with a given capacity and certain security and performance characteristics. The mechanism to aggregate multiple disk partitions to volumes is called

RAID (*redundant array of independent disks*, sometimes the "I" is interpreted as inexpensive). RAID is designed to integrate multiple hard disks into one array accessible as a volume. Data stored in a RAID is spread over all disks which is managed by the controller. RAID offers several different operation levels that allow tailoring transfer rate and level of security.

RAID levels

The most important *RAID levels* in practice are level 0 (striping), level 1 (mirroring) and level 5 (striping with parity). *Striping* spreads data equally on all hard disks, without any redundancy. The result is fast transfer rates and no fault tolerance. If one hard disk fails, then all data is lost. *Mirroring* writes all data redundantly to two disks instead of one. If one disk fails the other one still holds the data. The failed disk can be replaced and data is rebuilt on that disk, so that no data loss occurs. *Striping with parity* can be used with three disks and more. Data is written on two disks and parity information on the third disk. If one disk fails, data can be rebuilt using parity data. There is also a level 10 or 1+0 which is a mixture between striping and mirroring.

A proven strategy in practice is to use two disks with RAID level 1 for the operating system and the remaining disks with RAID level 5 for data. RAID logic can be implemented on the level of the controller, the driver, or even at file system level, but usually it resides in the controller's BIOS today. Table 2-9 summarizes the RAID levels, indicates their levels of security and speed (data transfer rate) and shows what capacity is actually usable when 6 disks with a capacity of 200 GB each are used.

Table 2-9. Example for different RAID configurations with six 200 GB disks

RAID level (name)	usable capacity	security	speed	description
0 (striping)	1200 GB	--	++	no redundancy
1 (mirroring)	600 GB	++	o	full redundancy
5 (striping with parity)	1000 GB	+	+	1 disk for parity information
10 (striping and mirroring)	600 GB	++	+	full redundancy and striping

Hot swap hard disks

Usually, hard disks used for enterprise applications are *hot swappable*, which means that they can be exchanged while the system is running. If a hard disk fails, then it can easily be removed from the system and a new blank hard disk of the same capacity can be inserted instead. Data on the old disk is rebuilt within a few hours so that the system runs without interruptions for RAID levels 1, 5 or 10.

Direct attached storage, RAID array

A next step in the development was that the whole storage sub-system has been outsourced from the computer (usually only for servers). The advantage of these RAID arrays that are outside of the computer chassis is that more space for additional disks is available. RAID arrays grew more

and more independent, so that a second controller was needed inside the computer chassis to connect to the controller inside the RAID array. Such a storage subsystem is called *direct attached storage* unit (DAS). Existing interfaces (e.g., SCSI) were designed to match transfer speeds of single disks or a hand full at most. New interfaces had to be introduced (e.g., ultra wide SCSI) that are able to deal with the accumulated transfer rate of a dozen and more disks. Fibre channel is the most common interface used today.

SAN

The next step was to further separate computer and storage sub system by sharing storage space between multiple servers. To accomplish this, a device similar to a network switch has to be introduced in order to route data from storage unit to servers and vice versa. Such *storage switches* exist for fibre channel. Centralized storage units together with connections to servers are called *storage area network* (SAN). Initially, only entire disks could be joined to RAID volumes and assigned to servers. This lead to inefficient space allocations, especially since standardization of parts forbids to use different disk sizes e.g., 18 GB disks for the operating system and 72 GB disks for data. The goal is to provide disk space as a central service, where capacity, speed and redundancy are the features and parts of the overall disk space with a certain feature set can be dynamically assigned to servers. That includes increasing or decreasing assigned disk space (as long as there is free space on the partition). Several servers can connect to one SAN and every server gets its own space. Alternatively, multiple servers can share disk space (for clusters). Advantages are better utilization of disk space and increased performance, because for RAID level 5 e.g., 20 hard disks can be used for striping instead of 4 or 5 in a single server, which results in theoretically 4 to 5 times more speed and nearly as much in practice. The disadvantage is higher costs, since more intelligence in controllers, SAN switches as well as in the driver and BIOS on servers are required.

IP-SAN

Latest development in the storage area is reuse of wide-spread standard technologies in order to lower costs. Fibre channel connections between servers and SANs are replaced by IP connections over Gigabit Ethernet. Since these technologies are much cheaper due to their universal usage in data and storage networks and resulting volume effects. This type of SAN is referred to as IP-SAN and requires its own IP address. Existing storage protocols have been extended, so that they can run over IP connections. Examples are iSCSI that extends SCSI (small computer systems interface) as well as iFCP (internet fibre channel protocol) and FCIP (fibre channel over IP) that extend the fibre channel protocol.

Network attached storage

Network attached storage (NAS) is directly accessible using standard network protocols such as NFS and SMB (section 2.3.2) and can be seen as a further enhancement of IP-SANs. They can be used for data storage

only, since an operating system is needed on servers to connect to a network share.

Figure 2-7 gives an overview of the storage sub-systems discussed here and shows how they differ regarding protocols, interfaces, hardware components, and software layers. DAS, SAN and IP-SAN operate with block I/O which enables raw partitions for swapping or data base, so that operating systems can be installed on volumes on these devices. NAS operate with file I/O so that they behave like file servers. In fact, many NAS appliances are file servers based on Linux.

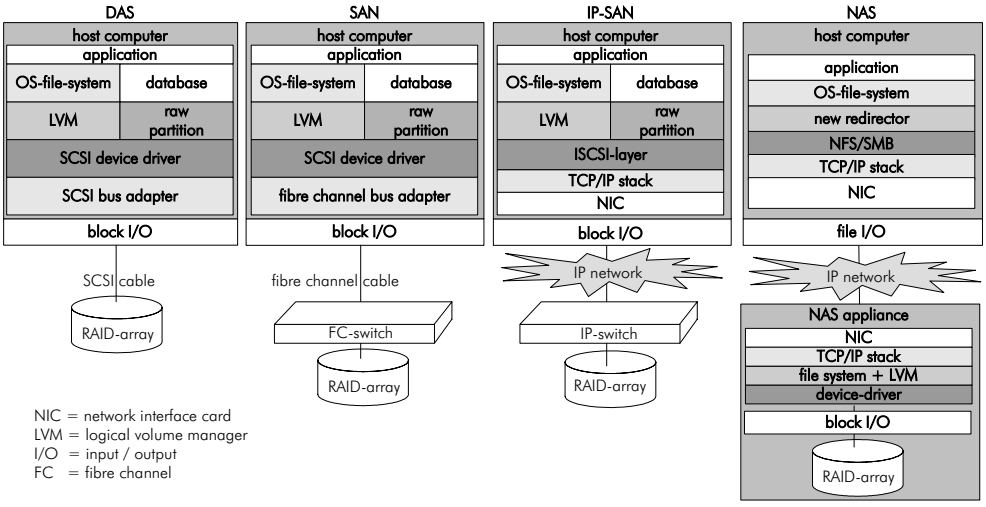

Figure 2-7. Overview of storage subsystems

2.3.2 Access

Standard Internet protocols are commonly used for access, so that the same mechanisms can be used for Intranet and Internet publishing of documents. Most of the implemented protocols are offered by a system class called Web server. Figure 2-8 recapitulates the protocols discussed in previous sections and gives an overview of some protocols that will be described in the next sections. Concrete protocols often implement multiple layers defined by the ISO/OSI reference model. In case of TCP/IP, there are only four layers called Internet layers.

<table>
<tr><td>ISO/OSI layer</td><td>Internet layer</td><td colspan="8">concrete implementations</td></tr>
<tr><td></td><td></td><td>WWW</td><td>email</td><td>file transfer</td><td>directory service</td><td>host sessions</td><td>name resolution</td><td>network monitoring</td><td>IP address assignment</td></tr>
<tr><td>7
6
5</td><td>Application</td><td>HTTP</td><td>SMTP, POP3, IMAP</td><td>FTP</td><td>LDAP</td><td>Telnet</td><td>DNS</td><td>SNMP</td><td>DHCP</td></tr>
<tr><td></td><td></td><td>80</td><td>25, 110, 143</td><td>20/21</td><td>389</td><td>23</td><td>53</td><td>161/162</td><td>67/68</td></tr>
<tr><td>4</td><td>Transport</td><td colspan="6">TCP</td><td colspan="3">UDP</td></tr>
<tr><td>3</td><td>Internet</td><td colspan="5">ARP
IPv4, IPv6</td><td colspan="4">ICMP</td></tr>
<tr><td>2
1</td><td>Network</td><td colspan="3">Ethernet, ...</td><td colspan="3">FDDI, ...</td><td colspan="2">WLAN, ...</td><td colspan="2">IrDA, ...</td></tr>
<tr><td colspan="2">Medium</td><td colspan="3">copper</td><td colspan="3">fiber</td><td colspan="2">radio</td><td colspan="2">light</td></tr>
</table>

Figure 2-8. Internet protocols and ISO/OSI layers

HTTP

One of the most important protocols is the *hypertext transfer protocol* (HTTP) that forms the basis of the *world wide Web* (WWW). Version 1.0 was specified in 1991 by the CERN institute in Geneva (current version: 1.1). Purpose of HTTP is delivering content in form of hyperlinked text which means HTML content. HTTP typically runs on port 80 of TCP connections. The basic mechanism is quite easy: a `request` for a resource is sent and the resource is being transferred back to the requestor (`response`).

URL

To identify the resource a mechanism called *uniform resource identifier* (URI) is used, or more precisely a special form of URI called *uniform resource locator* (URL, section 3.1.2, 151ff). The structure of a URL is always the same, although there are some mandatory and some optional parts. An example for a concrete URL is `http://www.heise.de/news/index.html`

URLs in Web browsers

First part of a URL is the protocol. Examples are HTTP, FTP and file (file means local files on the requesting machine). If no protocol is specified, Web browsers use HTTP by default. A user name and password can be specified for connections that require authentication, as FTP connections usually do. An @-symbol separates the user name and password part from the DNS name of the server (tld means top level domain). Any DNS name can be used here, as discussed in section 2.2.3, 104ff. Optionally, a port can be specified that is used on the destination machine to offer the service. If the port is omitted, the default port is being assumed, e.g., port 80 for HTTP or port 21 for FTP. Then a path to a file on the server follows which can contain folders and subfolders and usually ends with `html`. For user convenience, most Web servers support a mechanism called default file, which means that a certain Web page (often index.html) is being

transmitted if the path includes only a folder, but no file. The complete URL structure as used in Web browers is

```
protocol://username:password@server.domain.tld/path
```

URL structure

HTTP supports two methods for requests, `GET` and `POST`. Originally, `POST` was intended only for the rare cases were users fill in forms and send their content to the server, whereas `GET` was intended to be the default operation. Meanwhile, many Web sites are interfaces to complex applications and nearly every time these dynamic Web sites are refreshed, data is sent to the server (sometimes more, sometimes only a session id). A second form of sending data to the server used with the `GET` operation is called URL encoded sending of data. To accomplish that, the URL as discussed above is extended. A question mark "?" separates the path from parameters that are sent to the server. Parameters are listed in the form parameter=value and multiple parameters are separated with ampersands. Two parameters are encoded in the following example URL, `q` for the query term and `sourceid` for used browser.

GET, POST

```
http://www.google.com/search?q=knowkom&sourceid=mozilla
```

Example of URL encoding

A requests consist of a *header* specifying the request method, parameters and *payload*. Replies also have a header that contains a *status code* and contents of the requested file as payload. Status codes tell the requestor whether everything went fine (status `200 - OK`) or there was an error (e.g., `404 - file not found`). A new TCP connection is established for each request. There is also an encrypted version of HTTP called HTTPS (secure HTTP, section 2.3.4, 124ff).

A second protocol important for the Internet is the *file transfer protocol* (FTP). It has been designed to transfer large files of any type, whereas HTTP was designed for (small) HTML files and a few images. Although it is good practice to put larger files that can not be directly displayed within a Web browser on an FTP server, today a lot of *downloads* (e.g., installers, music files, office documents) are offered via HTTP. FTP uses port 21 to send control commands and list directory contents and port 20 (called `ftp-data`) to transfer files. Basic commands for FTP are `list` (to list the directory contents), `put` (to upload a file from the client to the server) and `get` (to download a file from the server to the client). The `help` command gives an overview of all available commands.

FTP

Another major Internet protocol is the *network news transfer protocol* (NNTP). It is used for newsgroups where users can post messages to certain topics (section 4.2). NNTP uses TCP-connections on port 119. Similar to FTP, NNTP is partly replaced by HTTP with the advent of HTML-based newsgroups and discussion forums that become increasingly popular. A kind of unification of protocols seems to take place that could be driven by

NNTP

the popularity of Web browsers as central access application. FTP and NNTP both require specialized clients to unfold their full potential and thus their popularity decreases.

WAP

The result of the attempt to bring the Internet to mobile phones is a protocol called *wireless application protocol* (WAP). WAP is a family of protocols that reach from layer three to layer seven of the ISO/OSI reference model. The *wireless control message protocol* (WCMP) has been proposed as a replacement of TCP/IP on the transport layer. The next layer in the WAP protocol stack is called *security layer* and replaces encryption standards like SSL (section 2.3.4, 124ff) with the *wireless transport layer security* (WTLS). The *wireless transaction protocol* (WTP) and the *wireless session protocol* (WSP) correspond to HTTP on the transaction and session layers. The top of the WAP stack is marked by the *wireless application environment* which is the connection point for applications using WML (*wireless markup language*) and WMLScript. Some applications only use some layers of the WAP stack.

NFS, SMB

Another mechanism that provides access to data over the network is network file access. It can be seen as network extension of file systems and abstracts from the underlying file system. The two most common network file systems are the NFS (*network file system*) used on Unix operating systems and SMB (*server message blocks*) used in Windows environments. Since Windows 2000, the SMB protocol has been enhanced to run over TCP/IP connections instead of requiring the proprietary NetBIOS protocol. This enhancement is called CIFS (*common internet file system*), although it is not commonly used in the Internet. A Unix application called Samba enables Unix servers to act as Windows file servers and most Unix derivatives also support access to Windows file servers. The purpose of both file system extensions is to provide access to files stored on file servers as if they were stored on local hard disks. In order to achieve this, users have to connect to network drives (Windows) or map folders on file servers to an entry point in the local file system (Unix, Windows XP). Other examples for network file protocols are Apple's AFP (Apple file protocol) and Novell's NCP (Netware core protocol).

WebDAV

WebDAV (*Web distributed authoring and versioning*) is an extension to HTTP to enable read and write access to documents on remote servers, regardless of the operating system used and no matter whether they are simple file servers or more complex systems, e.g., DMS. The standard describes a number of useful functions and mechanisms and is still being extended, e.g., recently with the WebDAV access control protocol, but most implementations support only a small subset of the standard. Especially versioning functions are seldom implemented. Main functions of WebDAV are locking of files that are currently in use by a user, handling of meta-data (properties) and copy or move operations directly on the server. Recent enhancements also allow for versioning and access control.

2.3.3 Messaging

Data often should not only be passively provided for access to somebody, but actively sent to a recipient. Communication can happen either synchronously (same time) or asynchronously (different time). Examples for synchronous messaging solutions are *instant messaging* (e.g., ICQ, MSN Messenger, Yahoo Messenger) and *internet relay chat* (IRC). Besides these popular (and somewhat standardized) possibilities there are also numerous proprietary applications that provide similar communication functions (e.g. Groove Virtual Office or IBM Notes Workbench, section 4.3, 267ff). In the following section we will review some of the more established asynchronous communication forms that are more closely related to infrastructure.

Email, SMTP

The most common form of asynchronous electronic communication is electronic mail, or short email. There are different network protocols on the application layer involved in sending and receiving emails. For sending them, the *simple mail transfer protocol* (SMTP) is used, which operates on TCP port 25 to communicate with the server. SMTP is really a simple protocol, so that users that want to try it on their own can directly communicate with the server in a telnet session if they use the following commands. `HELO`, to introduce yourself, `MAIL FROM:` to specify the sender address, `RCPT TO:` to tell the server the recipient's mail address, `DATA:` to start the message body and `SUBJECT:` to specify the subject. A single dot in a new line ends the message body and sends the mail. SMTP is also used to forward the mail from one mail server to the next until it reaches the final server that manages the post box of the receiver.

POP3, IMAP

Receivers can view contents of their post box and download mails with a protocol called *post office protocol version 3* (POP3, port 110). An alternative for POP3 introduced by Microsoft is the *interactive mail access protocol* (IMAP, port 143) which is a bit more flexible than POP3 (e.g., message synchronization between server and multiple clients). Common email servers are postfix and sendmail for Unix-based operating systems and Microsoft Exchange for Windows-based systems.

MIME

Besides the transport and retrieval of email, there also has to be agreement on content formats that should be used. For the text body of emails users can choose between plain ASCII text, rich text and HTML format. The MIME format (*multipurpose internet mail extensions*) is the most commonly used standard for entire messages. It specifies that a message consists of a header, the message body and zero to many attachments. A MIME header defines the encoding of a message (which character set is being used, e.g., ISO-8859-1 also known as Latin 1 or Western Europe encoding) and a content type (or MIME type) for each part of the message (e.g., `text/html`). Non-text parts are transformed from their binary format into a text format with an encoding known as *base64*. The MIME type is especially important as it is also used with HTTP.

SMS, EMS, MMS

A simplified and limited version of email is the mobile service SMS (*short message system*). Initially considered to be of minor importance by telecommunication providers, it is now one of the largest sources of income for them, especially in Germany with 25 billion SMS sent in 2003. An SMS is up to 160 characters long and is transmitted over free capacities of the signaling channel of mobile communication infrastructures, so that it does not interfere with voice and data connections. The successor of SMS is called EMS (*enhanced message service*) and enables users to transfer larger texts (up to 760 characters), small pictures and ring tones. Since MMS (*multimedia messaging system*) became available shortly afterwards, EMS has never been widely adopted. MMS supports multimedia objects in standard Internet formats instead of using proprietary formats like EMS, e.g., images in GIF and JPEG format audio files in MIDI and MP3 format and even video clips in MPEG format (section 5.2.4).

MOM

A form of asynchronous transport for messages that is only used for communication between servers is *message-oriented middleware* (MOM, also called *message queues*). In contrast to email, where delivery is not guaranteed, message queues provide a reliable way of sending messages where senders can be sure that messages are delivered. Message queues use proprietary protocols on different TCP ports (e.g., 1881 for IBM or 1801 for Microsoft). Some products additionally support SSL encrypted delivery. Common products in that system class are BEA message queue, IBM MQseries, Microsoft MSMQ and SonicMQ.

2.3.4 Security

In order to establish a secure area within company networks that are connected with each other, networks have to be split logically into different security zones.

Internet, ARPANET

Security zones Internet, Intranet, Extranet. Generally speaking, an internetwork or internet is formed when different networks are interconnected. Different means that they are maintained by multiple organizations and that various network technologies (e.g., different protocols or communication media) are connected with each other. The global network commonly known as the Internet is such an internetwork. Its origin dates back to the ARPANET which was developed in the late 1960s and put into operation in 1969 connecting a number of research institutions (e.g., the University of California with the research department of the US Department of Defence) in order to exchange data on joint research projects. The key underlying idea of the Internet is open-architecture networking where choice of individual network technologies should not be dictated by a particular network architecture, but rather could be selected freely by a provider. The second core idea is that the Internet should keep working if some hosts within the network break down, so that in case of an attack the

infrastructure could be still used to communicate, although several nodes would be disabled. Connection with other networks is possible through a meta-level *Internetworking Architecture*.

Internet growth

The Internet has grown tremendously from 213 hosts in 1981 over 14 million hosts in 1996 to 233 million hosts in 2004 (according to Internet Systems Consortium, www.isc.org). Access to the Internet is unequally distributed with respect to continents and social levels. For example, in North America 68.8% of the population had access to the Internet, whereas in Africa only 1.4% of the population had access (source: Internet world statistics, www.internetworldstats.com).

Institutions for network management

The Internet is organized decentrally with as little central control as possible. Nevertheless, some central institutions are necessary for network management. The most important ones are:

- *Internet Society* (ISOC), a non-commercial umbrella organization that hosts over 150 Internet-related organizations, e.g., the IETF,
- *Internet Engineering Taskforce* (IETF) that develops new and enhances existing Internet protocols and standards,
- *Internet Architecture Board* that steers development of protocols,
- *Internet Assigned Numbers Authority* (IANA) that manages protocol parameters (e.g., TCP ports) and their assignments to services,
- *InterNIC* (Network Information Center) that is responsible for globally unique addresses,
- the independent *Internet Corporation for Assigned Names and Numbers* (ICANN) that is responsible for the allocation of domain names and Internet addresses and will take over tasks of IANA and InterNIC which are controlled by the federal government of the United States.

Internet service providers (ISPs)

Internet service providers (ISPs) operate sub-networks with high bandwidths and provide access to the Internet. They are commercial (e.g., EUnet, NTG/XLINK) or non-profit organizations (e.g., DFN in Germany). National telephone companies like the German T-Com or international companies like America Online (AOL) or Microsoft Network (MSN) are commonly known ISPs offering Internet access for private users together with additional services like email services and Web portals.

Internet standardization

All participants of the Internet use the TCP/IP protocol family, a set of specifications that define how machines in the Internet communicate platform-independently. All Internet standards are documented in *requests for comments* (RFCs). TCP and IP specify two protocol layers of the ISO/OSI protocol stack (section 2.2.2, 99ff). Higher-level specifications of Internet services are HTTP for Web sites, SMTP for email, FTP for file transfer and Telnet for remote login (section 2.3.2, 119ff).

Changes and challenges

Even though some exuberant hopes were disappointed after the dot-com hype around the year 2000, the Internet has changed and still changes the business world substantially by offering new potentials for boundless communication and collaboration, enabling new business models and

threatening traditional industries like the music and film industry. Development of the Internet is not at its end, many challenges are still not resolved. Examples for current challenges are efficient search and retrieval and especially how contents can be structured and accessed on a semantic level (Semantic Web, section 3.2, enhanced search technologies, section 4.1), security issues (threats from viruses, secure communication, data security), unintended use of Internet resources (e.g., email spamming, excessive file sharing) as well as micro payment for services provided over the Internet.

Intranet

An Intranet is the internal network of an organization implemented with the help of technologies that based on standards of the Internet (e.g., TCP/IP, HTTP, SMTP). Advantages of Internet technologies are low costs for software, availability of skills, easy integration of internal and external systems, vendor-independent standards and their wide-spread usage. An Intranet commonly distributes internal information with the help of HTML pages and is accessible with standard Web browsers. It can deliver internal administrative information and services for employees like information about organizational units, projects and processes, booking of internal resources (e.g., meeting room, video projector, car) or requests for reimbursement of travel expenses which formerly needed paper-based forms or phone calls. Today, Intranets are an important medium for business-to-employee communication. Although the Intranet is usually connected to the Internet, it is separated from it by means of a firewall and therefore represents a secure zone, where internal information can be provided. To further enhance security, access to contents usually requires authentication, e.g., with user name and password, so that employees only have access to data that is intended for them.

Extranet

A portion of an organization's network accessible from the Internet, but only available to selected groups of authenticated users, e.g., customers or partners, is called *Extranet*. It supports the company's business processes at organizational boundaries, e.g., for procurement or order tracking, or enables joint work with the help of shared information spaces and cooperation portals.

Figure 2-9 shows how the three security zones public Internet, Intranet and Extranet are separated. The Web servers that host contents accessible in the Internet and the Extranet are placed in a so-called *demilitarized zone* (DMZ), which controls data flows to and from the public network with a firewall and uses another firewall to further protect the Intranet (see the paragraph on firewalls below). Employees have access to the Intranet by default, when working in their offices. Additionally, they often need access to the Internet and Extranet as well. Finally, they should be able to access information on the Intranet, when they are out of office. To enable the latter, they can either use dial-up connections directly to a server offering remote access services (RAS) in the company network over the public

switched telephone network (PSTN), or they can use an Internet connection to an Internet service provider and establish a virtual private network over the Internet to securely access the company's Intranet (see the paragraphs on IPsec and VPN below).

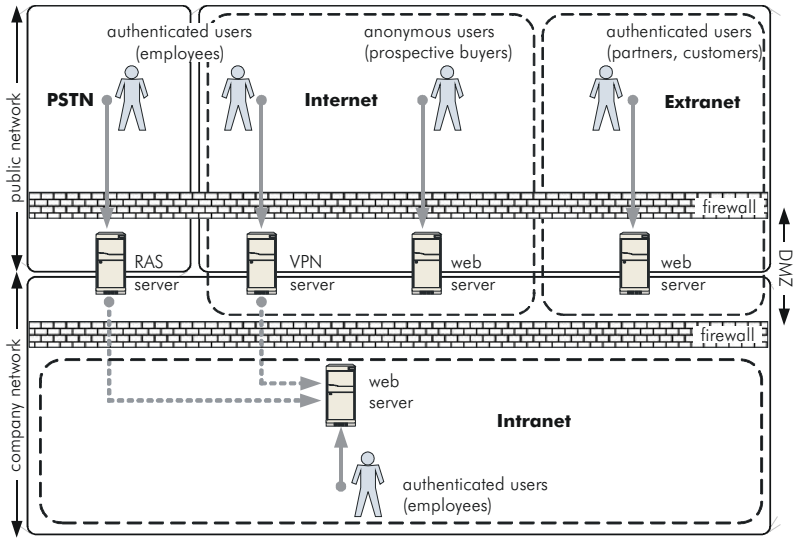

Figure 2-9. Internet, Intranet, Extranet

Security Threats. All data residing on servers or communicated through the network have to be protected against unauthorized access and other threats. For this purpose, security services must be available within the network and most importantly need to be applied wherever appropriate. The threats a company's infrastructure is facing can be grouped into the following classes:

- *Data loss* happens if important data is deleted, overwritten or lost accidentally, through external attackers or due to hardware failure. Reconstruction of lost data can be time-consuming and costly, if it is possible at all.
- Manipulation is the purposeful change of data so that it does not represent the original meaning any more. This can lead to e.g., wrong balance sheets or software code that does not work in the intended way.
- *Unauthorized access* to sensitive data is especially harming organizations if competitors get hold of business secrets. It can also be used to e.g., publish business secrets and influence the company's stock price.
- *Abuse*: company resources can either be abused by employees (e.g., by surfing in the Internet on company cost, often in combination with illegal, or semi-legal contents) or by external attackers that use e.g., hard

disk space and Internet connectivity to store and share illegal copies of videos, music files and software.

- *Downtime* means that required infrastructure services are not available or overloaded. For example, if the public Web site of a company is unavailable for a few hours, this is not only damaging the image, but could also mean loss of money due to lost sales over Web shops because customers buy from competitors.

Concrete attacks

These classes of threats can be detailed to a large number of existing attacks on enterprise infrastructures. We summarize those attacks briefly in Table 2-10 and discuss security measures afterwards.

Table 2-10. Attacks on enterprise infrastructures

attack	description
Adware	A software application in which advertisements are displayed while the program is running. These applications include additional code that displays the ads in pop-up windows or through a bar that appears on a computer screen. Some adware includes code that tracks a user's personal information and passes it on to third parties, without the user's authorization or knowledge (see spyware).
backdoor, Trojan horse	A method of bypassing authentication and obtaining remote access to a computer, while intended to remain hidden to casual inspection. Backdoors can be unknown programs that simulate useful functions (e.g., Back Orifice) or modifications of well-known programs. The term *Trojan horse* is derived from the classical myth of the Trojan horse in Homer's Illias.
brute force attack	A method to determine the decryption key of an encrypted message or a password by systematic guessing. This involves generation of a large number of keys either algorithmically or from a predetermined list. The latter is also known as dictionary attack.
(D)DoS	A *denial-of-service* attack (DoS) is an attack on a computer system or network that causes a loss of services to users, typically network connectivity. Such attacks are not designed to gain access to systems. Common forms aim at the consumption of computational resources, such as bandwidth, disk space or CPU time. Distributed denial-of-service attacks (DDoS) use multiple computers (often captured by worms) to overload the target. As the packets come from multiple sender addresses, it is difficult to repel such an attack.
man-in-the-middle	An attack in which an attacker is able to read and modify messages between two parties without either party knowing that the link between them has been compromised. The attacker must be able to observe and intercept messages between the two victims (e.g., by using spoofing). For example, communication to and from the Web site of an online bank could be eavesdropped and account numbers and passwords could be logged without being visible to users that see a simulation of the real Web site.
port scanning	Systematic checking for services presented on open TCP/IP ports, usually as part of an intrusion attempt or a security scan of administrators.

Table 2-10. Attacks on enterprise infrastructures

attack	description
social engineering	The practice of gaining people's trust to make them reveal sensitive data e.g., by calling them on the phone and telling them to give their password to the administrator for a routine check.
spamming	Sending identical or nearly identical messages to a large number of recipients without their permission. Addresses of recipients are often harvested from Usenet postings or Web pages, obtained from data bases, or simply guessed by using common names and domains. The terms *unsolicited commercial email* (UCE) and *unsolicited bulk email* (UBE) are synonyms.
spoofing	Forging the header of an IP packet, so it contains a different address and appears to be sent by a different machine. This is known as *IP spoofing*. Another form of spoofing called *DNS spoofing*, fakes DNS-names instead of IP addresses.
Spyware	Computer software that gathers information about a computer user and then transmits it to an external entity without the knowledge or informed consent of the user.
virus	A small program that can replicate and spread itself by means of placing copies of itself in uninfected files. A virus is only spread from one computer to another when an infected file is taken to the uninfected computer, e.g., if a user sends it over a network or carries it on a removable disk. Additionally, viruses can spread to other computers by infecting files on connected network folders.
worm	A self-replicating computer program, similar to a computer virus. A virus attaches itself to, and becomes part of, another executable program. However, a worm is self-contained and does not need to be part of another program to propagate itself. A worm can spread itself autonomously to other computers.

Security measures

Encryption of data stored on media and communicated in networks prevents spying and manipulation. Authentication together with restricted access to data is helpful against unauthorized access. Filtering of network packets coming into a company network or leaving it can reduce the danger of any threats and redundancy prevents the loss of data and downtimes. The following section presents implementations for these security measures. They are discussed sorted regarding their applicability to storage, access and messaging.

Data encryption

Security measures for storage. Encryption is the process of obscuring information to make it unreadable without special knowledge. A key, e.g., a password, is used to transform clear text data into encrypted data by applying an encryption algorithm. Data encryption denotes the permanent codification of data for secure storage. Communication encryption is encoding data for secure transfer over networks. There are a lot of synonyms for both encryption types like online and wire encryption for communication encryption and offline encryption for data encryption. The length of the key is an indicator for the security level of the encryption since it determines the complexity of breaking the code. Encryption and

decryption are computational complex. Thus, key length is limited. For data encryption, usually 512, 1024 or 2048 Bit keys are used whereas communication encryption uses 128 or 56 Bit keys.

Redundancy, RAID, backup

Redundancy effects storage in two ways. Firstly, hardware can be designed redundantly (e.g., a controller) so that no downtime occurs. Additionally, controlled data redundancy prevents loss of data. RAID levels 1, 5 and 10 are one way to hold data redundantly (section 2.3.1, 109ff). Another way is backup. Most users tend to forget about personal backups and it is more effective to have a large centralized backup device (like a tape library) instead of giving every user a small backup device (like a CD burner). Therefore, a central service that backs up data regularly (e.g., daily over night or weekly during week ends) over the network is advisable. Examples for backup strategies are *full backup* (all files are written to the backup device) and *incremental backup* (only files changed or added are written to the backup device). A proven strategy is daily incremental backup and weekly full back up, which represents a good balance between consumed storage space, network bandwidth and time for backup and restore. The most common media used for backups are tapes. Their advantages are low cost, long readability of data, relatively low space consumption per GB of data and an average fast access time if put into a tape library (section 2.3.1, 109ff).

Security measures for access. Encryption and authentication are combined to secure access to valuable resources. Both can occur on several layers of the ISO/OSI reference model. We will discuss encryption first.

Symmetric encryption, DES, AES

Symmetric and asymmetric encryption methods can be applied to both data and communication encryption. Symmetric encryption uses one key (e.g., a password) to encrypt and the same key is used for decryption. Well-known encryption algorithms for symmetric encryption are Blowfish, DES (data encryption standard), 3DES (triple DES) and AES (advanced encryption standard). For symmetric communication encryption, the main challenge is secure transmission of the key. This requires a secure channel, so the public Internet can not be used.

Asymmetric encryption, RSA

Asymmetric encryption uses two keys, a private and a public key (a so-called pair of keys). The private key always has to be kept secure and must not be given to anybody. The public key can be published e.g., via Internet to anybody who wants to communicate with the key owner. For example, the sender of a message (often called Alice in the security domain) takes the public key of the receiver (called Bob) and uses it to encrypt the message. Only the private key of Bob can decrypt the message. The mechanism can also be used for checking the identity of a sender. To do this, Alice takes her private key to encrypt, so everyone who has the corresponding public key can verify that the message is from her.

Digital signature

It is costly in terms of computing power to encrypt the entire message and the mechanism can be used for identification anyway. Therefore, digi-

tal signatures are used which are generated using the private key to encrypt checksums of messages (a so-called hash value, named after the algorithm used to generate checksums). The hash value uniquely identifies the message, but can not be used to reproduce its contents. The encrypted checksum is the digital signature of the message. Everybody who gets the message and possesses the corresponding public key can verify the identity of the sender as well as the integrity of the message, because if the hash value does not match the message any more, the message has been manipulated. Both mechanisms can be combined to digitally sign the message with Alice's private key and afterwards encrypt it with Bob's public key. An example for an asymmetric encryption algorithm is RSA (named after its developers Rivest, Shamir, Adleman).

Digital certificate, PKI

To guarantee that a public key published somewhere on a Web site is authentic, a system was created that embeds public keys into a kind of digital identity card called certificate. The standard for digital certificates is X.509. These certificates are issued by trustworthy companies or organizations called *certification authorities* (CA) that guarantee with their reputation and a digital signature that the public key is really owned by the person specified in the certificate. Examples for CAs are Verisign and TC Trustcenter on a global basis and Deutsche Post or Deutsche Telekom in Germany. CAs can issue certificates to persons and also to computers or other CAs. This leads to a hierarchical system with so-called *root CAs* and other subordinated CAs, which is called *public key infrastructure* (PKI). As trust in the integrity of a CA is the basis for the whole system, it is sometimes also called *web of trust*. Principally, everybody can establish a CA and issue certificates. However, those certificates are not trustworthy as they are not issued by a well-known organization. To help users determine which CAs are trustworthy and which are not, every Web browser and most other applications that deal with certificates (e.g., email clients) have a list of well-known CAs. Certificates issued by a CA not included in the list leads to warnings and users can decide to accept them temporarily, permanently or not at all. Users can also decide to put the certificate of the new CA into the list of trustworthy CAs so that every other certificate issued by this CA is also considered trustworthy in the future.

WEP, WPA

Encryption can be implemented on different layers of the ISO/OSI reference model. Moving through the layers from bottom up, we firstly discuss encryption on the physical layer. Wiretapping is possible for cables, but is much easier for wireless connections as the medium is freely accessible to anybody. Especially wireless networks with medium range (e.g., WLAN) are often accessible from outside of company buildings. The encryption standard for WLANs is WEP (*wired equivalent privacy*), which is often considered unsafe due to small keys used (56 bit) by default and cumbersome manual transmission of the shared access key. In addition, WLAN access points are often badly configured so that recent tests

still count over 50% of WLANs being open to wiretapping. The WPA standard (*Wi-Fi protected access*) is designed to overcome these problems with 128 bit encryption and easy configuration. The Wi-Fi Alliance is a consortium of companies that drives the development of WLAN standards. WPA2, the successor of WPA, uses AES for stronger encryption and became part of the IEEE 802.11i specification that deals with enhancements to the WLAN protocol family.

IPsec

Encryption can also be applied at the network layer. One of the most commonly used protocols there is IPsec (*IP security*). As the name suggests, it builds upon the Internet protocol (IP). It is an essential part of IPv6 and an optional addition to IPv4. Asymmetric encryption is used to encrypt IP-payloads (*transport mode*) or entire IP-packets including headers (*tunneling mode*). The latter is used if the final destination of the packet is not using encryption and the end-to-end encryption is established transparently between two machines between the sender and receiver (e.g., to secure the packets' way through the Internet that connects two company Intranets). IPsec is the only way to secure UDP-based communication as all other security protocols build at least partly upon TCP to establish connections that are needed for key exchange. This raises complexity, since IPsec has to deal with reordering and fragmentation of network packets on its own.

VPN

If secure tunnels through the public Internet are used to connect two private networks, we speak of a *virtual private network* (VPN). Servers responsible for the encryption are often central servers also managing Internet access or providing firewall functionality (see below). They encrypt IP packets and wrap them in a new IP envelope addressed to their counterpart in the other private network. The packet is then routed through the Internet, unwrapped and decrypted at the VPN server in the second private network. Afterwards, the unencrypted packets are routed to their final destination. IPsec is often used for users that use dial-up connections to an Internet service provider (ISP) and want to connect to their organization's Intranet. The tunnel is then being established between the client and the VPN server. Other protocols that can be used for VPNs are PPTP (*point-to-point tunneling protocol*) and L2TP (*layer two tunneling protocol*) which both are emulating a secure layer two infrastructure, so that there is one additional layer compared to IPsec (Figure 2-10).

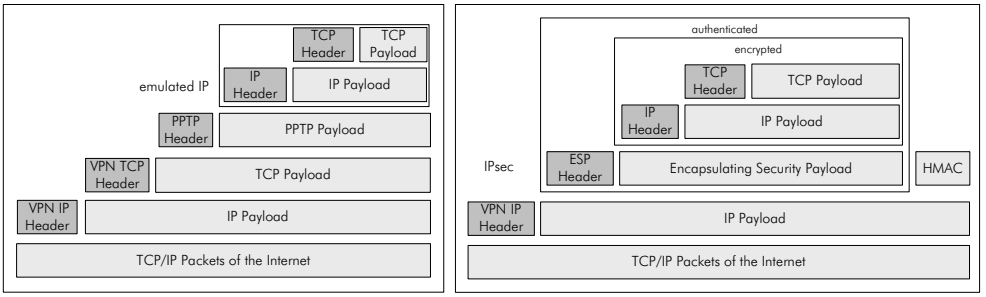

Figure 2-10. PPTP and IPsec packet layers.

SSL, HTTPS

Between transport and application layer, *secure sockets layer* (SSL) can be used to encrypt communication. Like IPsec, SSL uses a public key infrastructure and asymmetric encryption. It is widely used for Internet banking and transactions in online shopping and considered relatively secure when using 128 Bit keys. SSL encrypted HTTP connections are also known as HTTPS and use port 443. The successor of SSL is *transport level security* (TLS) which is also used as part of other protocols.

Authentication

Authentication means that users have to be uniquely identified in order to get access. Authorization is used to determine whether users are privileged to access the respective system or network (section 3.3, 198ff). Three different ways of identification are possible. Users can authenticate using a unique possession (e.g., an identity card), knowledge (e.g., a password) or characteristic (e.g., biometric characteristics like iris or fingerprint). Often a combination of these is used to make authentication more secure (e.g., EC-card together with a PIN). Authentication is used in different situations, either when users access a certain service in the Intranet, or before a connection to the Intranet is established.

Kerberos

Kerberos is an authentication protocol widely used within corporate Intranets for the first situation (default in Windows 2000 based infrastructures). It works in three phases:

1. Clients talk to the authentication server component and send the user name in clear text to it. The server looks up the corresponding password and uses it to send an encrypted message back to the client consisting of a *session key* and a key for the *ticket-granting server*. The client then uses the password provided by the user to decrypt this message. If the password is correct, the decryption will work and phase two can start. The user name was deleted on the server in the meantime.
2. The client uses (a) the session key to encrypt a timestamp and (b) the ticket key to encrypt the user name and the session key. These two parts are (c) sent to the ticket granting server together with the name of the service it wants to use on the server. The server uses the ticket key to decrypt the session key and sends back (i) the key for the requested ser-

vice and (ii) a session ticket that depends on the timestamp and is valid only for the requested service and a limited time frame.

3. Finally, the client issues a request to the service it wants to invoke. It exchanges the session ticket using the services key. The service answers by sending back the timestamp increased by one in order to grant its identity and the whole handshake is complete.

The advantage of this system is that the user's password is never communicated over the network. The disadvantage is that the protocol is complex and slow.

RAS

An example for the second authentication situation is authentication of users that use dial-up connections to the company, e.g., from a customer site or from their home office. Public phone lines are considered substantially more secure than the public Internet. Therefore, authentication is more important than encryption. The service that provides these dial-up connections for modems or ISDN-adapters is called *remote access service* (RAS). Authentication protocols used here include PAP (password authentication protocol), CHAP (challenge handshake authentication protocol) and EAP (extensible authentication protocol).

The ISO/OSI reference model gets a bit shaken if you try to apply it to dial-up networks, because the protocols used are partly on higher levels, than the protocols that run on top of them. The reason is that dial-up protocols emulate low level ISO/OSI layers. For example, the *point-to-point protocol* (PPP) resides on layers 3 and 4 and runs on top of ISDN (layers 1 and 2). TCP/IP runs on top of PPP which also covers layer 3 and 4 respectively (see also Figure 2-10 on page 133).

Filtering

Filtering in this context means analyzing network packets, messages or files, checking them for certain characteristics and executing a defined action (e.g., delete, pass through) based on the findings.

Firewall

A service implementing filtering that analyzes network traffic and blocks unwanted communication is called firewall. One has to distinguish between network or enterprise firewalls on the one hand and personal firewalls on the other hand. *Network firewalls* are central devices or computers that reside somewhere in the network and serve as kind of sluice where all the network packets have to pass. Firewalls are often combined with routers as both have to control each network packet (though with a different purpose). *Personal firewalls* are applications that run on a PC and block unwanted traffic to and from this single PC. Firewalls decide upon letting a network packet pass through or blocking it based on rules. Those rules consist of (1) sender IP, (2) destination IP, (3) sender port, (4) destination port, (5) transport protocol, (6) direction, (7) time and (8) a block or pass through instruction. It can be further specified to passively drop the packet or actively deny it. In the first case the sender gets a `time out` message as response, in the second case the sender gets a `service denied` message.

Personal firewalls can additionally inspect the sending or receiving application and are therefore also called *application-level firewalls* in distinction to *packet-level firewalls* that do not have this ability. Sophisticated firewalls support *stateful inspection* which enables them to decide not only based on the current packet, but also based on history of network traffic. For example, a request can be blocked if the same sender address has already sent the same packet ten times before.

Intrusion detection

Intrusion detection systems (IDS) are trained with complex patterns of network packets that indicate attacks to the Intranet. If such a pattern is recognized, the administrator is alerted and if possible the attack is prevented.

Content filter, proxy

Content filtering works on the application layer and is used to prevent employees from accessing Web sites with offending or illegal contents and thus preventing abuse of company resources (hardware and Internet connection). Often, content filters are installed on servers acting as proxies. A proxy has several functions. Outgoing HTTP connections are pooled and can be monitored more easily. Only authenticated and authorized users can access WWW, FTP or other services. Web sites that are accessed can be cached on the proxy so that they are available faster if several employees access the same site which reduces Internet traffic. Internal IP addresses are hidden from the public Internet which saves money as only one public address is needed and also prevents port scanning attacks against clients.

Network address translation

Network address translation (NAT) is used for translation between internal packets (with a client's address as sender) and packets going out to the Internet (with a proxy's address as sender) and can be used independently of proxy servers. Requests to the Intranet get translated to the IP address of the NAT server and the original sender port is being mapped to a new port on the NAT server. Many clients can be served since there are 65535 ports available and each client usually has only a few open connections at the same time.

Antivirus program

Security measures for messaging. Computer viruses are a threat that becomes an increasingly serious problem for companies. Viruses can be harmless and e.g., just display a message at a certain time, but can also be harmful and e.g., delete important files or damage the operating system. The removal of such viruses is costly even for viruses without a harmful part. Antivirus programs detect viruses based on characteristics of its program code that can either be a complete program file (.exe or .com) or can be part of a normally functioning program file. A centrally-hosted anti virus software (together with intelligent patch management to timely close security vulnerabilities of software) has become essential for organizations. Today, attachments of email messages are the most important source for computer viruses and other types of malicious software. Therefore, an antivirus program should be installed on the email server in addition to the one on the proxy server.

Spam filter

A second filter mechanism that should be installed on an email server is a spam filter. Spam or junk emails are email messages that are unsolicited from the recipient, often advertisements, but also other email messages that are distributed to masses of recipients. A synonym for spam email is junk email. Spam filters try to differentiate between wanted and unwanted email messages based on rules that define characteristics of spam emails (e.g., sender address, keywords). One challenge that arises is that wanted email messages should be lost by no means. Therefore, messages that are classified as spam are usually not deleted, but moved to a junk mail folder or appropriately labeled and then transferred to the recipient.

PGP, S/MIME

Encryption can also be applied to email messages. PGP (*pretty good privacy*) is a free tool that can be used to encrypt email messages using asymmetric encryption and works together with most common email clients. Due to its wide-spread use with millions of users world wide, it is the de-facto standard. Secure MIME (S/MIME) is a standard developed and maintained by the IETF (Internet engineering task force). Both have in common that they use a public key infrastructure and support encryption as well as digital signing of messages for sender identification and integrity checking.

2.4 Application Infrastructure

Software applications build on basic services for storage, access, messaging and security discussed in previous sections. They support users by reproducing business logic with electronic functions. This helps users to accomplish and coordinate their tasks by e.g., automating workflows as parts of business processes. Large parts of business logic are different for every industry, business process and maybe even company and is supported by business applications which can be both, standard or proprietary software systems. However, there are some parts of an application that are very similar or identical for almost every application. The application infrastructure is a foundation for applications and offers common functionality in terms of services, e.g., user management, transaction handling, object persistency. To understand this, first some principles of component oriented programming are discussed which is the key to application infrastructure.

2.4.1 Application Server

History of software development

In history of software development, programming evolved from low-level *coding in machine languages* (e.g., assembler) towards more abstract higher-level programming languages (e.g., Basic) that are closer to human

understanding, structuring and solving of problems. Along with that, software got more sophisticated and complex so that mechanisms had to be found helping to deal with that complexity.

A first step was *procedural programming* that allowed better structuring of code with sub-routines, also called functions or procedures (e.g., Pascal or C). The next step was to encapsulate data and functions for manipulating this data in *object-oriented programming* (OOP) so that data is hidden from the object's environment and thus can be manipulated and accessed only in a clearly defined way. Each object was responsible for its own local data (called attributes) and provides functions (called methods) that other objects can invoke in order to retrieve or manipulate this data in a defined way. Every object can be tested on its own which reduces complexity. Examples for OOP languages are Smalltalk, C++ and Java. With the advent of software that is distributed over several computers the need for new concepts arose again. *Component-oriented programming* was introduced with the promise to provide these new concepts.

Definition of software component

Software components are binary units of independent production, acquisition, and deployment that interact to form a functioning system (Szyperski 1997).

Components could be seen as a unit in between an object and a program. They are not able to run on their own. They need a container that executes them and provides additional services. This container is called application server. The following paragraphs give an overview of the services an application server provides for components.

Distribution

With OOP, it is possible to invoke methods of objects that run on other computers (e.g., with Java RMI or Microsoft DCOM, section 3.4.1, 203ff). The programmer has to specify exactly where this object is and what name it has. An application server releases the programmer from this burden by providing a distribution transparent way of invoking remote methods. One component calls another without having to care whether the other component runs on the same machine or not.

Load balancing

Application servers also take care of optimal distribution of the components by distributing them equally across all application server instances, so that the available machines are efficiently used which is called *load balancing*. The precondition for this is that multiple instances of one component can be created. Application servers also take care of *multi-threading* and thread coordination. To run multiple server instances on multiple machines that act as one logical server to the environment is called *clustering*. This mechanism should also be supported by application servers. The third related mechanism is called *fail-over* and provides for replication of state information and take over of user requests if one server fails.

Transaction handling

Enterprise applications frequently run into situations where several tasks have to either run completely successfully or have to be rolled back.

This group of tasks is called transaction. An example for a transaction is a credit transfer. If 100€ should be transferred from one account to another and one account is charged 100€, then the money has to be added to the other account. If the latter does not work, withdrawal of money from the first account has to be cancelled, so that no money is lost. A mechanism that guarantees that is called transaction handling. In OOP, programmers have to take care on their own that transactions are handled correctly. With component-oriented programming, programmers only have to specify which functions are involved in a transaction and the application server monitors correct execution of transactions with a final *commit* or a *rollback* of the entire transaction.

Access control

User authentication and authorization, i.e. the identification of a user and assigning of roles, privileges and respective granting of access to functions or data is another central part of any enterprise application (section 3.3). In contrast to OOP, programmers do not have to write their own user management module, but only have to specify which roles a user has been assigned, what privileges are associated with these roles and what privileges are required to access a function. The rest is done by the application server.

Persistence handling

Some data held inside components should be kept permanent. This data is usually stored in (relational) data base management systems. In OOP, programmers have to write calls to a data base middleware that instantiates the driver for the DBMS, establish a connection to the data base, execute SQL statements and close the connection afterwards. This is inefficient and tedious. With an application server, programmers only specify an object-relational mapping from attributes of the component to fields and tables of the data base. Loading from and saving to the data base is handled by the application server. It guarantees that data is stored persistently and kept current. The application server also performs runtime optimizations so that not every single data base operation has to open its own connection which is called *connection pooling*. The mechanism of making object data permanent is called persistence handling.

Maintenance

An enterprise application follows a certain life-cycle: additional functions are implemented, errors are corrected in the source code and the hardware that executes the application eventually has to be upgraded or supplemented with increasing numbers of users or transactions. These episodes in the application life-cycle usually lead to unavailability of the system due to maintenance tasks. An application server allows loading of new program code (e.g., extensions, corrections) during runtime. This feature is called *hot deployment*. The class files with the additional or corrected code are copied to a specific folder in the file system (hot folder) and the application server loads it automatically as soon as possible. If an additional server is added to a cluster, or a server is stopped to upgrade the hardware, the system does not have to be stopped. To help administrators to identify

potential problems or performance bottlenecks, application servers provide extensive functionality for logging, monitoring and auditing. *Logging* means that the system writes a log entry to a log file or to the system's event log every time an exceptional event occurs (warning or error). *Monitoring* means that the application server provides current status information about resource allocation (disk, memory and network) and runtime state (e.g., number of users logged in). *Auditing* is often based on logging and means in this context the systematic verification of user actions or application transactions.

Runtime optimizations

Application servers relieve programmers to a certain extent from manual optimization of runtime performance. Besides the above-mentioned connection pooling and load balancing, application servers also support caching of data base queries, temporary persisting of components that are currently not used and complete management of the life-cycle of a component. Caching is keeping data in memory (or on fast media in general), although they are currently not needed any more in order to have them quickly available when they are needed next time. Components not needed at the moment can be destroyed in order to free resources. If such a component is needed later on, a new one is created (*life-cycle management*). Component holding state information that are important in the future must not be destroyed. Therefore, their state is stored on a disk which is called *dehydration,* before the component is removed from memory. If it is needed again later, its state is restored from disk and the component can be used again.

Differences at a glance

Functions of application servers described above differ from traditional programming only in the way they are invoked. Instead of making explicit calls to specific middleware APIs that do the job, only the need for such a call is specified in component-oriented programming and the application server fulfills the need at runtime. Thus, these mechanisms are also called *implicit middleware*. It can be seen as a higher level of abstraction. To stress an important part of the discussion above we can note that OOP supports reuse at build-time whereas component-oriented programming supports reuse at runtime.

Web front ends

An additional trend in developing enterprise applications is the need for an easy way to generate Web front ends for the applications, due to the fact that Web browsers become the main access tool (section 5.2, 320ff). Application servers address this need by providing HTML derivatives that allow to embed server-side script code that is evaluated at runtime, so that dynamic HTML pages can be realized. Template libraries that help to compose GUI elements from basic HTML tags further support ease of use.

2.4.2 Component Frameworks

CORBA

The first standard approach to component systems is CORBA (common object request broker architecture) that has been developed by the Object

Management Group (OMG). The first widely adopted version 2.0 was specified in 1995 and released in 1997. Current version is 3.0 from 2002. The core of a CORBA system is a central application called *object request broker* (ORB) where all objects have to register, so that other objects can reference them and invoke their methods. The protocol used to communicate with the ORB is called IIOP (Internet inter-ORB protocol, TCP port 684). The name already tells that it can be used not only for communication from objects to ORB but also from one ORB to another. The methods that an object wants to register are specified in a language called IDL (*interface definition language*). IDL provides a set of data types independent of programming languages that can be used for parameters and return values. Each supported programming language has to provide a mapping from IDL data types to language-specific ones. Thus, CORBA provides both language and distribution transparency, but it does not provide a complete application infrastructure according to the functions that were discussed above.

J2EE

Java offers a different approach and can be seen as a reference for component frameworks. Services discussed above in section 2.4.1 are provided by a central application called Java application server (JAS). There is an overwhelming number of technologies and standards connected to JAS subsumed under the headline *Java 2 enterprise edition* (J2EE, in contrast to Java 2 standard edition J2SE and the Java 2 micro edition J2ME). J2EE is under control of Sun Microsystems and is a definition for a large set of interfaces on the one hand and an implementation of these definitions (Java class files) on the other hand. Besides the free implementation from Sun, there are several commercial and open source implementations of the J2EE specifications. IBM, BEA and others provide own commercial implementations with partly significantly increased performance. JBoss is an example for an open source implementation that receives increasing interest from business customers.

JRE, JVM

Java is an interpreted language which means that every Java program needs the *Java runtime environment* (JRE, also called *Java virtual machine*, JVM) in order to be executed. The Java compiler does not create directly executable machine code, but Java bytecode which is interpreted by a JVM. The reason is platform-independence as Java programs run on any platform that has a JVM. The disadvantage is less performance because of interpretation overhead. This disadvantage is partly compensated with *just-in-time compilation* (JIT), the compilation of byte code into machine code at runtime.

EJB

An application server can be seen as an extension of a JVM that provides additional services. The central concept in J2EE are *Enterprise Java Beans* (EJB) which must not be mixed up with Java Beans, a client-side technology. EJBs are components running in an application server. There are different types of EJBs, namely entity beans, session beans and mes-

sage-driven beans. *Entity beans* are briefly spoken wrappers around data or business data encapsulated in objects. They usually load and save their data from and to a DBMS and are therefore persistent. Application servers provide a mechanism for automatically handling data base operations which is called *container managed persistence* (CMP). *Session beans* hold business logic to manipulate entity beans and can be seen as a software implementation of a workflow action or an entire workflow. They are usually not persistent and their lifetime is (despite the name) not necessarily as long as a user session lasts (although it can be). Session beans come in two flavors, *stateful* and *stateless*, meaning that they preserve an internal state over time and various method calls or not. *Message-driven beans* are messaging objects that are similar to session beans as they represent actions, but in contrast to them process requests asynchronously.

Supporting technologies

In the following, technologies are described that supplement EJBs. The *Java messaging service* (JMS) is used by message-driven beans and acts as a vendor-independent interface to message-oriented middleware (section 2.3.3, 123ff). Therefore, it is similar to the *Java data base connectivity* (JDBC) which provides vendor-independent interfaces to communicate with a DBMS. It is not surprising that there is also an interface to directory services called *Java naming and directory interface* (JNDI). JNDI plays an important role in J2EE as it is used to register and locate components. Without JNDI, there is no communication between enterprise beans which does not mean that an LDAP directory service is required for a small J2EE application. A simple implementation for JNDI (often without persistence) is delivered with every application server. Other APIs worth mentioning in this context are *Java mail* for email messaging, *Java transaction API* (JTA) and *Java transaction service* (JTS) for secure transaction handling and *Java authentication and authorization service* (JAAS) for securing access to applications.

The following specifications are not directly related to the features described in section 2.4.1, although they are essential for modern applications. The *Java API for XML parsing* (JAXP) provides interfaces for XML parser (section 3.1.3, 153ff), the *J2EE connector architecture* (JCA) provides a defined way of accessing third-party enterprise systems (e.g,. on IBM mainframes or ERP systems like SAP R/3). *Java servlets* and *Java server pages* (JSP) provide functionality for Web presentation of the application. Altogether, Java application servers provide all functions of application servers as described above and offer even more.

.NET

Application infrastructure provided by Microsoft is tightly integrated into the Windows operating system which makes it harder to locate them since no central application exists. Component technologies of Microsoft are subsumed under the headline .NET (spelled “dot net”). Starting with the similarities to Java, .NET is a platform for interpreted programming languages. The runtime environment is called *common language runtime*

(CLR) and is interpreting code in Microsoft *intermediate language* (IL). There is also a JIT compiler that can be used to translate IL code into native machine code at runtime. The runtime environment is currently available for Windows platforms from Microsoft. An open source implementation of the .NET framework called *Mono* which is currently sponsored by Novell brings at least the core APIs of .NET to Linux and some other UNIX platforms.

COM+

The corresponding technology to EJBs is called *.NET managed components*, often referred to as *COM+ components*. COM+ is an enhancement of COM (*component object model*) and DCOM (distributed COM) and does not necessarily have to be a .NET managed component, so it is not quite right to mix those two up. .NET managed components are a subset of COM+ components. To make the confusion complete, sometimes the term `EnterpriseService` is used in that context which is the name of the packet that includes the root class (`ServicedComponent`) all components must inherit. COM+ components are equivalent to session beans.

MSMQ, ADO.NET, MTS, DTC

Message-driven COM+ components can be realized with the help of the *Microsoft Message Queue* (MSMQ), but there is no equivalent for entity beans. Persistence can only be managed manually or with the help of third-party add-ons to .NET. The data base access works with the ADO.NET API (ADO = *ActiveX data base objects*). The `System.DirectoryServices` namespace provides classes for communication with directory services. In contrast to J2EE, directory services are of minor importance in .NET. Transaction support is provided by the *Microsoft transaction service* (MTS) and the *distributed transaction coordinator* (DTC). The `System.Web.Mail` namespace bundles classes for email messaging. Authentication and authorization is not packaged within a special part of the .NET framework, but spread across multiple namespaces. The most important ones are `System.Web.Security` and `System.Net`. Not surprisingly, the recommended authentication modes are Windows integrated authentication (either with local users of the server or users in a Windows domain) and role-based authentication using group-membership information from Active Directory. However, it is also possible to use forms authentication (user name and password are entered in an HTML form) or Microsoft passport authentication as well as storing data about roles in SQL Server or Access data bases.

ASP.NET

Corresponding technologies for most other Java APIs described above exist as well. Classes for XML handling are in the `System.XML` namespace, Web front-ends are created using the ASP.NET API. Only a JCA equivalent is missing. The only product that could be seen as a single connector in a JCA sense is the MS *host integration server* (HIS), which can be used to connect to IBM mainframe systems.

Main differences between .NET and J2EE are that .NET is programming language independent (C#, Visual Basic, C++ and some other lan-

guages can be used to generate IL code) whereas Java is platform-independent. The infrastructure for J2EE is bundled mainly in a single program called application server whereas .NET uses a number of services from different Microsoft products, e.g., Windows, MSMQ and the Internet Information Server (IIS). Another difference is that there is no .NET support for container-managed persistence.

Questions and Exercises

1. Make yourself familiar with the commands `ipconfig`, `ping` and `tracert` (on Windows XP and Windows 2000, for Linux and Mac OS the command is `ifconfig`). Use the option "`/?`" or "`-?`" to get a help message that explains the usage of the respective command line tool. Note the IP address of your own computer and determine to which network class your computer belongs!
2. Use the `tracert` command to find out how many hops a network packet has to make to reach a server in New Zealand (e.g., www.nzx.com). The hostnames of routers often contain information about the location of the router and the provider that owns the network (e.g., `sl-bb22-chi-15-0.sprintlink.net` is located in Chicago and owned by Sprintlink). Which route does the ICMP echo packet travel? Identify major network providers! Use `tracert` several times with different targets which reside in different countries (e.g., University Web servers). Identify major network hubs.
3. Send an email message from a fictive address to your regular email account. All you need is an SMTP server where you have an account and a telnet session on port 25 to this server. First of all you should say hello to the server and tell him the name of the sending machine. The corresponding command to do that is "`HELO name`". Then you tell the server the sender address with the `FROM:sender@domain.com` command. Then you say who should receive the mail with `RCPT TO:receiver@domain.com`. Each command is followed by a <return>. Then you can start the message with `DATA:` followed by a return. Within the message body, you can use the `SUBJECT:here comes the subject` command to specify the subject and type a short message in the next line. Conclude the message with a dot in a new line and press return again to send the mail. The server should state something like `message accepted for delivery`.
4. Try to receive an HTML page from a Web server using telnet. Connect to `www.wiwi.uni-halle.de` on port 80. Type `GET` and press return to get the start page of the server. Why is the connection lost after the HTML code was transferred? Use a browser to connect to the Web server and have a look at the page source (right click). Try to retrieve the Web page `maier/index.php` from the same Web server. Use `GET maier/index.php` in the telnet session to port 80. Why is the content different from the page displayed in the browser?
5. Use the command line version of an FTP client (comes with Windows 2000 and XP and is also available on most other operating systems) to download files from an FTP server. Open a command window and type `ftp`. Open a connection with `o server_DNS_name`. List the contents

of the directory with `list` or `ls`. Change to other directories on the server with `cd directory_name`. You can do most usual file operations on the FTP server, like `cd` (change directory), `md` (make directory) and `rm` (remove file) if you have permissions to do so. You can do the same operations from within you FTP client on your local system if you put an exclamation mark "!" in front of the command, so `!ls` lists the contents of a directory on your local machine. If you have identified a file on the server you want to download use `get filename` to initiate the transfer. With `close` you can end the session and leave the client with `quit`.

6. Install a freely available personal firewall on your computer that allows you to create rules regarding ports, addresses and protocols (e.g., AtGuard, iptables, Kerio, Tiny). Create rules that allow outgoing ping commands, DNS resolution and access to the Web over HTTP.
7. Advanced readers can install a network sniffer (e.g., Ethereal). Start to capture the network traffic, then do a `ping` and visit a Web site. Then stop capturing the network traffic. Why are there UDP packets going to port 53?
8. Use the "component services" in the "administration" program group (Windows 2000 or XP) to inspect the COM+ components that are installed on your system.

Further Reading

Since many networking technologies like TCP, IP and HTTP are well established and have been quite stable for a decade there are several good textbooks that deal with the network basics. For application infrastructure, currently only books that deal with either EJB or .NET are available on the market.

Tanenbaum, A. S. (2003): *Computer Networks*, 4th edition, Prentice Hall, Upper Saddle River, New Jersey 2003

Tanenbaum covers every aspect of computer networks in great detail and can be seen as the reference in that area. Despite its depth and technical style, it is still very readable.

Roman, E., Ambler, S., Jewell, T. (2002): *Mastering Enterprise Java Beans*. 2nd edition, Wiley Publishing, New York 2002

The Java 2 Enterprise Edition with Enterprise Java Beans as their main concept can be seen as the mother of modern application servers. This book describes the challenges for application developers, how to solve them with application servers and discusses the details of related Java technologies.

Löwy, J. (2003): *Programming .NET Components*. O'Reilly, USA 2003

The book provides a complete introduction of the .NET framework at a detailed technical level together with a focus on component-oriented programming. The author shows how to build large-scale enterprise applications from architectural aspects up to concrete technical solutions with Microsoft technologies.

3 Integration Services

Chapter 2 gave an overview of the basic information and communication infrastructure which can be found in many organizations. Advanced services to support knowledge work are installed on the basis of this infrastructure. However, before we can touch specific knowledge services that directly target knowledge work processes, we have to first discuss the fundamental integration issue that is on hand when one implements an enterprise-wide knowledge infrastructure.

Term integration

Integration can be traced back to the Latin word "integro, integrare", restore, re-establish and means "to form, coordinate or blend into a functioning whole" (Merriam-Webster Dictionary). With respect to EKI, integration services coordinate a variety of data and document sources as well as infrastructure services that are blended into a semantic web of knowledge resources as well as pointers to knowledge resources.

Dimensions of integration

Integration in information processing can be classified according to the following dimensions (Mertens 2001, 2):

- *object*: (1) data, (2) functions, (3) workflows or processes, (4) methods and (5) programs,
- *direction*: (1) horizontal integration along the value chain and (2) vertical integration from administration (transaction) systems to planning and control systems,
- *scope*: (1) integration within a specific area, e.g., a process or an organizational unit, such as production, (2) integration spanning organizational functions or processes, (3) organization-wide integration and (4) inter-organizational integration,
- *degree of automation*: (1) automated with all integration tasks being fulfilled by the system, (2) system-initiated user intervention and (3) user-initiated semi-automation of integration tasks.

EKI integration focusses data, functions and workflows/processes as the main *objects*. It can be seen as primarily targeting a horizontal integration along the knowledge life-cycle rather than the business value chain as the main *direction*, with the *scope* of an organization-wide or even an inter-organizational integration. Finally, in terms of *degree of automation*, it aims at a fully automated integration, but nevertheless most often will require some form of human intervention in order to e.g., semantically integrate two sources of documents.

Knowledge services work on the basis of integration services, e.g., a knowledge repository which handles the organization's meta-knowledge describing knowledge elements that come from a variety of sources with the help of meta-data for a number of dimensions, e.g., person, time, topic,

location, process, type. A taxonomy, a knowledge structure or an ontology help to meaningfully organize and link the knowledge elements and are used to analyze the semantics of the organizational knowledge base.

Overview

Integration of knowledge elements is based on data integration (section 3.1). Particularly, XML, XML Schema and XSLT are discussed which offer a meta-language for annotation, description of the structure and transformation of semi-structured data. Then, semantic integration brings together knowledge elements on a conceptual level (section 3.2). Based on XML, a number of efforts have been made to specify the semantics of documents, e.g., in the Semantic Web approach. Moreover, integration services are needed to manage meta-data about users working with EKI, e.g., in directory services. Due to the importance of users as participants accessing EKI, integration of user data is discussed separately (section 3.3). EKI integration cannot, however, rely exclusively on data integration. Function and process integration (section 3.4) briefly describe the most important technologies that are used to connect applications. The focus of this chapter will be on the semi-structured data, especially the semantic integration. This is due to the fact that it is in this respect that EKIs differ most from more traditional approaches to enterprise application integration. The latter focus the integration of primarily relational data, function and process integration along the business processes of an organization.

Learning objectives

On completion of this chapter, you should be able to

- classify integration activities with respect to EKI,
- apply XML Schema and XSLT for the development of simple document definitions and for transforming simple instance documents,
- describe Web resources with the help of RDF and develop class models in RDF Schema,
- analyze the potentials of OWL for semantic integration of various knowledge sources described by individual ontologies,
- analyze the advantages of and potential barriers to an organization-wide management of user identities,
- give an overview of the most important approaches to function integration,
- elaborate on the challenges of describing and automating business processes with the help of e.g., the business process execution language.

3.1 Data Integration

Structured data and relational data bases

Integration of data has been a concern for many years. Codd's relational theory as a model for organizing and handling data in 1970, Chen's entity-relationship model in 1976 and the ANSI three-layer architecture of data base management systems can be seen as major steps towards a unified view of well-structured data. The unification of the way (transactional) data is handled in organizations is owed to the wide-spread use of relational data base management systems. This eases data integration not only within organizations, but also between organizations. The organization of structured, transactional data has been well-understood for years. There are many textbooks on data (base) management (e.g., Watson 2002).

Semi-structured data

However, the amount of semi-structured and unstructured data, such as (text) documents, messages, images, media files or Web content has grown substantially and needs to be integrated as well. The integration of these data sources requires other approaches.

Section overview

In the following, we will give a brief description of the various concepts and elements of the Semantic Web stack (section 3.1.1). The lower levels are discussed in this chapter as they remain on the level of addressing and a uniform character set (section 3.1.2), in-line description of (semi-)structured data with the help of XML (section 3.1.3), conceptual level restricting elements and tags allowed in the description with XML Schema (section 3.1.4) as well as the standard vocabulary for transforming XML documents, XSLT (section 3.1.5). The higher levels, starting from RDF, are discussed in section 3.2.

3.1.1 Semi-structured Data

Structured and semi-structured data

As mentioned above, the differentiation into structured and semi-structured data is often found in e.g., document management or digital asset management (see also sections 2.3.1, 109ff and 4.2.1, 247ff). However, there is no clear demarcation between structured, semi-structured and unstructured data. Generally, all data can be stored in data base systems, even unstructured data in the form of binary large objects (BLOBs). The differentiation is rather of a technical nature. It is postulated that the handling of semi-structured data requires somewhat different technical solutions from relational data base management. These solutions are on the one hand systems specially designed for managing semi-structured data, e.g., content management systems and document management systems. On the other hand, standardization of languages to describe data differ as well. Whereas SQL is the widely accepted standard to define and manipulate structured, data base-oriented data, standards based on XML are used in the realm of semi-structured, content- and document-oriented data.

Standards, institutions and initiatives

A number of institutions have developed standards and started initiatives to provide comprehensive frameworks for definition and exchange of meta-data, i.e. semantic information about documents, especially about books, journals, images, photographs, audio and video files. Examples for institutions, standards and initiatives are the World Wide Web (W3C) consortium with XML and the Semantic Web initiative, the International Standardization Organization (ISO) with a large number of standards for document exchange, e.g., the Motion Picture Experts Group (MPEG) 7 meta-data standard for images, audio and video files or the Topic Map standard as well as the Dublin Core standard for exchanging meta-data about text documents which was set up by a consortium including large public libraries.

Structuring, describing, translating, storing and securely accessing semi-structured data as well as reasoning about semi-structured data require a substantial effort. Figure 3-1 structures the main technologies that are involved to support these tasks.

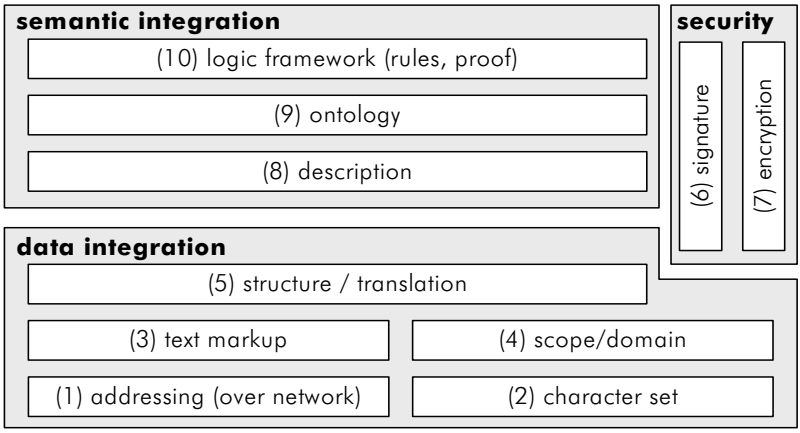

Figure 3-1. Overview of data and semantic integration[1]

Levels of semantic integration

Semantic integration of semi-structured data is a complex undertaking. Thus, the Semantic Web initiative breaks down the variety of tasks into a layered structure that helps to understand what concepts have to be defined so that semantic information about knowledge elements can be easily exchanged between a variety of heterogeneous ICT systems.

Data integration

Data integration requires that agents (users, institutions or applications) exchanging data agree (1) how to address data resources over a network, in the most generic sense over the Internet, (2) about what character set to

[1] This figure is a conceptual abstraction of the Semantic Web stack by Berners-Lee et al. (2001) discussed in section 3.2, 171ff.

use, (3) about the internal structure of documents, also called text markup, (4) about the scope or domain in which the specified names in the markup are valid, (5) about how to define a schema, a structure of the elements in the semi-structured text and how to translate a document that is an instance of one schema so that it conforms to another schema.

Security

Secure access to semi-structured data requires technologies for verifying both, the sender and the receiver of data with the help of (6) electronic signatures and technologies that prevent eavesdropping with the help of (7) encryption which were discussed in section 2.3.4, 124ff.

Semantic integration

Based on the standards that support the internal structuring of documents corresponding to a schema, semantic integration aims at providing standards for describing documents or, more generally, Web resources. This is done with the help of (8) statements that describe the resources, (9) ontologies that show the relationships between the concepts used in the descriptions, and (10) rules as well as a logic framework that allow for reasoning about documents and their descriptions.

Figure 3-2 focusses on data integration and gives an overview of the standards being part of the Semantic Web initiative which correspond to the three levels of data integration in Figure 3-1. These standards will be discussed in the remainder of this section.

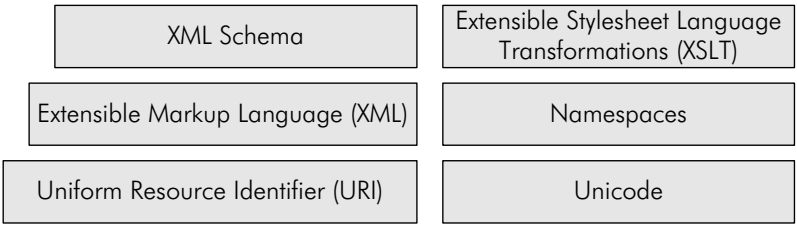

Figure 3-2. Data integration standards as part of the Semantic Web stack

3.1.2 Uniform Resource Identifier and Unicode

The fundamental concepts that form the base layer of the Semantic Web stack are universal, standardized identification (Unified Resource Identifier) and a universal, standardized character set (Unicode).

Uniform Resource Identifier. Unified Resource Identifiers (URIs) are formulated in a standard syntax that is used to support information exchange in the Internet, particularly on the World Wide Web. Their aim is to provide a means to uniquely identify objects (or resources) located in any directory on any machine in the network accessed via a specified access method. Resources in turn are coded (or deliver code) in the character set Unicode. There are two types of URIs, URLs and URNs.

Uniform Resource Locator (URL)

A URL is a type of URI that identifies a resource via a representation of its primary access mechanism (e.g., its network location). An example URL is `http://www.wiwi.uni-halle.de/maier/index.htm`. The Uniform Resource Locator (URL) comprises information on how to access a resource and provides

- the name of the *protocol* to be used (in our example: `http`),
- the *address*, either a domain name or an IP address, of the machine on which the resource is located (in our example the domain name `www.wiwi.uni-halle.de` of the Web server of the School of Business and Economics of Martin-Luther-University Halle-Wittenberg),
- the *hierarchical path* describing the location of the resource on the machine (here: the folder `maier`) and
- the name of the *resource* (here: `index.htm`).

URLs can point to any resource, e.g., to files, queries, documents stored within data bases or the results of a finger or archie command, using any access method that a browser understands. Examples are file URLs, Gopher URLs, HTTP URLs or News URLs.

Uniform Resource Name (URN)

A Uniform Resource Name (URN) is a locally unique and persistent identifier of a resource in a namespace that dictates the syntax for the URN identifier. Whereas a URL specifies the location of a resource, a URN specifies its name independent of the location. URNs are coded by the prefix "`urn:`" and are used to define subspaces of identifiers, called namespaces. For example, the set of URNs of the form `urn:isbn:n-nnn-nnnnn-n` is a URN namespace. ("`isbn`" is a URN namespace identifier).

Unicode. XML is defined as being neutral to the character set used. However, Unicode is used as the standard character set in which XML documents are authored. The Unicode Standard is the universal character encoding scheme for multilingual text. Required in new Internet protocols and implemented in all modern operating systems and computer languages, Unicode is the basis of text exchange and software that must function all around the world. Unicode is promoted by the Unicode Consortium and standardized by the International Standards Organization (http://www.unicode.org). The Unicode standard, Version 4.0, is identical with International Standard ISO/IEC 10646.

Unicode forms

Unicode is modeled on the ASCII character set, but instead of ASCII's limits to upper- and lowercase letters A-Z, Unicode can encode all characters used for all written languages in the world. The Unicode Standard specifies a numeric value (code point), a name, its case, directionality, and alphabetic properties for each of its characters. Unicode consists of three encoding forms, a 32-bit form (UTF-32), a 16-bit form (UTF-16), and an 8-bit form (UTF-8). UTF-8 has been designed for ease of use with existing

ASCII-based systems with code points strongly corresponding to ASCII values.

Unicode coverage

Theoretically, more than one million characters can be encoded. The Unicode Standard, Version 4.0, contains 96,382 characters. The standard includes European alphabetic scripts, Middle Eastern right-to-left scripts, and scripts of Asia (more than 70,000 characters in the unified Han subset), punctuation marks, mathematical symbols, technical symbols, geometric shapes, and dingbats as well as many others. Thus, Unicode character encoding treats alphabetic characters, ideographic characters and symbols equivalently, so that they can easily be mixed in text. Most common characters of the major languages are encoded in the first 65,536 code points, called the Basic Multilingual Plane (BMP).

3.1.3 Extensible Markup Language (XML)

The eXtensible Markup Language (XML) provides the syntactical basis for defining and exchanging semi-structured data and thus forms the foundation on which semantics can be defined. XML has been standardized by the World Wide Web Consortium (W3C). XML is used as a standard way to define the structure of documents that is suitable for automatic processing. "Automatic processing" in this respect refers to *parsing*, i.e. extracting both, the content and the structure of XML documents and checking *well-formedness*, i.e. checking whether an XML document conforms to rules defined by the XML standard, and *validation*, i.e. checking whether and XML document conforms to the document type declaration, e.g., an XML Schema (section 3.1.4).

SGML, HTML, XML

XML is a tag-based markup language for describing tree structures. However, XML cannot be directly applied, but can be seen as a set of syntax rules for creating markup languages in a particular domain. Just like HTML, XML is based on the Standard Generalized Markup Language (SGML), but is limited to a handful of language elements in order to reduce complexity which was seen as a main factor hindering broad acceptance of SGML.

XML language elements. Table 3-1 gives descriptions and examples of the means to structure documents that XML provides (Fensel 2004, 11ff).

Prolog

An XML document consists of the prolog and the instance of the document. The prolog is an obligatory element that states the XML version, the type of the document and other characteristics of the document. It has two components, the XML declaration and the document type declaration.

Table 3-1. XML language elements

element	description	example
tag	A key principle in XML is that markup is separate from content. Thus, XML offers a set of marks, that surround or tag a portion of a document in order to structure the content and attach meta-data or meaning to it.	`<tag>`

element	An XML element is a portion of content that is surrounded by a start tag and an end tag.
	In the case that there is no content for a certain tag, a so-called empty tag is used.
		`<tag>contents</tag>`
		
		`<tag/>` equivalent to `<tag></tag>`

attribute	Pairs of names and values can be attached to the start tag or the empty tag of an element. An element may have more than one attribute.	`<car color=”red” make=”BMW” model=”Z4”>content </car>`
entity	In order to use reserved symbols, such as `<` and `>` in text and avoid that they are interpreted as commands, these characters are replaced by special codes. Another application for entities is abbreviation e.g. for large URLs.	`&lt;` for `<`, `&gt;` for `>` `&Uuml;` for Ü
comment	XML documents may contain comments that are ignored by the processor. They begin with `<!--` and end with `-->`.	`<!-- comment -->`
processing instruction	The otherwise purely declarative elements of XML also contain a procedural element, the processing instruction. Being part of the markup, processing instructions are written in a special tag with the form `<?name pidata ?>`. Processing instructions are ignored by the XML processor, but must be passed through to the application that executes all processing instructions it knows.	`<?xml: stylesheet href=”style.css” ?>`
CDATA	CDATA sections are arbitrary strings that are not interpreted by an XML parser. They can contain further markup that is not interpreted or provide information for other interpreters, e.g., JavaScript code. CDATA sections begin with `<![CDATA[` and end with `]]>`.	`<![CDATA[JavaScript: if (field[x]>0) y=x; ]]>`

XML declaration

The XML declaration has to state the XML version and can contain further elements: `<?xml version="1.0" encoding="UTF-8" standalone="yes"?>`. This example document uses version 1.0 of the XML

standard (attribute `version`), has to be interpreted in UTF-8, a subset of Unicode (attribute `encoding`) and is independent of other documents (attribute `standalone`). Formally, the prolog is a processing instruction.

Document type declaration

The document type declaration sets the rules for the structure of XML documents. It is either part of the prolog or the prolog points to it. The most popular formats for document type declarations are Document Type Definition (DTD) and XML Schema. Compared with DTDs, XML Schema offers the advantage that the schema code itself conforms to the XML standard, it has more options to describe the document, a large number of pre-defined data types and allows for complex, user-defined data structures and data types together with mechanisms for inheritance, primary key attributes and foreign key attributes for reference purposes (section 3.1.4)

Document instance

The document instance contains the elements, attributes, entities, processing instructions, comments and strings (CDATA) of the rest of the document. Every XML document can be mapped to a tree, with each XML element being a node of the tree. Listing 3-1 gives an example of an XML document.

```
<?xml version="1.0" encoding="UTF-8" ?>
<building name="Universitaetsring 3">
   <descr>People at our University office</descr>
   <person id="3470">
      <firstname>Ronald</firstname>
      <lastname>Maier</lastname>
      <year-of-birth>1968</year-of-birth>
      <email>maier@wiwi.uni-halle.de</email>
   </person>
   <person id="3477">
      <firstname>Thomas</firstname>
      <lastname>Haedrich</lastname>
      <year-of-birth>1977</year-of-birth>
      <email>haedrich@wiwi.uni-halle.de</email>
   </person>
   <person id="3472">
      <firstname>Rene</firstname>
      <lastname>Peinl</lastname>
      <year-of-birth>1975</year-of-birth>
      <email>peinl@wiwi.uni-halle.de</email>
   </person>
</building>
```

Listing 3-1. Example XML document

The document contains a prolog with the XML version and a specification of the character set used. The document instance consists of a root ele-

ment, also called document element, <building>. The building's name is an attribute to the element. <building> contains the elements <descr> and <person>. The latter element in turn contains an id as attribute and four elements that comprise the first and last name, email address and the year in which the person was born.

Every XML document can be mapped into a tree (Figure 3-3).

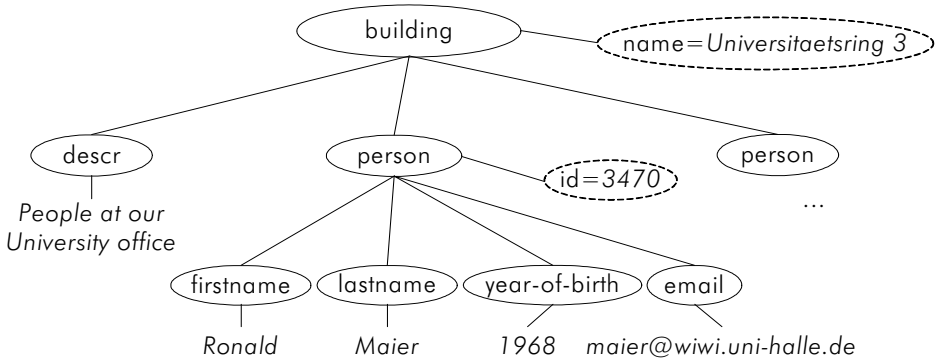

Figure 3-3. Example for XML document represented as tree

Well-formed XML documents

An XML document must comply to the following set of rules for the use of elements, attributes and entities in order to be well-formed:

- there is only one root or document element that contains all other elements,
- opening tags must be matched with corresponding closing tags. Note that XML is case-sensitive. Exception: empty tags draw together opening and closing tag in case there is no content between the two,
- elements must be nested properly, so if element <y> is nested in element <x>, then </y> has to be written before </x>,
- attribute names have to be unique within one element, so an element cannot contain more than one attribute with the same name,
- attribute values must be surrounded by quotes (" or '),
- entities are declared before they are used.

Namespaces. XML markup defines a vocabulary, also called a markup vocabulary, in Figure 3-3 for example a vocabulary to describe persons that work together in a building. Once a markup vocabulary is defined, it can be reused by other XML documents. XML documents in turn might contain elements from multiple markup vocabularies. This could create problems of recognition and collision. Software processing XML documents has to be able to recognize the tags and attributes it should process.

Also, collisions between element or attribute names from different sources might occur.

These considerations require that document constructs have universal names the scope of which extends beyond their containing document. XML namespaces are a mechanism for creating universally unique names for elements and attributes. An XML namespace is a collection of names, identified by a URI reference, which are used in XML documents as element types and attribute names.

Qualified names

The use of namespaces requires that every name in the XML document specifies the namespace in which it is defined, e.g.: `<book:title>`. Thus, names from XML namespaces appear as qualified names which contain a single colon, separating the name into a namespace prefix and a local part. The prefix selects a namespace and is mapped to a URI reference. The prefix, in this case `book`, is an abbreviation for the actual namespace assigned in a namespace declaration. `title` denotes the local part of the qualified name.

Namespace declaration

The namespace prefix must have been declared in a namespace declaration attribute in either the start-tag of the element where the prefix is used or in an ancestor element, i.e. an element in whose content the prefixed markup occurs. The exceptions are the two namespaces `xml` and `xmlns`. The prefix `xml` by definition is bound to the namespace name `http://www.w3.org/XML/1998/namespace`. The prefix `xmlns` is used only for namespace bindings and is not itself bound to any namespace name.

Namespace scoping

The namespace declaration applies to the element where it is specified and to all elements within the content of that element, unless overridden by another namespace declaration, e.g.:

```
<b:book xmlns:b="urn:publicid:books">
   <title>The Adventures of Huckleberry Finn
   </title>
</b:book>
```

In this case, the namespace declaration applies to the element `book` and consequently to the element `title` that is surrounded by the element "`book`". This is also called namespace scoping.

Namespace defaulting

A default namespace applies to the element where it is declared (if that element has no namespace prefix), and to all elements with no prefix within the content of that element. A default namespace is declared by leaving out the prefix in the declaration:

```
<book xmlns="urn:publicid:books"
      xmlns:isbn="urn:ISBN:0-395-36341-6">
   <title>The Adventures of Huckleberry Finn
   </title>
   <isbn:number>0140620648</isbn:number>
</book>
```

In this case, the default namespace for the element `book` is `urn:publicid:books` and the same applies for the un-prefixed element `title` whereas the `number` is taken from the prefixed namespace `isbn`.

3.1.4 XML Schema

We have already discussed the notion of a well-formed XML document which requires the document to conform to a number of syntactic rules. However, as XML is used for semi-structured data, there is also the need to define more constraints for documents. For this reason, a number of schema definition languages have been developed that serve this need, e.g., Document-Type Definition, RELAX NG, Schematron or XML Schema.

Document-Type Definition (DTD)

Document-Type Definition (DTD) was the original schema definition language defined as part of the XML 1.0 standard and has been used to establish a set of constraints for XML documents and to define how XML documents should be constructed. DTDs have been used widely, but recently they are replaced by XML Schema due to the following reasons:

Challenges of DTDs

- *definition in XML*: unlike DTDs, XML schemas are defined in XML and thus can be processed in the same way as XML documents are,
- *rich set of data types*: DTDs lack data types to define allowed values of elements,
- *user-defined data types*: XML Schema provides mechanisms to define new data types not included in the base data types,
- *content model*: XML Schema provides much richer means to define nested elements,
- *namespaces*: DTDs lack support for namespaces.

Purpose of XML Schema

The purpose of a schema is to define a class of XML documents and thus it is not surprising that the term instance document is often used to describe an XML document that conforms to a particular schema (=class). However, neither instances nor schemas need to exist as documents (e.g., they may be stored as fields in a data base record), but when referring to "documents" constructed for a specific schema, most literatures talk about schemas and instances as if they were documents or files respectively.

Local vs. global definition

XML Schemas can be designed in two different ways: with a focus on types or with a focus on the structure of the document. When *types* are focused, all type definitions reside on the same level in the XML tree of the schema file and the structure of the whole XML document that is modeled emerges from the composition of the elements and their complex types. This is called global definition and provides maximum reuse as complex types can be referenced at multiple points in the schema. When *structure* is focused, then the type of each element is directly specified within this element instead of referencing it with the type attribute. This is

called local definition. As reuse of type definitions is not possible when using local definitions, this is only suitable for small schemas.

To explain the various elements of XML Schema, we will start from an example of an XML document instance:

Example of XML document instance

```
<?xml version="1.0" encoding="UTF-8"?>
<building location="Universitaetsring 3">
   <descr>People at our University office</descr>
   <employee id="3477">
      <firstname>Nadine</firstname>
      <year-of-birth>1979</year-of-birth>
      <workhours>19.25</workhours>
      <phone>+49-345-55 23471</phone>
   </employee>
   <employee id="3474">
      <firstname>Florian</firstname>
      <year-of-birth>1977</year-of-birth>
      <workhours>40.0</workhours>
      <email>bayer@wiwi.uni-halle.de</email>
   </employee>
</building>
```

Listing 3-2. Example XML instance document "building"

Elements and subelements

The instance document consists of a main element, `building`, and the subelements `descr` and `employee`. The latter in turn contains other subelements, e.g., `firstname`, `workhours` that contain strings or numbers rather than any subelements. Some elements have attributes, in this case `building` and `employee`.

Complex and simple types

Elements that contain subelements or carry attributes have *complex types*, whereas elements that contain numbers, strings etc., but do not contain any subelements have *simple types*. Attributes always have simple types. The complex types and the simple type `year` are defined in the schema for employees in buildings. The other simple types are built-in simple types defined in XML Schema. These are similar to data types used in data bases. Examples are boolean, integer, decimal, float, double, date, time. Additionally, some data types are specific to XML Schema, such as ID, a unique identifier attribute type specified in XML 1.0, IDREF, a reference to an ID, entity, language or name.

Example of XML Schema

Listing 3-2 shows the corresponding XML schema with global definitions:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
 <xsd:element name="building" type="buildingtype"/>
 <xsd:complexType name="buildingtype">
  <xsd:sequence>
   <xsd:element name="descr" type="xsd:string"/>
   <xsd:element name="employee" type="person"
    minOccurs="1" maxOccurs="unbounded"/>
  </xsd:sequence>
  <xsd:attribute name="location" type="xsd:string" use="optional"/>
 </xsd:complexType>
 <xsd:complexType name="person">
  <xsd:sequence>
   <xsd:element name="firstname" type="xsd:string"/>
   <xsd:element name="year-of-birth" type="year"/>
   <xsd:element name="workhours" type="xsd:decimal"/>
    <xsd:choice>
     <xsd:element name="email" type="xsd:string"/>
     <xsd:element name="phone" type="xsd:string"/>
    </xsd:choice>
  </xsd:sequence>
  <xsd:attribute name="id" type="xsd:short" use="required"/>
 </xsd:complexType>
 <xsd:simpleType name="year">
  <xsd:restriction base="xsd:integer">
   <xsd:minInclusive value="1900"/>
   <xsd:maxInclusive value="2100"/>
  </xsd:restriction>
 </xsd:simpleType>
</xsd:schema>
```

Listing 3-3. Example XML schema "building"

The schema consists of a `schema` element and a number of subelements, particularly `element`, `attribute`, `complexType`, and `simpleType` that constrain the appearance of elements and their content in instance documents.

Prefix xsd: Each of the elements in the schema has a prefix `xsd:` associated with the XML Schema namespace through the declaration in the schema element: `xmlns:xsd="http:..."`. The prefix `xsd:` is used by convention to denote the XML Schema namespace. The prefix `xsd:` is also found in the qualified names of built-in simple types, e.g., `xsd:string` which clarifies that these types belong to the vocabulary of the XML Schema language and not to the vocabulary of the schema author.

Complex types Definitions of complex types typically contain a set of `element` declarations, element references, and `attribute` declarations. For example,

buildingtype is defined as a complex type that contains two element and one attribute declarations. Consequently, elements in instance documents whose type is buildingtype, must consist of two elements, called descr and employee and an contain an optional attribute called location as specified by the values of the name attributes. The elements must appear in the order in which they are declared. The descr element is declared the built-in type string, therefore descr elements in instance documents must be strings. The definition also comprises a complex type person which is also defined in the schema. Finally, the buildingtype definition contains the attribute declaration location which assigns the type string. As attributes cannot contain other elements or attributes, all attribute declarations have simple types. Listing 3-3 is an example for an schema with global definitions.

minOccurs and maxOccurs

The attributes minOccurs and maxOccurs restrict the number of elements allowed. In our case, their has to be at least one employee described as part of a buildingtype and the maximum number is not restricted. The attribute minOccurs can take on values between 0, meaning that the specified element is optional and any positive integer number maxOccurs can take on integer numbers starting from 0 or the term unbounded meaning that the number of elements is not constrained.

Required, optional, prohibited

Attributes may appear once or not at all and can be declared with the use attribute to indicate whether they are required, in our case the id attribute, optional, in our case location, or prohibited meaning that the attribute must not occur.

Deriving simple types

Listing 3-3 declares several elements and attributes that have simple types, some built in to XML Schema, e.g., string and decimal, and the simple type year that is derived from the built-in type integer. Both, built-in simple types and their derivations can be used in all element and attribute declarations. In our case, the derived type year is used in the element definition year-of-birth.

Restrictions

New simple types are defined by deriving them from existing simple types, i.e. restricting the range of values allowed for the new type as a subset of the existing type's range of values. The restriction element is used to indicate the base type which can be a built-in or an already derived type, and to identify the facets that constrain the range of values. In our case, year is restricted by the two facets minInclusive and maxInclusive to values between 1900 and 2100. XML Schema offers a number of facets to restrict allowed values that are quite similar to the options found in high-level programming languages. Examples are length restrictions, patterns, enumerations, and inclusive and exclusive minimal and maximal values.

Content models

The definitions of complex types in Listing 3-3 declare sequences of elements that can or must appear in the instance document. The occurrence of individual elements may be constrained depending on the values of

`minOccurs` and `maxOccurs`. XML Schema also provides constraints that apply to groups of elements appearing in a content model.

Choice, sequence and all

The `choice` group element allows only one of its children to appear in an instance. In Listing 3-3, the complex type `person` contains a choice group with the two elements `email` and `phone`. Hence, in an instance document, a person element must contain either an `email` element or a `phone` element. The choice group and the declarations of the elements `first-name`, `year-of-birth` and `workhours` are children of a `sequence` group. The `sequence` group assures that the elements appear in exactly the declared order. Finally, the `all` group specifies that all elements in the group may appear once or not at all, but they may appear in any order. The `all` group is limited to the top-level of any content model.

Mixed content

All XML Schema mechanisms as used in Listing 3-3 have now been explained. Summing up, XML Schema provides a powerful instrument to define structured data sets and offers even more mechanisms which have not been touched here. One important difference to data definitions as found in data bases is the use of mixed content, i.e. the use of defined XML elements that are mixed with plain text in document instances. To enable character data to appear between the child elements of building, an additional attribute, called `mixed` on the type definition of `building` is set to `true`. Listing 3-4 gives an example instance document with tagged elements conforming to the schema in Listing 3-3 as well as plain text.

```
<?xml version="1.0" encoding="UTF-8"?>
<building location="Universitaetsring 3">
 This document contains a list of <descr>People at our University
 office </descr>. <employee id="3477"><firstname>Nadine</firstname>
 is born in <year-of-birth>1979</year-of-birth>and holds a part-time
 contract with <workhours>19.25</workhours> hours per week.
 Her department phone number is <phone>+49-345-55 23471</phone>, but
 she can also be reached by pager.</employee>
  ...
</building>
```

Listing 3-4. Semi-structured text as example for mixed content

3.1.5 XSL Transformation

To make semi-structured data interchangeable, it is not sufficient that both, the sending and the receiving application support XML. At first glance, there is no difference between two different, vendor-specific XML file formats and two file formats in a proprietary binary format. Both results in the fact that one file can not be read in the other program and vice versa. What makes the difference is that it is costly and cumbersome to write a converter for the binary formats and usually the structure of the format is

not publicly published. For XML data, there is a standardized way to translate one format into the other. The language that supports the conversation is called extensible stylesheet language (XSL).

XSL parts

XSL consists of the two parts transformation (XSLT) and formatting (XSL-FO, formatting objects) and is based on the XPath language that is used to address parts of XML documents. The name of XSL suggests that it fulfills a similar purpose for XML documents as cascading stylesheets (CSS) fulfill for HTML documents. However, CSS can be used for XML documents as well as for HTML documents to specify display parameters whereas XSL is specifically designed for converting XML documents from one structure (or schema specification) to another. We will first discuss XPath, before we have a closer look at XSLT and end with an outlook to XPath version 2 and XQuery, a query language for XML documents similar to SQL for relational data bases. XSL-FO will not be further discussed, as it does not support transformation and integration of documents, but is mainly intended for formatting XML documents for PDF or other print output.

XPath. XPath is used to address elements and attributes in an XML document. The syntax is similar to specifying a path in a file system. The forward slash (`/`) is used as delimiter and also specifies the root node of the XML tree. The XML document listed in Listing 3-5 is used as reference throughout this section to illustrate XPath and XSLT features. The result of an XPath "query" is always a node set containing zero, one or more XML nodes (elements and/or attributes). XPath instructions take the hierarchical tree structure of XML documents into account: This is necessary because a `descr` element as child of `building` can mean something different than the same element as child of `employee`.

Absolute vs. relative paths

Paths can either be absolute (starting with the root node) or relative (starting with the current context node) with absolute paths always starting with a forward slash. The XPath expression `/building/descr` is an absolute path whereas `employee/firstname` denotes a relative path. This relative path is only valid if the `building` node is the current context node for the XPath instruction. It is also possible to address nodes irrespective of their location in the XML tree. Two slashes (`//`) are used to express this. Therefore `//descr` selects all description elements regardless whether they are children of `building` or of `employee`. It is also possible to use wildcards for selecting several nodes with different names. The XPath expression `employee/*` matches all child elements of `employee`. One can further create unions over node sets selected by different XPath instructions with the "`|`" operator. The XPath expression `//firstname | //lastname` selects all firstname and lastname elements anywhere in the document.

```
<?xml version="1.0" encoding="UTF-8"?>
<building id="Universitaetsring 3">
    <descr>Main office building</descr>
    <floor id="0">
       <descr>base floor</descr>
       <employee id="3442">
          <firstname>Ronald</firstname>
          <lastname>Maier</lastname>
          <descr>professor</descr>
       </employee>
       <employee id="3477">
          <firstname>Rene</firstname>
          <lastname>Peinl</lastname>
          <descr>researcher</descr>
       </employee>
    </floor>
    <floor id="1">
       <descr>first floor</descr>
       <employee id="3481">
          <firstname>Thomas</firstname>
          <lastname>Haedrich</lastname>
          <descr>researcher</descr>
       </employee>
    </floor>
</building>
```

Listing 3-5. XML file for XSLT transformation

Formal XPath constituent

Formally, an XPath statement consists of three parts, an axis, a node test and optionally a predicate `axisname::nodetest[predicate]`. An example is `child::employee[lastname='Peinl']`.

The axis is used to specify further context that is used to evaluate the node test. The context determines whether the node should be an element or an attribute and whether it is above, beneath or at the same level in the tree. The predicate is used to restrict the node test. Table 3-2 summarizes available axes and their function and lists some abbreviations that can be used instead of writing the full axis name. Additionally, examples of XPath expressions are given and described in terms of their meaning with respect to Listing 3-5 (starting with node `floor`, `id="0"`). Abbreviations are given in parantheses, e.g., `@` for attribute. The default axis is child.

Table 3-2. XPath axis types and examples

axis	example	description
ancestor	`ancestor::*/descr`	selects the `description` elements of the `building` the `floor` is in
ancestor-or-self	`ancestor-or-self::*/@id`	selects the `id` attributes of the `floor` and the `building`
attribute (`@`)	`attribute::id`	selects the `id` attribute of the `floor`
child	`child::descr`	selects the `descr` element of the `floor`
descendant	`descendant::/@id`	selects the `id` attributes of all employees at this floor
descendant-or-self (`//`)	`descendant-or-self::*/@id`	selects the `id` attributes of the floor and of all employees at this floor
following	`following::*/descr`	selects the `descr` elements of the next `floor` and all employees there
following-sibling	`following-sibling::*/descr`	selects the `descr` elements of all following elements at the same level in the hierarchy
namespace	`namespace::*`	selects all elements defined in the same namespace
parent (`..`)	`parent::*/descr`	selects the `descr` element of the `building`
preceding	`preceding::*/descr`	selects the `descr` elements of all elements before the `floor` element
preceding-sibling	`preceding-sibling::*/descr`	selects the `descr` elements of all preceding elements at the same level in the hierarchy
self	`self::descr`	selects the `descr` element of the current `floor`

A predicate can be used to further restrict the selection to a subset of the nodes selected by the axis and node test. A predicate can be any XPath expression that is valid when being evaluated on the unrestricted result set as the context. Restrictions can be made regarding characteristics of the result set (e.g., `position()=2`) or regarding the value of a child element or attribute (e.g., `firstname='Peinl'`). Even equations with calculations can be used as restrictions (e.g., `@id mod 2 = 0`). In calculations, the usual operators can be used (`+`, `-`, `*`). The division operator is `div` and there is also a modulo operator `mod`. The Boolean operators `and` and `or` can be used as well as the `=` and `!=` (inequality) operators and the relational operators `>`, `<`, `>=` and `<=`. Besides that, there is also a number of

XPath functions that can be used. Table 3-3 gives an overview of the most important functions categorized by function group.

Table 3-3. XPath functions

group	subgroup	functions	description
node set	position	`count()`, `position()`, `last()`	number of nodes, position of the current or the last node in the node set
	identification	`id()`, `name()`, `namespace-uri()`	returns id, name or namespace of the node
string	manipulation	`concat()`, `normalize-space()`, `translate()`	concatenates two strings, eliminates additional white spaces or replaces single characters within the string
	substring	`substring()`, `substring-before()`, `substring-after()`	returns the substring at the given position, the substring before, or the substring after the given string
	string test	`contains()`, `starts-with()`	returns true if the string contains the substring, or starts with it
	other	`string-length()`, `string()`	returns the length of the string or the node content converted to a string
number	round	`round()`, `ceiling()`, `floor()`	rounds a float value correctly, always up, or always down
	other	`number()`, `format-number()`, `sum()`	converts the string to a number (formatted) or sums up several numbers
boolean	core	`true()`, `false()`, `not()`	returns the boolean value `true`, `false` or the negation of the expression
	other	`boolean()`	converts the argument to boolean

XSLT. XSLT builds upon the XPath addressing features to convert XML documents conforming to one schema to XML documents conforming to a second schema. Although XSLT has to be seen as accompanying XML Schema, it neither requires the source nor the destination document to explicitly specify a schema or DTD it conforms to. An XSL transformation document itself is a valid XML document conforming to the XSLT schema. The core element of XSLT is the `template` element. Each XSLT document has to have at least one `template` element. A template matches a type of node in the source document and transforms it. The current context node is set to the matching node when a template is applied, so that relative XPath expressions are evaluated relative to the template match. All transformations have to be within a template. The example stylesheet

shown in Listing 3-6 that converts the document from Listing 3-5 into the document shown in Listing 3-7. We will discuss the example line by line.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
    <xsl:template match="/">
       <employees>
          <xsl:attribute name="total_number">
             <xsl:value-of select="count(//employee)"/>
          </xsl:attribute>
          <xsl:apply-templates select="//employee"/>
       </employees>
    </xsl:template>
    <xsl:template match="employee">
       <person personnel_number="{@id}">
          <xsl:element name="name">
             <xsl:value-of select="lastname"/>
                <xsl:text>, </xsl:text>
             <xsl:value-of select="firstname"/>
          </xsl:element>
          <location>
             <xsl:for-each select="ancestor::*">
                <xsl:value-of select="concat(name(), ': ', @id)"/>
                <xsl:if test="position()!=last()">, </xsl:if>
             </xsl:for-each>
          </location>
       </person>
    </xsl:template>
</xsl:stylesheet>
```

Listing 3-6. XSLT stylesheet

The example uses two templates. The first template matches the root node. This is common as the output document also has to have a root node. It is named `employees` here. The attribute `total_number` is added to the root element. The value of this attribute is calculated using the `count()` function that counts the number of elements in the node set that contains all `employee` elements at any level. The `xsl:apply-templates` element instructs the XSLT engine to process all other templates in the stylesheet. In this case, the `select="//employee"` attribute restricts the node set to which the other templates are applied to the `employee` nodes.

XSLT elements

The second template is applied as it matches the `employee` nodes. For each employee from the input document, a `person` element in the output document is generated. The `person` element also has an attribute. Here, the shorter form of specifying an attribute is chosen that directly states the name of the attribute and its value. Curly braces are used so that the XSLT

engine evaluates the content of the attribute as an XPath expression rather than just putting it in as a string value. Persons have a `name` and a `location` as child elements. The long version of specifying an element is used for `name`. The element `location` is specified using the short alternative. Both versions are equivalent. The value of the `name` element consists of the last and the first name of the employee, connected within an `xsl:text` element. This should be used, if preservation of whitespace is important. Otherwise, the text can also be directly put into the template.

Templates vs. xsl:for-each

The value of the `location` element consists of the `floor` and the `building` the employee works at. As both have child nodes with identical names we can treat them both with one expression. We use an `xsl:for-each` element here that loops over all ancestors of the employee node. Most `xsl:for-each` loops could also be expressed with an additional template and an `apply-templates` element. In contrast to the composition of the `name` element's value we use the `concat()` function here to more compactly create a compound value. It consists of the node name, a string containing a colon and a blank and the value of the node's `id` attribute. Finally, we add a string containing a comma and a blank that separates the `building` part from the `floor` part of the value. This separator should be appended for all cycles of the `for-each` loop except the last one. The `xsl:if` element allows us to specify this. Listing 3-7 shows the result of the transformation.

```
<?xml version="1.0" encoding="UTF-8"?>
<employees total_number="3">
    <person personnel_number="3442">
       <name>Maier, Ronald</name>
       <location>building: Universitaetsring 3, floor: 0</location>
    </person>
    <person personnel_number="3477">
       <name>Peinl, Rene</name>
       <location>building: Universitaetsring 3, floor: 0</location>
    </person>
    <person personnel_number="3481">
       <name>Haedrich, Thomas</name>
       <location>building: Universitaetsring 3, floor: 1</location>
    </person>
</employees>
```

Listing 3-7. Result of the XSL transformation

An exhaustive overview of XSLT with all its capabilities is far beyond the scope of this book. The interested reader is pointed to Kay (2001) or Tidwell (2001) for excellent treatise of the topic. Some other XSLT elements considered useful are summarized in Table 3-4.

Table 3-4. Advanced XSLT elements

elements	description
`xsl:choose, xsl:when, xsl:otherwise`	XSLT version of the select … case or switch instruction known in many programming languages
`xsl:sort`	sorts a node set in the specified way
`xsl:number`	generates a consecutive number
`xsl:decimal-format`	specifies the number format for `format-number()`
`xsl:variable`	creates a variable that contains the node set of an XPath expression; value can only be assigned once, no update possible, referenced with `$variablename`
`xsl:param, xsl:with-param`	parameters can be assigned to templates, similar to parameters for function calls in other programming languages

XSLT engines

The application that applies XSLT stylesheets to XML documents is called XSLT engine, XSLT processor or transformation engine. The most wide-spread ones are the open source product Xerces from the Apache Software Foundation and MSXML from Microsoft. MSXML also includes an XML parser that can validate XML documents against XML Schemas or DTDs. For Xerces, the corresponding parser is Xalan.

XML messaging

XML documents can be exchanged building on synchronous or asynchronous communication measures (HTTP, email, MOM). This is called XML messaging. It has become popular in organizations for the purpose of exchanging data between systems within an organization. A central application receives all messages from the applications in one (often application-specific) XML format, converts it into the XML format of the destination application using XSLT and forwards it to the receiver. During the conversion, it can also do logging or any other monitoring tasks that help making data communication in the organization visible. Example products for XML messaging applications are Microsoft BizTalk Server, WebMethods Business Connector and Seeburger Business Integration Server. Other vendors have integrated their XML messaging products into a more complex solution, e.g., IBM with WebSphere and Bea with WebLogic.

XQuery, XPath 2.0, XSLT 2.0

XSLT provides a standardized way to convert XML documents from one format into the other. The development effort for converters is thereby reduced. The forthcoming update to version 2.0 of XPath and XSLT will further enhance the capabilities of XSLT and integrate the XPath standard with the XQuery specification. XQuery brings SQL-like query capabilities to XML documents and contribute to the integration of XML features into relational data bases. Today, several data base vendors already have products on the market that are capable of translating nearly seamlessly

between XML documents and relational tables and vice versa. Current implementations are all vendor-specific. XQuery is intended to bring some standardization into the market. The basic syntax of XQuery statements is intuitive for any person familiar with SQL. Listing 3-8 shows an example.

```
for       $f in doc("employees.xml")/building/floor
let       $p := doc("salary.xml")//person
where     $f/@id=0 and $f/employee/@id = $p/@id
order by  $f/employee/lastname
return    $f/employee/firstname, $f/employee/lastname,
          $p/salary
```

Listing 3-8. XQuery FLWOR-statement

The example shows how data from two XML documents is merged similar to the join operation in SQL. The file `employees.xml` is assumed to contain the data shown in Listing 3-5, the `salary.xml` file represents salary data in the form

```
<person id="..."><salary>...</salary></person>
```

Two variables, `$f` and `&p` are used to hold node sets from the two files. In the `where` clause, first the node set in `$f` is being restricted to the base floor and then the two node sets are joined by specifying the matching attribute ids. Results are sorted with the `order by` clause and `first-name`, `lastname` and `salary` are returned as results of the query. This type of query is referred to as FLWOR query termed after the starting letters of the single clauses.

In addition to the features shown above, XQuery allows to statically assign XML data to either a variable or the result set. In Listing 3-9, two values are assigned to the variable `types`. These values are wrapped in the additional root element `all_types` as defined in the return clause.

```
let     $types := <type>professor</type>
          <type>researcher</type>
return  <all_types>{$types}</all_types>
```

Listing 3-9. XQuery assignment of XML data to variables

These capabilities make XQuery a powerful query language. However, they are also the reason why it has to consider a large amount of interdependencies and technical hurdles.

3.2 Semantic Integration

From data integration to semantic integration

Knowledge elements can be scattered across a variety of systems. In addition to data integration, semantic integration provides standards and technologies to integrate knowledge elements from different systems on the conceptual level. Thus, it is not data - or Web resources itself alone that is brokered from system to system, but "understanding" of what data means, its semantics and context. Many of these standards and technologies build on XML.

Figure 3-4 shows a four-step model leading from unintegrated to semantically integrated (text) documents.

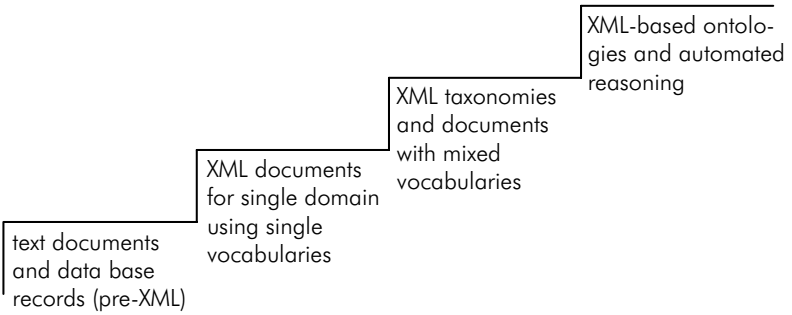

Figure 3-4. From unintegrated text documents to semantic integration (Daconta et al. 2003, 3f)

The first step does not take advantage of XML as the standard for structuring text documents which the second step introduces. The third step relies on the definition of namespaces and XML Schema as a meta-language to define and limit the allowed structure and contents of documents. The fourth and final step heads towards machine-understandable semantics and automated reasoning about documents and is called *semantic integration* in this book.

Section overview

This requires the use of (semantic) meta-data standards for the description of documents or, more generally, Web resources (section 3.2.1) and knowledge modeling, also called the development of ontologies (section 3.2.2).

Knowledge modeling techniques and methods differ with respect to the degree of formality. On the one hand, methods and techniques from the field of artificial intelligence and knowledge-based systems are highly formal and represent knowledge in the form of ontologies or domain models that can be processed by computers. On the other hand, knowledge mapping techniques are less formal and often primarily serve as a tool for human beings to better understand the (highly aggregated) structure of important areas of knowledge or competence and their relationships to,

e.g., persons, groups or other organizational units that create, hold, seek, distribute or apply knowledge (section 4.1.4, 242ff).

Semantic Web stack

Figure 3-5 gives an overview of the Semantic Web stack (Berners-Lee et al. 2001) covering both, data and semantic integration. Based on XML, a number of standards have been developed that provide means to make statements about Web resources and to relate Web resources to each other. We will discuss two standards in this realm, the ISO Topic Maps standard (section 3.2.3) and the W3C standard Resource Description Framework (RDF, section 3.2.4). The Semantic Web stack focusses on RDF and, on the next level, provides a language that supports the design of vocabularies, e.g., classes for instance RDF specifications, called the RDF Vocabulary Description Language or, shortly, RDF Schema (section 3.2.5). Whereas RDF and RDF Schema have been around for some time and can be considered as fairly stable language specifications, there is still a lot of debate going on at the higher levels of the Semantic Web stack. There seems to be some convergence on the level of ontology with the Web Ontology Language (OWL, section 3.2.6). However, standards that allow for specifying entire logic frameworks, exchanging proofs and thus building of trust between agents still remain to be seen. Concepts for these ambitious aims are discussed in section 3.2.7.

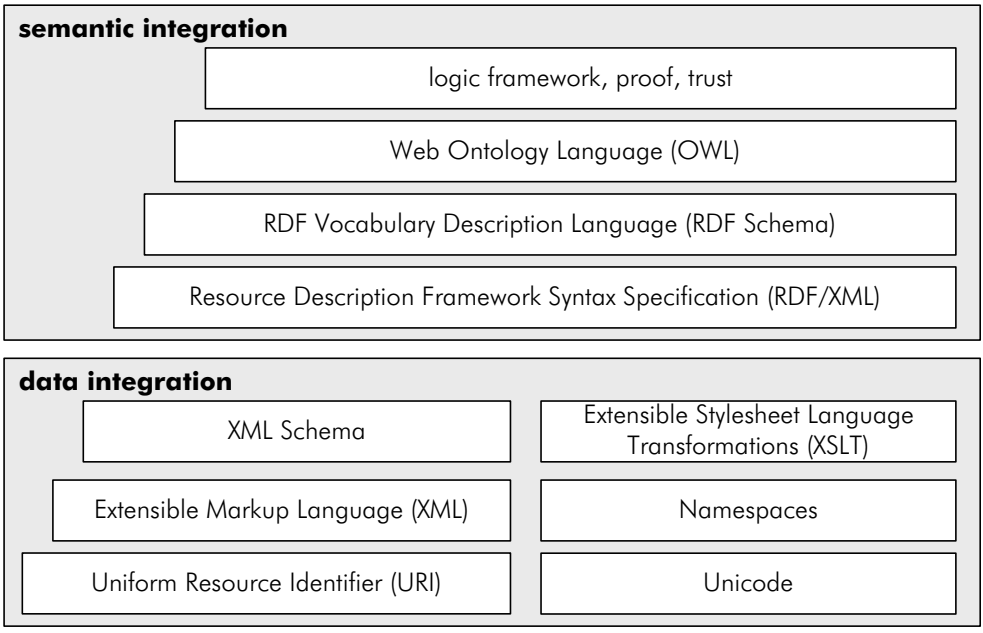

Figure 3-5. Standards of the Semantic Web stack

3.2.1 Meta-data Standards

Meta-data are data about data. A knowledge structure contains knowledge elements and the relations between them as well as meta-data which give further information about their content and associations. One knowledge element's meta-data can simultaneously be another knowledge element's data. It is often only a technical distinction between both.

Use of meta-data

There is a number of reasons to assign meta-data to knowledge elements (cf. Gilliland-Swetland 2002):

- Increased accessibility: Meta-data are a first step to provide meaning about knowledge elements and can be used for smarter information retrieval.
- Retention of context: The context of a knowledge element is crucial for the reconstruction of knowledge by a user. New knowledge elements can only be correctly interpreted and integrated into the personal knowledge base, if the user already has knowledge about the context.
- Versioning: Knowledge elements often exist in multiple versions according to storage format (e.g. rtf, ps, pdf) and content (e.g., an abstract, a research paper, an update of the paper and a book based on that research paper). Meta-data helps to maintain relations between the versions.
- Legal and security issues: Access privileges and copyright information have to be maintained to assure correct handling of knowledge elements.
- System improvement and economics: Meta-data about the usage of knowledge elements can help to improve the system, e.g., by providing shortcuts for often used elements, or to reduce cost, e.g., by automatically transferring little used elements to cheaper storage media.

Kinds of meta-data

Meta-data can be used to describe any kind of data from structured to unstructured. The structure itself already is a form of meta-data and usually provides information about the name of the data element, its data type and its relation to other data elements (e.g., an XML Schema for an XML document). Element names are often not sufficient to carry all relevant information. Additional meta-data is needed that either describes the content (e.g., keywords, domain) or the context of the data especially for semi-structured data. The context can be further subdivided into creation context (e.g., author, creation date, project) and the application context (e.g., customer, intended use). Summing up, we can identify three types of meta-data:

- Content meta-data relates to what the object contains or is about, and is intrinsic to an information object.
- Context meta-data indicates the aspects associated with the object's creation and/or application and is extrinsic to an information object (e.g., who, what, why, where and how aspects).

- Structure meta-data relates to the formal set of associations within or among individual information objects and can be intrinsic or extrinsic.

The structure is *extrinsic* in data base tables (data and structure are separated) and *intrinsic* in XML documents (tags and content mixed). Meta-data can be *informal* (e.g., free text description), *semi-formal* (e.g., structured according to an user-invented structure) or *formal* (e.g., structured and compliant to a standard).

ID3

There are numerous standards for meta-data elements. In the following, some examples are briefly discussed. ID3 is a standard that is used to describe audio data stored in an MP3 file[2]. The elements specified in ID3 version 1 are title, artist, album, year, genre, and comment. ID3 version 2 extends this basic list with a large number of other meta-data elements and allows addition of user-defined elements. Examples are composer, lyrics, language, play counter, popularity, and volume adjustment.

MPEG-7

MPEG-7 is a standard that is used to describe multimedia data, especially data stored in MPEG4 video files (Motion Pictures Expert Group). The MPEG-7 descriptions of content may include meta-data[3]

- describing creation and production processes (e.g., director, title),
- related to usage (e.g., copyright pointers, broadcast schedule),
- about storage features (e.g., storage format, encoding),
- on spatial, temporal or spatio-temporal structure (e.g., scene cuts, segmentation in regions, region motion tracking),
- about technical features (e.g., colors, textures, sound timbres),
- about the portion of reality captured (e.g., actors, objects and events, interactions among objects),
- about how to browse the content in an efficient way (e.g., summaries, variations, spatial and frequency subbands), and
- about interaction of users (e.g., user preferences, usage history).

Dublin Core

An example for a standardization effort primarily aimed at the description of text documents is the Dublin Core Metadata Initative[4]. The standard defines a set of elements that are mainly based on experiences made in public libraries (e.g., Library of Congress, Deutsche Bibliothek; Table 3-5).

---

[2] URL: http://www.id3.org, last access: 2005-02-04
[3] URL: http://www.chiariglione.org/mpeg/standards/mpeg-7/mpeg-7.htm, last access: 2005-02-04
[4] URL: http://www.dublincore.org, last access: 2005-02-04

Table 3-5. Examples of Dublin Core elements

element	description
title	name of the object; could be derived from the filename or from the content
description	abstract or summary of the content in free text form
subject	keywords can be assigned to illustrate topics
creator	entity responsible for authoring the content, e.g., a person, an organization or a service
date	date of an event in the lifecycle of the resource, e.g., creation
relation	links to Web resources (relation.uri) or other sources (relation.other)
language	country code (e.g., us, uk, de) representing the language of the object
rights	e.g., copyright, intellectual property rights, or digital rights (DRMS)
type	categorization, genre or similar aggregation
format	physical or digital manifestation of the object, usually in form of a MIME type

Meta-data standards and RDF

These meta-data standards define a set of meta-data elements that can be used to describe resources and especially documents in a standardized way. However, they also need a standardized language that can express statements such as "book Enterprise Knowledge Infrastructures is_authored_by {Maier, Hädrich, Peinl}". The Resource Description Framework (RDF, section 3.2.4) provides such a language. Consequently, an RDF version of these standards exists, e.g., of the Dublin Core set of meta-data elements.

3.2.2 Ontology

Knowledge modeling aims at a formal description of (documented) organizational knowledge that can be processed by computers, e.g., the Knowledge Interchange Format (KIF), and at a visualization of topics that are of interest in a KM initiative and/or that are supported by contents of an EKI and their relationships. There are relationships (1) between topics and persons, (2) between topics and ICT systems, especially which documents and other resources contain information on a certain topic as well as how they are related to each other and (3) relationships between topics themselves.

Foundation in AI

Several groups of authors have recently extended methods, techniques and tools that were originally developed to model knowledge used in knowledge-based systems. Examples are the CommonKADS method (Schreiber et al. 1999) or the tools OntoEdit and OntoBroker (Staab et al. 2001).

The two terms *ontology* and *taxonomy* are used widely for the results of modeling efforts. Depending on the semantic richness of the constructs

that can be used to formalize topics, knowledge objects and their relationships, some authors distinguish between (simpler) taxonomies and (more powerful) ontologies. In the following, these two terms and their usage with respect to EKI are briefly reviewed.

Taxonomy

The term taxonomy denotes the classification of information entities in the form of a hierarchy, according to the presumed relationships of the real-world entities that they represent (Daconta et al. 2003, 146). A taxonomy can contain definitions and explanations, synonyms, homonyms and antonyms, as in a thesaurus. A taxonomy is often modeled as a hierarchy of terms and can be used as the semantic basis for searching and visualizing a domain, e.g., a collection of documents.

Definition of ontology

Ontologies in KM are formal models of an application domain that help to exchange and share knowledge with the help of ICT systems. "An ontology is a (1) formal, (2) explicit specification of a (3) shared (4) conceptualization" (Gruber 1993, 199).

(1) An ontology has to be *formal* which requires that the ontology is machine-readable. However, there are different degrees of formality of ontologies, from a thesaurus like WordNet[5] to ontologies capturing formal theories for common-sense knowledge like Cyc[6]. (2) *Explicit specification* means that the concepts and relationships as well as constraints on the use of concepts are defined openly and not left to the interpretation of the ontology's users. (3) *Shared* refers to the requirement that the conceptualizations made in an ontology have to be agreed upon by a group of people that intend to use the ontology for knowledge exchange. (4) Finally, *conceptualization* is an abstract model, a representation of a domain or phenomenon which investigates the concepts of that domain or phenomenon that are relevant to the ontology's users.

Elements of ontologies

Ontologies are therefore developed to provide machine-processable semantics of information sources that are accepted by a group of users and facilitate knowledge sharing and reuse. Ontologies are not static, but evolve over time. An ontology not only defines basic terms and relations comprising the vocabulary of a topic area, but also comprises rules for combining terms and relations to define extensions to the vocabulary. Ontologies model (1) objects in domains, (2) relationships among those objects, (3) properties, functions and processes involving the objects and (4) constraints on and rules about objects (Daconta et al. 2003, 190). Thus, ontologies support clear-cut, concise, semantically rich and unambiguous communication between persons aided by EKI and/or between different (parts of an) EKI.

---

5 URL: http://wordnet.princeton.edu, last access: 2005-02-04
6 URL: http://www.cyc.com, last access: 2005-02-04

Taxonomy and ontology compared

Compared to the term taxonomy, the term ontology is usually used not only to describe definitions of terms and basic relationships between terms, e.g., is_a-relationship, but also to support an extended set of relationships, e.g., symmetric, transitive or inverse relationships, and reasoning about concepts that are defined in the ontologies. More specifically, ontologies provide the concept of rule that is used e.g., to check not only syntactic, but also semantic validity of a statement or that is used to derive new relationships between terms from existing ones. Semantic rules, e.g., in the form of inference rules, describe how knowledge can be gained from existing statements. An example is: if two companies operate in the same industry and the same geographic region, then they are competitors (Listing 3-10).

```
FORALL Company1,Company2,Sector1,Region1
is_Competitor(Company1,Company2) <- Company1:Company[
  "operates_in_region"->>Region1:Region;
  "operates_in_sector"->>Sector1:Sector]
  and Company2:Company[
  "operates_in_region"->>Region1:Region;
  "operates_in_sector"->>Sector1:Sector].
```

Listing 3-10. Example for semantic rule in F-Logic

Types of ontologies

The definition of the term ontology is broad enough to cover different types of ontologies that play a number of roles in the development of an EKI (Fensel 2004, 5f):

- *domain ontologies* capture knowledge of a particular type of domain and are thus restricted to the context of this domain,
- *meta-data ontologies* provide a vocabulary used to describe contents in an EKI, e.g., the Dublin Core meta-data standard,
- *common-sense ontologies* capture basic notions and concepts for e.g., time, space, state, event and relationship that are valid across several domains,
- *representational ontologies* comprise definitions of ways to represent knowledge and are not restricted to particular domains, e.g., frame ontology defining concepts such as frame, slot, slot constraint that can be used to explicate knowledge in frames,
- *method and task ontologies* provide concepts specific to particular problem-solving methods, e.g., the concept correct state in a propose-and-revise method ontology, or concepts specific for particular tasks, e.g., the concept hypothesis in a diagnosis task ontology.

Formalization

Ontologies can be formalized with the help of a number of languages, e.g., F-Logic, that are in turn supported by tools, e.g., Ontobroker[7]. However, the term ontology is sometimes used to describe conceptualizations

on a spectrum that extends from weak to strong semantics starting from *taxonomy*, via *thesaurus* and *conceptual model* to *logical theories* that describe semantically rich, complex, consistent and meaningful knowledge (Daconta et al. 2003, 156ff).

Applications in KM

In KM, ontologies facilitate communication, search, storage and representation of knowledge (O'Leary 1998, 58). Methodically, recent ontology modeling methods build upon object-oriented modeling methods that are extended so that they can not only be used to develop software during build-time, but also can be used as an explicit element of the user interface that is used during runtime (Staab 2002, 200).

Most organizations that are about to implement or have implemented a KMS or an EKI have also created at least a minimal taxonomy or ontology (O'Leary 1998, 58). However, the development and continuous maintenance of an ontology requires a substantial amount of effort. Also, ontologies developed individually in organizations are likely to be incompatible and thus cannot be used to share knowledge across organizational boundaries. Consequently, there is a need for standardization, both in the language used to develop an ontology and also with respect to the concepts of ontologies.

As the portion of documents that is stored in XML format and the number of tools that can export and import documents in XML increase, it is not surprising that XML has been proposed as the basis for a description of the content of documents and their relationships, for a standardized taxonomy of topics and ultimately, for comprehensive, machine processable standardized ontologies. Examples for efforts are XML Topic Maps (section 3.2.3), RDF (section 3.2.4), RDF Schema (section 3.2.5) and OWL (section 3.2.6).

3.2.3 Topic Maps

Elements of Topic Maps

One of the most important standards for the description of semantics of documents and Web resources that foster "intelligent" information search and processing is the ISO standard 13250 - Topic Maps that has been established in 1999. The central elements of Topic Maps are topics, associations between topics and occurrences, i.e. resources that are linked to topics. Table 3-6 gives an overview of the elements of Topic Maps.

Standardized by ISO

In December 2000, a specification of Topic Maps in XML was presented by the TopicMaps.Org Authoring Group which was led by the authors of the original ISO standard 13250 about Topic Maps[8]. The XML Topic Maps (XTMs) standard was aimed at porting the ISO standard to XML. The XTM standard introduced a couple of new characteristics and

[7] URL: http://www.ontoprise.de, last access: 2005-02-04

[8] The current version of the XML Topic Maps specification is 1.0, http://www.topicmaps.org, last access: 2005-02-04

renamed a number of element and attribute names as well as the structure of documents[9].

Table 3-6. Elements of Topic Maps

element	description	example
topic	is a representation of an elementary subject, where a subject can be virtually anything from real-life objects to things that only exist in human imagination and abstract concepts. Each topic has an identifier, the *public subject descriptor*. A topic has a name, the *base name*, and can have additional names, e.g., display names, synonyms, abbreviations, etc.	Austria, Vienna, Philharmonic Orchestra, Schönbrunn castle, W. A. Mozart
association	is a relationship between *members* that can reference a topic or a resource. An association has a type, the *association type*, which in turn is a topic.	member: Vienna, association type: is in, member: Austria
role	specifies the part which a topic plays in an association and in turn is a topic. Each topic that is assigned with an association can have an *association role*.	Vienna, role: city, association type: is in, Austria, role: country
occurrence	is a link to any data element relevant to the modeled topic, either as plain text description inline, *resource data*, or as link to an external document or Web-resource, *resource reference*. Each occurrence can have a role associated which it plays with respect to the topic, the *occurrence role* which in turn is a topic.	html-document of Mozart's vita, audio file of the New Year's concert by the Philharmonic Orchestra, map of Austria with Vienna's location
scope	is a domain in which certain assumptions are valid. A topic might have different meanings in different domains, i.e. homonyms. Thus, Topic Maps can contain scopes that define domains in which the topic and its attributes have one and only one meaning.	topic: Vienna, scopes: European cities, songs
facet	is an attribute-value pair that represents a characteristic of a topic, an association or another facet (only in ISO Topic Maps)	topic: W. A. Mozart, facet: year of birth, 1756

Storing XML Topic Maps

Every Topic Map file (.xtm file) is an XML file that contains one single Topic Map. The topicMap element is the root element of the XML document instance. XTMs can also be merged with other XTMs. That creates the possibility for common maps that provide basic topics and associations to be used by all specialized maps.

Published subject indicator

Topics or self-describing resources that are published permanently in the Internet and that can be referenced by others are called *published sub-*

[9] A detailed discussion of the differences between the ISO standard and the XML Topic Maps specification can be found in Widhalm/Mück 2002, 370ff.

ject indicators (PSIs) in the XTM language. There are already a number of collections of topics published by TopicMaps.Org including a base collection of topics called "Psi1" that extends the meta model by a number of model constructs. Examples are sort and display names or association templates for a class-instance or a superclass-subclass association. Besides this core PSI, there are also PSIs for languages and countries that can be downloaded or referenced from the XML Topic Maps Web site[10].

Visualization

The standard only defines the representation in XML and leaves visualization to other standards or application software. However, there are already a couple of tools that provide visualization of XTMs. An example is the Ontopia Knowledge Suite[11]. Visualization of XTM can use the scalable vector graphics (SVG) standards with the help of an XSLT (section 3.1.5, 162ff) processor with the appropriate XSL style sheet that can generate a graphical representation of an XTM on demand.

Example

Listing 3-11 shows the XML document instance for a Topic Map example and is visualized in Figure 3-6.

Figure 3-6. Visualization example of XML Topic Map

Discussion

Topic Maps support structuring knowledge as they provide enough formalism to validate maps and simultaneously allow to represent virtually any knowledge domain with topics, associations and occurrences. Formalism and specification in XML support standardization of the description of knowledge maps and thus ease the exchange of knowledge models between different ICT systems and various organizations. Especially the possibility to merge a domain-specific map with existing published subject indicators seems to be an efficient approach to build ontologies that cover both, standardized and organization-specific parts. Questionable is whether there is already enough restriction, or whether Topic Maps could be further enhanced with more "standard associations" like the existing class-instance or superclass-subclass associations, at the expense of making the approach more complicated.

[10] URL: http://www.topicmaps.org/xtm, last access: 2005-02-04
[11] URL: http://www.ontopia.net, last access: 2005-02-04

```
<?xml version="1.0" encoding="UTF-8"?>
<topicMap xmlns="http://www.topicmaps.org/xtm/1.0/"
 xmlns:xlink="http://www.w3.org/1999/xlink">
 <topic id="km">
  <baseName><baseNameString>Knowledge Management</baseNameString>
  </baseName>
 </topic>
 <topic id="p2p">
  <baseName>
    <baseNameString>Peer-To-Peer Technologies</baseNameString>
  </baseName>
 </topic>
 <topic id="km_individual">
  <baseName>
    <baseNameString>Individual Knowledge Management</baseNameString>
  </baseName>
 </topic>
 <topic id="infotop">
  <baseName><baseNameString>Infotop</baseNameString></baseName>
  <occurrence>
   <instanceOf><topicRef xlink:href="#document"/></instanceOf>
   <resourceRef
    xlink:href="http://www.wiwi.uni-halle.de/maier/Infotop.pdf"/>
  </occurrence>
 </topic>
 <association>
  <member><topicRef xlink:href="#km"/></member>
  <member><topicRef xlink:href="#km_individual"/></member>
 </association>
 <association>
  <member><topicRef xlink:href="#km_individual"/></member>
  <member><topicRef xlink:href="#infotop"/></member>
 </association>
 <association>
  <member><topicRef xlink:href="#p2p"/></member>
  <member><topicRef xlink:href="#infotop"/></member>
 </association>
</topicMap>
```

Listing 3-11. Example for an XML Topic Map

3.2.4 Resource Description Framework (RDF)

As the Semantic Web stack in Figure 3-5 on page 172 shows, there are a number of additional layers on top of XML and XML Schema that have been proposed by the W3C in order to promote the exchange of and reasoning about semantics on the Web. In this and the following sections, we review the higher levels of the Semantic Web stack. The higher the level, however, the more standardization is still in progress. Thus, we will explain the more stable parts, such as RDF, in more detail than work still in progress, such as OWL.

RDF is based on XML

The Resource Description Framework (RDF) is an XML-based language for representing information about resources in the World Wide Web. Semantic information in XML documents is a by-product of defining the structure of a document and thus semantics and structure are interwoven. XML attaches meta-data (tagged information) to parts of a document. RDF provides mechanisms to add semantics to a document as a standalone entity without making any assumptions about its internal structure.

Purpose of RDF

RDF is particularly intended for representing meta-data about Web resources, e.g., author and creation date of a Web page. This is especially useful in case of resources that have a non-textual (binary) format, e.g., images, audio or video files. However, RDF can also be used to represent information about things that can be identified on the Web, even if they cannot be directly retrieved on the Web. Examples are meta-data about items that are described in a Web page, e.g., products in a Web shop, or meta-data about persons, e.g., user preferences or profiles. RDF provides a common framework for expressing meta-data so it can be exchanged between applications without loss of meaning rather than between people. An expected benefit from using a common framework such as RDF is that application designers can profit from the availability of common RDF parsers and processing tools.

RDF statements

RDF is based on the idea of identifying things using URIs (section 3.1.2, 151ff), and making statements about these resources, i.e. describing resources with properties and property values. RDF can represent statements about resources basically in two forms: (1) as a graph representing resources, their properties and values and (2) with the help of an XML-based syntax (called RDF/XML) for recording and exchanging these graphs. The process of denoting an RDF graph in RDF/XML is called *serialization*.

Subject, predicate, object

An RDF statement consists of a triple of subject, predicate and object (Table 3-7).

Table 3-7. RDF elements

RDF elements	description	examples
subject	identifies the thing or resource the statement is about	a Web site, e.g.: URL: http://www.wiwi.uni-halle.de/maier/
predicate	identifies the property or characteristic of the subject that the statement specifies	(1) creator: URI: http://purl.org/dc/elements/1.1/creator (2) date-of-creation: URI: http://purl.org/dc/elements/1.1/date
object	identifies the value of the property	(1) a person, e.g.: "Ronald Maier" identified by: URI:http://www.wiwi.uni-halle.de/maier/team#maier (2) a date: "2002-10-07"

RDF graphs

Statements are represented in RDF graphs that consist of a node for each subject, a node for each object and arcs directed from subject nodes to object nodes to represent predicates. Figure 3-7 gives an example of a simple RDF graph representing the information from Table 3-7. Objects in RDF statements may be either URIrefs, e.g., `http://www.wiwi.uni-halle.de/maier/team#maier`, or constant values, called literals, representing property values, here `2002-10-07`. Literals may not be used as subjects or predicates in RDF statements. Nodes that are URIrefs are drawn as ellipses, nodes that are literals are shown as boxes.

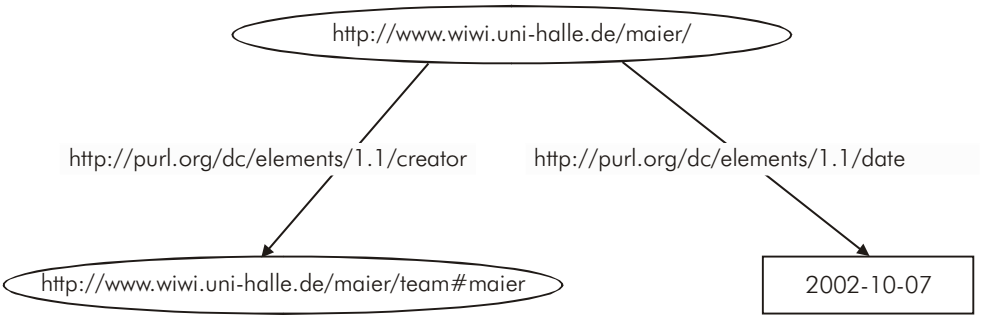

Figure 3-7. Example RDF graph

RDF/XML

RDF also provides an XML-based language to represent RDF graphs: RDF/XML. Listing 3-12 shows the serialization of the RDF graph from Figure 3-7 and will be explained in the following.

```
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
 xmlns:dc="http://purl.org/dc/elements/1.1/">
 <rdf:Description rdf:about="http://www.wiwi.uni-halle.de/maier/">
  <dc:date rdf:datatype=
   "http://www.w3.org/2001/XMLSchema#date">2002-10-07</dc:date>
 </rdf:Description>
 <rdf:Description rdf:about="http://www.wiwi.uni-halle.de/maier/">
  <dc:creator rdf:resource=
   "http://www.wiwi.uni-halle.de/maier/team#maier"/>
 </rdf:Description>
</rdf:RDF>
```

Listing 3-12. Example RDF/XML serialization

rdf:RDF and declaration of namespaces

After the XML declaration, the `rdf:RDF` element indicates that the following XML content, ending with `</rdf:RDF>`, represents RDF code. The `xmlns` declaration specifies that all tags prefixed with `rdf:` are part of the namespace identified by the URIref to the W3C RDF namespace which provides the RDF vocabulary. The next line declares another namespace, prefix `dc:`, this time for the Dublin Core vocabulary.

rdf:Description and about

The two elements `<rdf:Description>` are the RDF/XML equivalent of the two statements shown in Figure 3-7. An RDF statement is considered a *description* which is *about* the subject of the statement, indicated here by the `rdf:about` attribute assigned to the URIref of the subject: `http://www.wiwi.uni-halle.de/maier/`.

Property element

The element `<dc:date>` is the property element, and represents *predicate* and *object* of the statement. The tag contains the qualified name of the predicate, here prefix plus local part put together give the URIref `http://purl.org/dc/elements/1.1/date`. The content of the element represents the object of the statement, the literal `2002-10-07`. The property element is nested within the `rdf:Description` element, indicating that this property applies to the resource specified in the `rdf:about` attribute.

Typed literals

In Listing 3-12, a typed literal is used as property value instead of the plain literal used in Figure 3-7. The data type of the literal is represented by the `rdf:datatype` attribute specifying a datatype URIref to the property element containing the literal.

Short form for multiple properties

Each statement is written as a separate `rdf:Description` element, here the two statements in Figure 3-7 have the same subject and thus the same URIref is written in the `rdf:about` attribute. Statements about the same resource could also be written as sub-elements to one `rdf:Description` element and thus avoid repeating the `rdf:Description` element.

Literals and resources as properties

While the property element in the first statement contains a plain literal as object, the `<dc:creator>` element in the second statement represents a property whose value is another resource. In order to indicate the difference, the `<dc:creator>` element is written using an empty-element tag with no separate end-tag. The property value is written using an `rdf:resource` attribute. Because the URIref is being used as an attribute value, RDF/XML requires the complete URIref as an absolute or relative URIref, rather than abbreviating it with a qualified name as in the element and attribute names.

Abbreviations

As can be seen in Listing 3-12, serialization with RDF/XML results in a substantial amount of code. In order to tackle this, RDF/XML also specifies a number of techniques for abbreviation. The abbreviation of the specification of multiple properties to one resource was already discussed above. Examples for other techniques are:

- *XML entities*: are defined in a `!DOCTYPE` declaration at the beginning of the RDF document for URIrefs used as attribute values, thus shortening the value of `rdf:datatype` in Listing 3-12 to `"&xsd;date"`,
- *relative URIrefs*: an `rdf:ID` attribute is used instead of an `rdf:about` attribute and specifies a fragment identifier which will be interpreted relative to a base URI, either the URI of the RDF document containing the `rdf:ID` attribute or a base URI explicitly stated in an `xml:base` attribute at the beginning of the RDF/XML document.

Reification

So far, the discussed RDF elements provided mechanisms to describe statements about resources. Reification addresses "meta-statements", i.e. statements about statements, e.g., to record information about when statements were made, who made them, or who supports them. RDF provides a built-in vocabulary for describing RDF statements which is called reification. The vocabulary consists of the type `rdf:Statement`, and the properties `rdf:subject`, `rdf:predicate`, and `rdf:object`. RDF reification always involves four statements (the "reification quad").

Reification example

Listing 3-13 shows an original statement about the creation-date of a Web site, one of the two original statements from Listing 3-12, together with the reification quad about that statement. The four statements declare that the resource identified by the URIref `triple4711` is an RDF statement with (1) the subject being the URIref of the Website, (2) the predicate referring to the Dublin Core element `dc:date`, (3) the object of the statement being the typed literal `2002-10-07` and (4) that a person identified by `http://www.wiwi.uni-halle.de/maier/team#maier` asserted that statement.

```
<rdf:Description rdf:about="http://www.wiwi.uni-halle.de/maier/">
 <dc:created rdf:datatype=
  "http://www.w3.org/2001/XMLSchema#date">2002-10-07</dc:created>
</rdf:Description>
<rdf:Statement rdf:about="#triple4711">
 <rdf:subject rdf:resource="http://www.wiwi.uni-halle.de/maier/"/>
 <rdf:predicate rdf:resource=
  "http://purl.org/dc/elements/1.1/created"/>
 <rdf:object rdf:datatype="http://www.w3.org/2001/XMLSchema#date">
    2002-10-07
 </rdf:object>
 <dc:creator rdf:resource=
  "http://www.wiwi.uni-halle.de/maier/team#maier"/>
</rdf:Statement>
```

Listing 3-13. Example of reification

Simplicity versus fidelity of meta-data

It is important to note that asserting the reification is not the same as asserting the original statement, and neither statement implies the other. That is, when we declare that Ronald Maier asserted the creation-date of the Web site was on October, 7th, 2002, this does not necessarily mean that this is the actual creation-date of the Web site. It simply makes a statement about something Ronald Maier asserted. Reification is a powerful concept which matches natural language, but is an unusual concept in data management. Most applications treat data as facts. However, with reification, statements are assertions rather than facts and applications could be required to follow chains of assertions that might even contradict each other. As a consequence, several RDF implementations rule out the use of reification (Daconta 2003, 99). The trade-off is between higher-fidelity and simplicity of meta-data management.

Containers

Groups of resources can be dealt with in RDF with the help of the container vocabulary. A *container* is a resource that contains *members*. The members of a container may be resources or literals. For example, one might want to state that the Web site in Figure 3-7 was created by several authors. A creators container would then contain several persons as members (Listing 3-14).

Types of containers

The RDF container model distinguishes three types: bag, sequence, and alternative:

- `rdf:Bag`: A *bag* is an unordered list of resources or literals, possibly including duplicate members. It is used to declare that a property has unordered multiple values.
- `rdf:Sequence`: A *sequence* is an ordered list of resources or literals, also possibly including duplicate members. Sequence is used to declare that a property has ordered multiple values. The resources are listed so that they can be found easily, e.g., in alphabetical order.

- `rdf:Alt`: An *alternative* is a list of resources or literals that represent alternatives for the (single) value of a property. It might be used e.g., to describe alternative language translations for the title of a book.

```
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
 xmlns:dc="http://purl.org/dc/elements/1.1/">
 <rdf:Description rdf:about="http://www.eki.org/maier/">
  <dc:creator>
   <rdf:Bag>
    <rdf:li rdf:resource=
     "http://www.wiwi.uni-halle.de/maier/team#peinl"/>
    <rdf:li rdf:resource=
     "http://www.wiwi.uni-halle.de/maier/team#sandow"/>
    <rdf:li rdf:resource=
     "http://www.wiwi.uni-halle.de/maier/team#maier"/>
  </rdf:Bag>
  </dc:creator>
 </rdf:Description>
</rdf:RDF>
```

Listing 3-14. Example for RDF Bag container

Membership: `rdf:li`

The container resource denotes the group as a whole. The members of the container can be described by defining a container membership property for each member with the container resource as its subject and the member as its object. These container membership properties have names of the form `rdf:_number`, e.g., `rdf:_1`, `rdf:_2`. RDF/XML provides `rdf:li` in order to avoid having to explicitly number each membership property. `rdf:li` refers to the term "list item" in HTML. Listing 3-14 shows a bag of creators of a Web site specified in the `rdf:Description` element.

Collections

Containers only state that identified resources are members, but they do not define that other members do not exist. RDF collections describe groups containing only the specified members. An RDF collection is a list structure in the RDF graph. The pre-defined collection vocabulary consists of the type `rdf:List`, the properties `rdf:first` and `rdf:rest`, and the resource `rdf:nil`. Each member of a collection is the object of an `rdf:first` property whose subject is a resource that represents a list. For each member of the list, there is another statement that links the rest of the list by an `rdf:rest` property. The list is ended by the resource `rdf:nil` (empty list) as object of the last `rdf:rest` property.

Parsing instructions: `rdf:parseType`

In RDF/XML, the description of a collection can be abbreviated by a property element that has the attribute `rdf:parseType="Collection"`, containing a group of nested elements representing the members of the

collection (Listing 3-15). Generally, the `rdf:parseType` attribute indicates that the contents of an element are to be interpreted in a special way. In this case, the attribute indicates that the nested elements create a list structure. Other examples for attribute values are `rdf:parseType="Resource"` which abbreviates the description of a new (blank) resource or `rdf:parseType="Literal"` which tells the RDF parser that the contents of the element are to be interpreted as an XML fragment.

```
<rdf:Description rdf:about="http://www.wiwi.uni-halle.de/maier/">
 <dc:creator rdf:parseType="Collection">
  <rdf:Description
   rdf:about="http://www.wiwi.uni-halle.de/maier/team#peinl"/>
  <rdf:Description
   rdf:about="http://www.wiwi.uni-halle.de/maier/team#sandow"/>
  <rdf:Description
   rdf:about="http://www.wiwi.uni-halle.de/maier/team#maier"/>
 </dc:creator>
</rdf:Description>
```

Listing 3-15. RDF Fragment showing a collection using `rdf:ParseType`

Semantics of containers

The types of containers, bag, sequence, alternative and collections are pre-defined RDF types, however, there is no special meaning associated with these containers. RDF has no built-in understanding of what a resource of type `rdf:Bag` is. It simply provides types and properties to construct RDF graphs for describing each type of container. In order to assign meaning to certain user-defined classes of RDF resources or literals, an additional language is required, RDF Schema.

3.2.5 RDF Schema

RDF vocabulary description

In addition to the basic techniques for describing resources using RDF statements, RDF users also need a way to describe the vocabularies (terms) they intend to use in those statements. Specifically, vocabularies are needed for describing types of things, properties, and the types of things that can serve as the subjects or objects of statements involving those properties. RDF itself provides no means for defining classes and properties. The basis for describing such vocabularies is the RDF Vocabulary Description Language: RDF Schema.

RDF Schema and object-orientation

RDF Schema provides a type system that was constructed with the type systems of object-oriented programming languages in mind. RDF Schema defines resources as instances of one or more classes which can be organized in hierarchies. For example, a class `Convertible` can be defined as a subclass of `Sportscar` which is a subclass of `Vehicle`. RDF class and property descriptions therefore give additional information about the RDF resources they describe.

RDF Schema terms

RDF Schema consists of a set of pre-defined RDF resources that have URIrefs starting with `http://www.w3.org/2000/01/rdf-schema#` which is normally bound to the prefix `rdfs:`. Consequently, vocabulary descriptions are valid RDF graphs. Table 3-8 gives an overview of the most important terms in RDF Schema.

Table 3-8. RDF Schema language elements

RDF Schema elements	description	example
`rdfs:Class`	A class corresponds to the generic concepts of type or category. RDF classes represent e.g., Web pages, people, documents or abstract concepts.	Vehicle, Car, Convertible
`rdf:type`	A standard property that defines that an RDF subject is an instance of a type defined in RDF schema. The property is also required for each class declaration to state that the class is of type `rdfs:Class`.	class "Convertible" is of type `rdfs:Class`, resource "BMW Z4" is of type "Convertible"
`rdfs: subClassOf`	This element specifies that a class is a specialization of another class. An application supporting the RDF Schema vocabulary can infer that an instance of subClass "Sportscar" is also an instance of "Vehicle".	class "Sportscar" is a specialization of class "Vehicle"
`rdf: Property`	An RDF property defines a characteristic of a class using `rdfs:domain` and the range of values it can represent using `rdfs:range`.	power, numberOfSeats, registeredTo, roofType
`rdfs: domain`	This property defines which class(es) an RDF property applies to. An RDF property may have 0, 1 or more domain properties.	"power" is an RDF property of class "Vehicle"
`rdfs:range`	This property defines the allowed set of values and is used to indicate that the values of a particular RDF property are instances of a designated class.	all integer values; all persons defined in a class "Person"
`rdfs:subPropertyOf`	RDF Schema provides a way to specialize properties as well as classes. This property is used to define that the RDF property in the subject is a specialization of the RDF property in the object.	"grossWeight" is a specialization of "weight"

RDF graph

Figure 3-8 shows an example RDF graph of a schema defined with RDF Schema, a class hierarchy for vehicles. There are five RDF triples in the graph with the subject being the specialized class, the `rdfs:subClassOf` element as the property and the general class as the object of the triple. In the example, trucks, sports cars and passenger cars are specializations of vehicles. Convertibles in turn are specializations of two classes, sports cars and passenger cars.

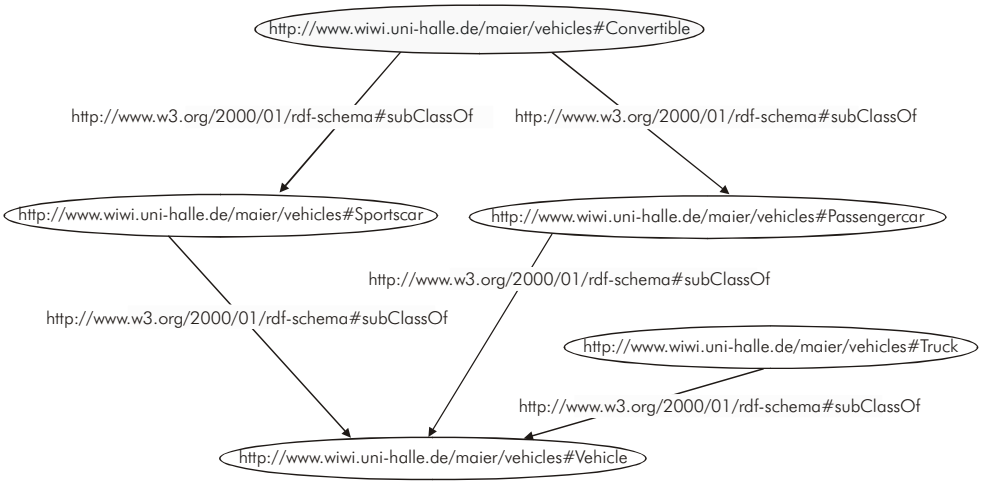

Figure 3-8. RDF Schema example "Vehicles" as graph

Classes and specializations

Listing 3-16 shows the class hierarchy specified in Figure 3-8 in RDF/XML notation using the typed node abbreviation. The URIs for the `rdf` and `rdfs` language definitions are declared and the base URI `http://www.wiwi.uni-halle.de/maier/schemas/vehicles` is explicitly stated in the `xml:base` attribute at the beginning of the RDF/XML document. The individual classes are defined using the `rdfs:Class` element. Specializations are assigned with the `rdfs:subClassOf property`. The class `Convertible` has two `rdfs:subClassOf` properties indicating that the class is a specialization of both, `sportscar` and `passengercar`.

Properties

In Listing 3-16, three properties are defined. The `rdfs:domain` property defines which classes the properties are assigned to. The `rdfs:range` property is used to restrict the allowed values. There can be zero, one or more `rdfs:range` properties to one definition of an `rdf:Property`. For example, the property `registeredTo` is assigned to the class `Vehicle` and RDF statements using this property have instances of class `Person` as objects. In the case of the `power` property, the `rdfs:range` property is used to indicate that the value of this property is a typed literal, in this case integer.

Datatypes

Datatypes are defined externally to RDF and to RDF Schema, and referenced by their URIrefs. The RDF Schema class `rdfs:Datatype` is used to explicitly state that `xsd:integer` is the URIref of a datatype. In the case of the property `roofType`, there is no range property meaning that the values of the property are not restricted.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE rdf:RDF [<!ENTITY xsd
 "http://www.w3.org/2001/XMLSchema#">]>
<rdf:RDF
 xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
 xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
 xml:base="http://www.wiwi.uni-halle.de/maier/schemas/vehicles">
 <rdfs:Class rdf:ID="Vehicle"/>
 <rdfs:Class rdf:ID="Sportscar">
  <rdfs:subClassOf rdf:resource="#Vehicle"/>
 </rdfs:Class>
 <rdfs:Class rdf:ID="Passengercar">
  <rdfs:subClassOf rdf:resource="#Vehicle"/>
 </rdfs:Class>
 <rdfs:Class rdf:ID="Truck">
  <rdfs:subClassOf rdf:resource="#Vehicle"/>
 </rdfs:Class>
 <rdfs:Class rdf:ID="Convertible">
  <rdfs:subClassOf rdf:resource="#Sportscar"/>
  <rdfs:subClassOf rdf:resource="#Passengercar"/>
 </rdfs:Class>
 <rdfs:Class rdf:ID="Person"/>
 <rdf:Property rdf:ID="registeredTo">
  <rdfs:domain rdf:resource="#Vehicle"/>
  <rdfs:range rdf:resource="#Person"/>
 </rdf:Property>
 <rdfs:Datatype rdf:about="&xsd;integer"/>
 <rdf:Property rdf:ID="power">
  <rdfs:domain rdf:resource="#Vehicle"/>
  <rdfs:range rdf:resource="&xsd;integer"/>
 </rdf:Property>
 <rdf:Property rdf:ID="roofType">
  <rdfs:domain rdf:resource="#Convertible"/>
 </rdf:Property>
</rdf:RDF>
```

Listing 3-16. RDF Schema example “Vehicles” in RDF/XML

RDF instance referencing schema

Listing 3-17 gives an example of an instance document that references the RDF Schema example “Vehicles”. The qualified name `vehicles:Convertible` becomes the full URIref `http://www.wiwi.uni-halle.de/maier/schemas/vehicles#Convertible` and thus references the `Convertible` class in the schema in Listing 3-16. The two properties `registeredTo` and `power` reference the corresponding RDF properties defined in the schema. The schema describes the range of the `power` property as `xsd:integer`. Thus, the value of the property in the

instance document should be a typed literal of that datatype. The range declaration does not automatically "assign" a datatype to a plain literal. Therefore, a typed literal of the appropriate datatype must be explicitly provided.

```
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:vehicles=
  "http://www.wiwi.uni-halle.de/maier/schemas/vehicles#"
  xml:base="www.wiwi.uni-halle.de/maier/cars">
 <vehicles:Convertible rdf:ID="BMW Z4">
  <vehicles:registeredTo
   rdf:resource="http://www.wiwi.uni-halle.de/maier/team#peinl"/>
  <vehicles:power rdf:datatype="&xsd;integer">140</vehicles:power>
 </vehicles:Convertible>
</rdf:RDF>
```

Listing 3-17. Example instance of RDF Schema "Vehicles"

Summing up, RDF Schema extends RDF by a vocabulary to define classes and properties similar to object-oriented programming. RDF Schema proposes well-defined rules for writing these definitions so that it is possible to access and exchange descriptions of complex properties and relations of Web resources. The descriptions can be parsed automatically to extract semantic information about Web resources.

3.2.6 Web Ontology Language (OWL)

Purpose

The Web Ontology Language (OWL, sometimes also called Ontology Web Language to conform to the acronym) is a language for defining and instantiating Web ontologies that include descriptions of classes, properties and their instances. The OWL formal semantics specifies how to derive the ontologies' entailments, i.e. facts not literally present in the ontology. Entailments can be based on multiple distributed documents containing ontologies that are combined using defined OWL mechanisms. One advantage of OWL ontologies is seen in the availability of reasoning tools. Tools will provide generic support that is not specific to any particular subject domain, thus easing the task of applying knowledge representation in EKIs to building domain ontologies rather than building reasoning systems.

Revision of DAML+OIL

OWL is a revision of the DARPA (Defense Advanced Research Program) Agent Markup Language-Ontology Inference Layer (DAML+OIL). DAML and OIL are two XML-based languages stemming from a US DARPA program and a European Union program that have recently been

integrated. OWL incorporates lessons learned from the design and application of DAML+OIL and is meant to be its successor.

OWL compared to RDF (Schema)

The normative OWL exchange syntax is RDF/XML, i.e. every OWL document is an RDF document. OWL has been designed for compatibility with RDF and RDF Schema. Compared to RDF Schema, OWL is a vocabulary extension that offers more facilities for expressing meaning, i.e. it adds vocabulary for describing classes and properties, e.g., relations between classes, cardinality, equality, richer typing of properties and characteristics of properties. OWL thus provides richer means to represent machine-interpretable content on the Web.

Ontology as used in OWL

Ontology is understood in OWL as the formal definition and structuring of a set of terms and their relationships in order to describe the terminology used in Web documents, thus representing the semantics of Web resources. Whereas the term ontology traditionally denotes the definitions of classes, properties and relationships defining a certain domain, it is used here in a broadened sense and also includes instances of classes and relationships between instances.

OWL's three sublanguages

Due to the fact that ontologies can be potentially useful in a variety of contexts, OWL provides three increasingly expressive sublanguages that cater to specific communities of users: OWL Lite, OWL DL and OWL Full. Each of these sublanguages extends its simpler predecessor in what can be legally expressed and in what can be validly concluded.

OWL Lite

OWL Lite contains a subset of commonly used features of OWL and limits their use. It provides simple constraints for the definition of classification hierarchies. The main reason for OWL Lite is that due to its lower formal complexity it is simpler to provide tool support for OWL Lite than for OWL DL and especially OWL Full. Also, OWL Lite supports migration for thesauri and other taxonomies.

OWL DL

OWL DL supports all OWL language constructs, but limits their use so that computational completeness (conclusions are guaranteed to be computable) and decidability (computations are guaranteed to finish in finite time) are retained. DL is short for "description logics" and thus refers to OWL's formal foundation.

OWL Full

OWL Full removes many limits that OWL DL imposes and thus retains the syntactic freedom of RDF, however, with no computational guarantees. OWL Full is an extension of RDF, while OWL Lite and OWL DL are extensions of a restricted view of RDF. Consequently, every RDF document is an OWL Full document. The choice between OWL DL and OWL Full mainly depends on the extent to which RDF Schema's meta-modeling facilities are required. OWL Full allows to place restrictions on the vocabulary itself, therefore to augment the meaning of the pre-defined RDF or OWL vocabulary. In OWL Full, reasoning support is less predictable because currently there are no complete OWL Full implementations.

As mentioned above, OWL extends the RDF Schema vocabulary. Table 3-9 gives an overview of the most important language elements introduced by OWL.

Table 3-9. OWL language elements

OWL elements	description	examples
`owl:Class`	Just like RDF classes, OWL classes define groups of individuals that share properties. Classes can be organized in a specialization hierarchy using `rdfs:subClassOf`. The built-in class `Thing` is the class of all individuals and is a superclass of all OWL classes. The built-in class `Nothing` is the class that has no instances and ids a subclass of all OWL classes.	Vehicle, Car, Convertible class "Convertible" is a specialization of class "Vehicle"
`owl:Object Property` `owl:DatatypeProperty`	RDF properties are further detailed according to whether they define binary relations between instances of two classes (object property) or relations between instances of classes and RDF literals or XML Schema datatypes (datatype property). Property hierarchies may be created using `rdfs:subPropertyOf`.	object: owns (Person, Car), powers (Engine, Car) datatype: make (Car, `xsd:string`), year-of-birth (Person, `xsd:integer`)
`<[class] rdf:ID="[instance]"/>`	Individuals are instances of classes. Subclasses represent subsets of the members of a class whereas instances represent individual members. It depends on the intended usage of an ontology whether a certain concept is considered a class or an individual.	"Audi A6" is instance of "Car"; in a different context, "Car" with license plate "HAL-LE 4711" is instance of "Audi A6"
`owl:TransitiveProperty`	If pair (x,y) and pair (y,z) are instances of the transitive property P, then pair (x,z) is also an instance of P.	ancestor (Bob, John) and ancestor (John, Bill) imply ancestor (Bob, Bill)
`owl:SymmetricProperty`	If pair (x,y) is an instance of the symmetric property P, then pair (y,x) is also an instance of P.	partner (Ford, VW) implies partner (VW, Ford)
`owl:inverseOf`	One property may be stated to be the inverse of another property. If property P1 is stated to be the inverse of property P2, then pair P1 (x,y) implies pair P2 (y,x).	registeredTo (Car, Person) implies owns (Person, Car)
`owl:FunctionalProperty` `owl:InverseFunctionalProperty`	Properties may be stated to have a unique value. If a property is a functional or unique property, then it has no more than one value for each individual. FunctionalProperty is shorthand for stating that the property's minimum cardinality is zero and its maximum cardinality is 1. If a property is inverse functional or unambiguous then the inverse of the property is functional.	Functional: madeBy (Car, Manufacturer) InverseFunctional: produces (Manufacturer, Car)

Table 3-9. OWL language elements

OWL elements	description	examples
owl:allValuesFrom owl:someValuesFrom	This restriction requires that for every instance, if any, of a class that is related by the property to a second individual, the second individual is a member of the class indicated by the allValuesFrom clause. owl:someValuesFrom is similar and means that at least one second individual is a member of the class indicated by this clause.	class “Manufacturer”, onProperty “produces”, allValuesFrom “Car”, class “Course”, onProperty “isTaughtBy”, someValuesFrom “Professor”
owl:cardinality owl:minCardinality owl:maxCardinality	The cardinality clause specifies the exact number of elements in a relation. minCardinality and maxCardinality refer to the lower and upper bound respectively. In OWL Lite, only 0 and 1 may be specified, in OWL DL and OWL Full, all positive integer values are permitted.	class “Car”, onProperty “yearLaunched” cardinality(1), onProperty “driver” minCardinality(1)
owl:hasValue	Classes are specified with this clause based on the existence of particular property values. A member of the class is required to have at least one of its property values equal to the hasValue resource.	class “DieselEngine”, onProperty “requiredFuel”, hasValue “Diesel”
owl:equivalentClass owl:equivalentProperty	The equivalentClass restriction indicates that two classes have precisely the same instances. The equivalentProperty restriction states that two properties relate one individual to the same set of other individuals.	class “Automobile”, equivalentClass “Car”; property “lectures”, equivalentProperty “teaches”
owl:disjointWith	This restriction declares two classes to be different. Thus, an OWL reasoning processor detects a conflict if an individual is an instance of two classes declared to be distinct.	class “Motorcycle”, disjointWith class “Car”
owl:sameAs	Two individuals are stated to be the same. OWL does not have a unique name assumption and thus two different names might refer to the same individual. A typical use of sameAs is to declare equality between individuals defined in different documents, as part of unifying two ontologies.	Author “Mark Twain”, sameAs Person “Samuel L. Clemens”
owl:differentFrom owl:AllDifferent owl:distinctMembers	differentFrom declares two individuals to be different, i.e. the opposite of sameAs. A number of individuals, especially a set of individuals, may be stated to be mutually distinct in one AllDifferent statement with distinctMembers declaring all members to be pairwise disjoint.	CarColor “Oceanblue”, differentFrom CarColor “Midnightblue”

Table 3-9. OWL language elements

OWL elements	description	examples
`owl:intersectionOf` `owl:unionOf` `owl:complementOf`	These elements are used to define arbitrary Boolean combinations of classes and restrictions. These constructors specify the members of a set as in a definition.	class "GermanCar", intersectionOf class "Car", onProperty "madeIn", hasValue "Germany"; class "EuropeanCar" unionOf class "AustrianCar", class "BritishCar", etc.
`owl:oneOf`	A class can be specified via a direct enumeration of its members. The members of the class are exactly the set of enumerated individuals.	class "DaysOfWeek", oneOf Day "Monday", Day "Tuesday", etc.

Complex classes

OWL Lite does not support all language elements (e.g., oneOf, disjointWith) whereas OWL DL and OWL Full support all language elements in Table 3-9. Furthermore, OWL Full allows arbitrarily complex class descriptions, consisting of enumerated classes, property restrictions, and Boolean combinations. Also, OWL Full allows classes to be used as instances, so that they can be used as class descriptions in one context and as individuals in another.

Sharing ontologies

OWL is explicitly designed for distribution and sharing of ontologies. The `owl:imports` tag as part of the ontology headers provides a mechanism to include the entire set of assertions provided by another ontology. In order to minimize the effort required to build ontologies, many (standardized) concept definitions might be re-used with the help of this mechanism.

Ontology mapping

OWL provides a number of language elements that specifically target the integration of concepts defined in different ontologies. Examples are on the level of classes and properties `equivalentClass`, `equivalentProperty`, `disjointWith`, `intersectionOf`, `unionOf` and `complementOf` and on the level of individuals `sameAs`, `differentFrom`, `AllDifferent` and `distinctMembers`. This is especially useful when several systems storing parts of the contents held in an EKI have to be brought together, e.g., several document management systems (DMS) that contain documents for specific knowledge-intensive processes. For each of the DMS, an ontology might be developed to capture concept definitions in this specific environment. These concepts then might be mapped with the help of OWL, e.g., to provide sophisticated discovery services. It is a challenge to merge a collection of ontologies that have been designed with different application contexts in mind. The OWL standardization effort of the W3C is seen as encouraging the development of tools for this task and for maintaining consistency between ontologies.

Security issues

OWL's ability to link data from multiple sources described each with an ontology that is mapped to another source's ontology is a powerful feature that can be used in many applications. However, the capability to merge data from multiple sources, combined with the inferential power of OWL, has potential privacy implications. A number of organizations are addressing these issues with a variety of security and preference solutions (e.g., OASIS Security Assertion Markup Language SAML, W3C platform for privacy preferences P3P, see also section 2.3.4, 124ff).

3.2.7 Higher Levels of the Semantic Web Stack

The higher levels of the Semantic Web stack are still under construction. The vision of Tim Berners-Lee, the founding father of the WWW and Semantic Web, is to provide a framework fully capable of exchanging logical rules, proofs as well as digital signatures using encryption and thus turn a web of reason into a web of trust. The most important difference to ontology-based systems which have long been known as expert systems is that this framework is envisioned to be able to handle a lattice of ontologies distributed within and across organizations and thus provide a strong mechanism for semantic integration in an EKI. In the following, we will briefly describe the vision of the layers beyond the ontology layer.

Rules layer

On the *rules layer*, a limited declarative language should be provided that standardizes the way to query RDF statements. A rule language allows inference rules to be given which allow a machine to infer new assertions from existing ones.

Logic layer

A comprehensive *logic framework* is meant to provide a vocabulary to fully describe and exchange logic assertions over the Web. The logical layer should turn a limited declarative language into a Turing-complete logical language, with inference and functions. The vision here is to provide a framework that allows for standardized sophisticated logic processing as specialized logic frameworks do today. Berners-Lee sees this language as being a universal language to unify all data systems just as HTML was a language to unify all human documentation systems.

Proof layer

On the *proof layer*, applications or agents can share logic proofs. One agent can send an assertion together with the inference path to that assertion starting from assumptions acceptable to the receiver. This requires a standardized language and a standard proof engine. Berners-Lee suggests to build this engine on the basis of the logic layer or even on the basis of the less expressive rules layer. He sees a practical need for the exchange of proofs that might not want to wait until a sophisticated logic framework is standardized.

Trust layer

The proof language together with digital signatures signing proofs should turn a web of reason into a *web of trust*. Thus, the logic framework inferences not only on logical assertions, but also on digital signatures.

Future of the Semantic Web

Berners-Lee has been optimistic about the kind of solutions that will be built using Semantic Web standards. If an engine of the future combines a reasoning engine with a search engine, it may be able to get the best of both worlds, and actually be able to construct proofs in a certain number of cases. The vast amount of data and meta-data on the World Wide Web leads to an overwhelming amount of logical processing that would be required to answer arbitrary questions about an unrestricted number of document sources. However, many specific real-life problems might be solved using inference engines that are constrained to a specific set of data, meta-data and logical assertions.

Tool support and commercial use

Whether and when this bold vision will be fully embraced in commercial applications on the Web still remains to be seen. The Semantic Web initiative was for a long time accepted mostly in the artificial intelligence and agent communities, but has made progress to extend the community of users as the standardization process continues. XML, XML Schema and XSLT have gained wide acceptance and as a consequence are supported by a large number of tools. Compared to that, tool supply for creating, managing and using semantic meta-data created with RDF, RDF Schema and OWL is still in its infancy.

3.3 User Integration

Today, most office workers have to access several enterprise systems and commonly they have different login information for each of them. Many company security policies regulate that passwords have to be changed regularly (e.g., every three months). Furthermore, the password should satisfy certain complexity criteria, e.g., it must be at least 7 characters long, contain upper and lower case letters as well as characters and numbers. That leads to the fact that often users write down every single password so that they do not have to remember them, which is not desirable. Additionally, time is wasted for login processes and user data is spread across applications although many user preferences (e.g., colors, fonts) are applicable to any system. User integration denotes efforts to integrate data about users that is managed by different applications. Integration of user data is treated separately from data and semantic integration here, because users as participants play an important role in enterprise knowledge infrastructures. A large number of advanced knowledge services requires integration of data about users across applications in order to e.g., personalize services and contents. We will discuss the basic terms involved in that topic in section 3.3.1 before we describe account management comprising wide-spread techniques for management of user data in section 3.3.2. Identity management as an umbrella term for recently evolving techniques that provide

more advanced possibilities to handle user data is presented in section 3.3.3.

3.3.1 Terms and Definition

Identification, authentication, authorization

In order to gain access to an enterprise system, a user has to have a (user) account to log in to the system. One component of an account is the data that uniquely identifies a user, e.g., a user name, a personnel number or an ID stored on a chip card (identification). The second component is data that is used to verify that the person is really the user he or she pretends to be, e.g., a password, an image of the fingerprint or a certificate stored on a chip card (authentication). The third component is data used to control access to certain functions of the system or data stored in the system (authorization). The latter is called privileges.

Privileges, roles, security groups

A privilege grants access to a set of functions and/or data. Every user could theoretically be assigned a personal set of privileges. As this is time-consuming, usually users are combined to groups and privileges are assigned to those groups instead of single users. Alternatively, privileges can be combined to a set of associated privileges called *role* and users are assigned to one or more of these roles. Although both approaches are different and could be described as top-down and bottom-up, they usually lead to similar results. Today, the role-based approach is often favored because the term role is also used in organizational contexts to describe a set of tasks a user has to fulfill (e.g., project manager or data base administrator). Users often have several roles (e.g. project manager and sales representative for a region) especially with respect to systems where often many roles are defined on a finer level of detail.

User account, master and transaction data

Together with account data, there are two other blocks of user data that are stored in an enterprise system. One block is the user master data and contains e.g., the user's real name, address, telephone number. The other part is data about user preferences and transaction data like past user actions the system uses to customize its look & feel and behavior (section 5.1.2, 316ff). Both together (sometimes even the second block alone) build a user profile.

3.3.2 Account Management

Centralized access to PCs, files and printers

A first approach to centralize the management of user data was mainly targeting user accounts on operating systems level in order to unify access to central resources like printers and files on file servers. Users should also have the possibility to log on to different computers using their centrally managed account. That greatly simplified the management of user accounts as the user's login information did not have to be managed locally on every computer the user accesses, but only once.

LDAP directoy services

The central application that holds data about all user accounts is often an LDAP directory service (section 2.3.1, 109ff). Users and resources are both assigned to a hierarchical structure that is used to partition administration tasks. The hierarchical structure is inherited from the LDAP hierarchy, but is usually structured according to DNS domains. The DNS names have a structure that fits to the LDAP objects (Listing 3-18).

```
peinlpc.wiwi.uni-halle.de
c=de, o=uni-halle, ou=wiwi, computer=peinlpc

peinl@wiwi.uni-halle.de
c=de, o=uni-halle, ou=wiwi, cn=peinl
```

Listing 3-18. Example for LDAP storage of DNS names

Similar to the notation in email addresses a user name is specified using an @ symbol. They hierarchy specifies the object classes `country` (`c`), `organization` (`o`) and `organizational unit` (`ou`). Please note that the object class `computer` is not specified in the LDAP standard, but in a vendor-specific schema extension. Therefore, it can also be called `server` or `machine` depending on the software. Centrally managed accounts are called *domain accounts* (referencing the DNS) in distinction to *local accounts* on a PC.

Security group

Besides the DNS hierarchy that is used to group user accounts and resources, there are also security groups that are independent of domains and can be used to e.g., grant access to a file share to a project team consisting of employees of various organizational units. Employees are assigned to the same security group and access is managed based on the group instead of single users. In some directory service implementation there are a number of different group types that fulfill different purposes and can be nested. Windows 2000 for example distinguishes the group types security group and distribution group (for email messaging) and both types can be either domain local (users from any domain can access resources from one domain), domain global (users from one domain can access resources in any domain) or universal (users from any domain access resources from any domain).

Access control list

The permissions to access a specific resource are finally managed in so-called access control lists (ACL). An ACL consists of one or more entries that map user accounts or security groups to permissions. Possible permissions are: list folder content (folder only), read, execute, write (create new files), change (override existing files) and full access. Each permission can be explicitly granted or denied, otherwise it is unspecified. Access is only granted if explicitly stated in the ACL, otherwise denied. The reason for this is that permissions can be inherited from objects higher in the hierarchy (folders or domains) which can lead to conflicts due to users belong-

ing to multiple groups with different and potentially contrary privileges. Denying permissions always overrides granting permissions. The inheritance of permissions together with the group memberships makes evaluating a concrete user request to read or write a file a complex task. So there is a tradeoff between security and ease of administration on the one hand and fast access and lightweight implementations on the other hand.

Example implementations for account management are Windows Active Directory or OpenLDAP together with a Samba server that provides similar functionality.

3.3.3 Identity Management

Account management is well established in medium-sized and large companies, but it represents only the first step to user integration since focuses access to operating system and network drives. However, access to enterprise applications should also be centrally managed. The term identity management was coined for a holistic approach to do that.

Definition of digital identity

A digital identity is the set of attributes that uniquely identifies a physical user throughout all applications in the network.

A user can have many different accounts that are used to access various applications in the network. Applications can identify these accounts by different attributes of the user, e.g., email address, personnel number or user name.

Components of identity management

Figure 3-9 shows the components that form an identity management solution. The components can be grouped into layers that reach from the storage of user data to the concrete services the user gets from the solution.

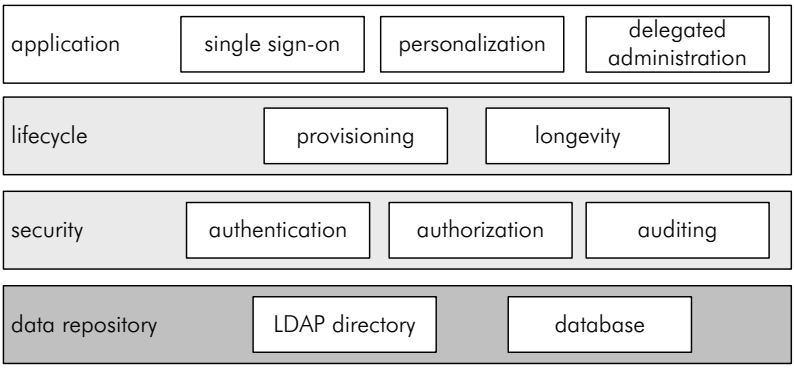

Figure 3-9. Components of an identity management solution (De Clercq, Rouault 2004)

Data repository layer

Like account management, identity management solutions usually build on LDAP directories to store the user data, but data base management systems could be used as well (section 2.3.1, 109ff).

Security layer

On top of the data repository, the security layer provides methods for authentication and authorization (section 2.3.4, 124ff). Furthermore an auditing component logs all changes and thus makes them traceable. Besides different protocols (e.g., Kerberos, CHAP, EAP) and authentication methods (e.g. user name and password, biometric, smartcards), the important aspect for identity management is that not all installed enterprise applications provide LDAP authentication. Therefore, wrapper applications have to be developed which are a kind of middleware and translate between central and application-specific user accounts, so that central authentication is sufficient for accessing the application.

Life-cycle layer

The life-cycle layer consists of a provisioning and a longevity component. Provisioning is the automation of procedures and tools to manage the life-cycle of an identity: (1) creating an identifier for the identity; (2) linking it to authentication providers; (3) setting and changing attributes and privileges; and (4) decommissioning the identity (De Clercq, Rouault 2004). Propagation of data from a central repository to applications that do not support central user management is crucial and has to be developed on a case by case basis. Longevity tools create historical records of identities. These tools allow examination of the evolution of identities over time. Longevity is linked to the concept of attestation or the ability to attest what actors had access to what resources in what timeframe, irrespective of whether they actually exercised their access rights, which is a matter of auditing.

Application layer

Life-cycle and security layer together provide the core components of identity management in a narrow sense. The components on the application layer bring the benefits of identity management to the user. Based on the core components, users get single sign-on functionality, so that they only have to log in once to get access to all applications they need. It is further possible to personalize applications based on one consistent set of user data (section 5.1.2, 316ff).

Delegated administration, self service

Delegated administration is a means to decentralize administrative tasks by giving administration privileges for defined parts of the infrastructure, e.g., a user group or a server, to a number of employees. Self service goes one step further and allows end-users to change their own password, address information and other person-related data on their own. The additional benefit is that the central management overhead is kept small and there is no danger of bottlenecks for changing user data or other administrative tasks.

Personalization

Identity management targets not only account data, but also other user-specific data that resides in enterprise applications. The usage of user profiles and data about user actions across applications can significantly

improve the perceived usefulness of system functions and contents if systematically used for personalization.

Product examples

Software vendors that provide identity management solutions include Computer Associates, IBM, Novell, Oracle and Sun. The solutions consist of multiple parts. For example, Oracle bundles Oracle Internet Directory (an LDAP directory service) together with Certificate Authority, Application Server Single Sign-On, Directory Integration, Directory Provisioning Integration Service, and Delegated Administration Services. Sun offers Sun Java System (SJS) Identity Manager, SJS Access Manager and Directory Server Enterprise Edition as part of its identity management solution.

Identity management and Internet

There are even efforts to expand identity management to the Internet. Microsoft has taken a first step with its .NET Passport initiative. A .NET passport can be used to get access to a number of Web sites from Microsoft network (MSN), hotmail and developer network (MSDN) to several other e-commerce sites. Many people fear that a monopoly for the management of digital identities could harm privacy. Therefore, several major software vendors have founded the Liberty Alliance under lead of Sun Microsystems that tries to establish its own identity management solution on the Internet based on a federated infrastructure (several servers with trust relationships between them) instead of the centralized approach of Microsoft.

3.4 Function and Process Integration

In contrast to data, semantic and user integration that builds on data and meta-data, there is another fundamental pillar of integration: functional integration and process integration that builds upon the functions, brings them into a meaningful order and defines preconditions for their start. Functions are the atomic unit for interaction between information systems.

3.4.1 Function Integration

Function and similar terms

There are several synonyms in use that express the different views on functions that different programming paradigms impose. The term *function* itself has roots in mathematics and has a strong focus on the result that is returned.

Procedure

In the procedural paradigm the term *procedure* is used that focuses more on processing and was separated from functions that had to return a value. However, many programming languages used functions that return `void`, an empty value without a real data type, instead of procedures.

Method

In object-orientation the term *method* is used for both, functions and procedures. Objects encapsulate data and methods that operate on that

data. The term method refers to the freedom of choice that the object has, whether it performs a requested operation or not (e.g., because of illegal parameters). Therefore, people sometimes speak of "sending a message to an object" instead of "invoking a method". The agent-oriented programming paradigm further strengthens this freedom of choice (section 1.5.1, 49ff).

Service

Another term with a meaning similar to function that is popular today is the term *service*. It is especially used in the context of Web services and service oriented architectures (SOA) and denotes that programs, objects or modules perform something for the user of that service. Usually, service was only used in conjunction with people, but with SOA, it is programs that use services.

Application Programming Interface, DLL

From an integration perspective, only those functions are of interest that are explicitly made publicly available and thus can be called by other programs. The collection of all accessible functions is referred to as *Application Programming Interface* (API). An API consists of specification implementation and documentation. The *specification* lists all functions and their parameters. The *implementation* usually is provided in form of a dynamic link library (DLL) that encapsulates program code and can directly be used by other programs. The term DLL is mainly used in Windows environments whereas in Unix environments the term shared library (.so, shared object) is common. The *documentation* comprises a description of the function's purpose and intended effects, what parameter values are allowed and required and what return values can be expected. Good documentations also contain information about possible error cases. Besides integration, APIs are also commonly used for generating layers of abstraction that can be used to build upon. The operating system is a good example for that. It exposes APIs for drawing common screen elements like windows and buttons, file handling and communication. Examples for APIs used to integrate standard software are SAP R/3 BAPI in the ERP domain and Opentext Livelink API in the KM domain.

Remote Procedure Call. Remote function calls are required in an integration scenario, as enterprise systems usually run on a dedicated server each. The term remote procedure call (RPC) is used for any kind of function call to software on remote systems. The idea behind RPC is to make a remote procedure call look as much as possible like a local one for a programmer. The goal is to simplify programming by relieving application developers from the need to open a socket, establish a network connection, pack and unpack parameters into a stream every time they access a remote system. This is performed by so-called *stubs* and the communication protocol used is often UDP. The client (calling program) and the server (called function) do not communicate directly with each other but instead have a stub each in their own local address space that is simulating the remote object.

Figure 3-10 illustrates the functioning of an RPC. Firstly, the client is calling the client stub. This call is a local procedure call, with the parameters pushed onto the stack in the normal way. In step 2 the client stub packs the parameters into a message and maks a system call to send the message. Packing the parameters is called *marshaling*. In step 3 the kernel sends the message from the client machine to the server machine. In step 4 the kernel passes the incoming packet to the server stub. Finally, in step 5 the server stub calls the server procedure with the unmarshaled parameters. The reply traces the same path back in the other direction.

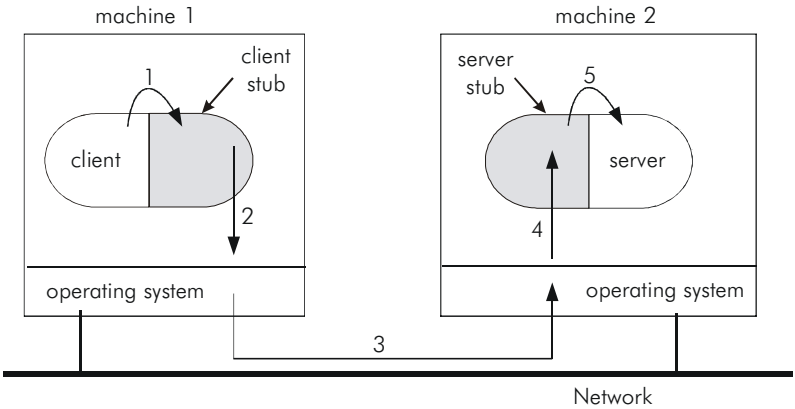

Figure 3-10. Principle of a remote procedure call (Tanenbaum 2003, 528)

Limitations of RPCs

There are a couple of problems that prevent RPCs from behaving exactly like local calls. Passing parameters by reference certainly does not work, as client and server operate in different address spaces and therefore cannot share a data structure. The same reason makes it impossible to pass pointers to complex data structures like lists or arrays as the client stub has problems in determining where the structure ends and therefore cannot transfer it to the server. The client stub may also have problems in determining the data type of the parameters for weakly typed functions with varying numbers and types of parameters.

Java RMI

Popular technologies for RPC on certain systems are Java RMI (Remote Method Invocation) and Windows DCOM (section 2.4.2, 139ff). As an example, we will take a closer look at Java RMI. In order to develop a Java program that uses RMI, the programmer has to design a number of classes and one interface that defines the methods that can be called remotely. The server part usually consists of two classes. One class implements the service interface and therefore provides the functionality requested by the client. In addition to that it implements the `Remote` interface to be capable of receiving RMI calls. The other class loads instances of the implementation and lets them listen for network connections. On the

client, the calling class is needed and the client stub that is generated automatically by the `RMIC` tool that ships with the Java SDK (Software Development Kit). The client uses JNDI (section 2.4.2, 139ff) to locate the remote object. Listing 3-19 shows an example of how a Java RPC is implemented.

```
// Server
RemoteClassImpl myImpl = new RemoteClassImpl();
Naming.bind("remote_class", myImpl);

// Client
RemoteClass rc = (RemoteClass)
      Naming.lookup("rmi://remoteserver/remote_class");
```

Listing 3-19. RMI code example

Object lookup

`RemoteClassImpl` is the class on the server that implements the functionality the client needs. An object of this class is instantiated and it is being bound to a unique name. The client uses the interface `RemoteClass` to get a reference to the implementation on the server (`RemoteClassImpl`). The lookup is made using JNDI, but instead of an LDAP directory service, a tool called `rmiregistry` is responsible for the answer. It manages the names and bindings and has a function similar to the ORB in CORBA (section 2.4.2, 139ff). It can even use CORBA IIOP instead of the Java remote method protocol (JRMP) to connect client and server stub. The default port for RMI communication is 1099. Besides instantiation of the object and additional `Remote Exceptions` that can occur each time a method of the remote object is called, the handling is identical to that of local objects. RMI even overcomes some of the RPC limitations listed above.

Serialization

Complex data structures in the form of objects can be passed as parameters to remote methods if they are serializable (i.e. implementing the `Serializable` interface of the Java API). Serialization in a nutshell means making a *deep copy* (all attributes with basic data types and copies of all objects referenced in attributes) of the object and creating a byte stream out of it. There also has to be the reverse mechanism that reads a byte stream and subsequently fills all the attributes with values.

Web services. However, RMI and similar standards still have a limitations: they are bound to a specific platform. CORBA is one approach to solve this problem and provide language- and platform-independent RPCs, but due to complexity and initial performance problems, CORBA has not been adopted as quickly and widely as expected.

XML over HTTP

Nowadays, a different approach gets more attention. Web services use XML data that is sent over HTTP connections to call remote functions.

Two XML standards have been defined for Web services: WSDL (Web service Description Language) and SOAP (Simple Object Access Protocol). WSDL is used to describe the function, their parameters and the result and therefore is similar to an interface in Java or C# (Listing 3-20).

```
<definitions>
    <types> … </types>
    <message> … </message>
    <portType> <operation>… </operation>…<portType>
    <binding> … </binding>
    <service> … </service>
</definitions>
```

Listing 3-20. Structure of a WSDL document

WSDL

A WSDL file consists of five sections that define the types used, the messages, the portTypes, binding and service. The `types` section contains the definitions of data types that the Web service uses as input or output parameters. The XML Schema definitions (xsd:string, xsd:float, …) are used for the basic types. Complex types are specified using the XML Schema language. The `message` section defines name, input and output of functions that are involved. It can be seen as an API specification. The `portTypes` define so-called `operations` that use the messages as input or output. The `operations` distinguish four types that are listed in Table 3-10. The `binding` section maps the abstract definitions of the `operations` to concrete protocols, mainly SOAP and HTTP. The `service` section finally provides the URL where the Web service can be found.

Table 3-10. Types of WSDL operations

	waiting for response	**without response**
sending	solicit-response	notification
receiving	request-response	one-way

SOAP

SOAP is used to wrap function calls including parameters into an envelope so that they can be sent over HTTP connections. The SOAP message consists of a header and a body section. As the contents of both sections are specific to the specific Web service, only a few things are specified. The elements in the header can use the three attributes `encoding`, `actor` and `mustUnderstand` to specify the encoding standard, which endpoint is addressed with the message (the final recipient or a station on the way) and whether the specified elements contain information that have to be understood by the receiver or that are optional. The body can contain a fault section if an error occurs. It describes the cause of the fault in a

machine-readable (`faultcode`) and a human-readable format (`faultstring`). The `faultactor` element holds information about which part caused the error.

Web services are sometimes also called XML-RPC, but one has to be careful as this term also denotes a Java technology that uses proprietary XML envelopes instead of SOAP to send data over HTTP connections. Java XML-RPC is still used sometimes, because it is much simpler than SOAP, although SOAP is the standard and is needed for interoperability with other Web services. Prominent examples for companies that offer services on the Internet using WSDL and SOAP are Google and Amazon.

UDDI

Web services can either be directly invoked by specifying the URL referencing their implementation, or they can be looked up in a registry, similar to CORBA or RMI registries. The standard for Web service registries is UDDI (Universal Description, Discovery and Integration). UDDI is XML-based like SOAP and WSDL and is regarded as the third pillar of Web services, although its use is optional. Originally developed by IBM, Ariba and Microsoft, it was handed over to the OASIS standards organization after the release of version 3 of the specification in July 2002. The goals of UDDI are on the one hand dynamic lookup and binding of Web services during runtime and on the other hand supporting developers in finding information about services, so that they know how to design clients that use these services. Originally, UDDI was meant to be a Universal Business Registry (UBR) on the Web where everybody could list and query Web services for free. However, more and more private UDDI registries have been set up by organizations that list only Web services within their boundaries and the development of the UDDI specification has reflected this evolution of purpose.

White, yellow and green pages

The information in a UDDI registry can be categorized in analogy to a phone directory. White pages list organizations with contact information and the services the organization provides. Yellow pages list Web services classified according to a taxonomy. Green pages finally hold detailed information how a Web service can be invoked together with a link to the WSDL file.

UDDI entity types

There are four main entity types in UDDI that are the basis for the descriptions. A *businessEntity* holds information about the provider of a Web service including name, address and contact information. A *businessService* describes a group of Web services that usually differ only in technical aspects like the URL at which the service is provided or different protocol bindings. A *bindingTemplate* holds technical information about how a service can be used, essentially the URL and parameters. The *tModel* finally is a container for any kind of specification like a WSDL file, a classification or a protocol.

WS-Addressing

In addition to these three basic Web service standards, there is a number of other standards under development or that have recently been adopted. These standards extend or modify the Web service specification to address

lacking mechanisms or bad performance. WS-Addressing extends the addressing capabilities of Web services to consider not only the target address, but also reference properties for routing of SOAP envelopes. These properties can specify additional relevant information, like a client id or session information. Reference properties store similar information as cookies do for common Web applications.

WS-Security

More security was one of the most often demanded features for Web services. WS-Security introduces a new block within the header of a SOAP envelope that can hold authentication information (user name and password or a certificate), signatures or encryption information. It also specifies how this block has to be processed and builds on XML standards like XML-Security and XML-Signature.

WS-Policy

The WS-Policy framework allows specifying requirements for Web services that a client or server must fulfill so that both can communicate. A single requirement within such a policy is called assertion. Examples for assertions are authentication methods, transport protocols, or availability (e.g., 24x7).

WS-Attachments

The WS-Attachments proposal and the "SOAP with Attachments" addendum to the SOAP specification describe how SOAP envelopes can be transmitted together with related documents like images (e.g., a scanned document) or technical drawings (e.g., a CAD file). DIME (Direct Internet Message Encapsulation) and MIME (Multipurpose Internet Mail Extensions) are used for encapsulating the SOAP envelope and other documents. Other standards that build upon Web services are discussed in section 3.4.3.

Other integration technologies. Although Web services recently became the most popular approach to function integration, there are some other mechanisms worth mentioning. First of all, remote procedure calls do not have to be synchronous. Requests can also be put in a queue (section 2.3.3, 123ff) and processed asynchronously by the server according to the "publish and subscribe" model. But there are also some other forms of function integration. (Alonso et al. 2004, 33f)

TP monitors

TP monitors are the oldest and best-known form of middleware. They are also the most reliable, best tested, and most stable technology in the enterprise application integration arena. Put simply, TP monitors can be seen as RPC with transactional capabilities. Depending on whether they are implemented as 2-tier or 3-tier systems, TP monitors are classified into TP-lite and TP-heavy monitors. TP-lite monitors typically provide an RPC interface to data base, whereas TP-heavy monitors provide more functionality and can be applied in several cases.

Object broker

RPC was designed and developed at a time when the predominant programming languages were imperative languages. When object-oriented languages took over, platforms were developed to support the invocation of remote objects, thereby leading to object brokers. These platforms were

more advanced in their specification than most RPC systems, but they were not substantially different in terms of implementation. Many object brokers use RPC as the underlying mechanism to implement remote object calls. The most popular class of object brokers is based on CORBA.

Object monitor

Object brokers were designed to specify and standardize the functionality of middleware platforms. It soon became apparent that much of this functionality had already been available from TP monitors. At the same time, TP monitors, initially developed for procedural languages, had to be extended to cope with object-oriented languages. The result of these two trends was a convergence between TP monitors and object brokers that resulted in hybrid systems called object monitors. Object monitors are TP monitors extended with object-oriented interfaces.

3.4.2 Process Integration

Like semantic integration builds on data integration, processes build on isolated functions and connect them in a shared context to reproduce and support business processes in an electronic form. In the following, we first define the terms process and workflow and study types of workflows suitable for EKI before we discuss workflow management systems, a system class that can be used to define and execute workflows. Then, some of the standards are presented that build on top of Web services and coordinate and orchestrate their execution. Software implementations of these standards fundamentally change the way in which process management works in organizations with a move to service-oriented architectures (section 1.5.2, 59ff). We will conclude with an outlook towards semantic Web services that promise to join semantic, function and process integration to enable even more automation.

Definition of business process

A business process is a closed, successional and repeating sequence of tasks which follows certain rules and is necessary to fulfill a business function.

Process support for knowledge work

Business process and workflow management is important for organizations but focuse that part of an enterprise infrastructure that is not considered in this book (section 1.5, 47ff). Knowledge work is characterized by unpredictable task sequences and complex tasks that cannot be described in sufficient detail with conventional process languages. Nevertheless, there is still a need for process integration in EKIs. Some parts within complex knowledge-intensive tasks can be automated or at least supported by workflows, e.g., the publication of a document to readers interested in the domain to which the document belongs which involves a knowledge broker who is responsible for checking the quality of the document and

proper assignment to the company's ontology. Table 3-11 characterizes three types of tasks.

Table 3-11. Types of processes in office work

	problem	**information need**	**cooperation partner**	**solution**
individual process	high complexity, low predictability	undefined	changing, not defined	open
case-based process	medium complexity, medium predictability	problem-dependent, partly defined	changing, defined	partly regulated
routine process	low complexity, high predictability	defined	invariant, defined	defined

Workflow Management. Routine tasks are best suited for workflow support but are seldom for knowledge work. Case based tasks occur more often in knowledge work and can be supported by ad-hoc workflows. Finally individual cases do also play an important role in knowledge work and can not be supported by workflows. Different concepts are needed here (section 6.1, 352ff)

Definition of workflow

Workflows automate business processes, in whole or part. They pass documents, information or tasks for action from one participant to another, according to a set of procedural rules. Participants can be humans or machines.

Views on workflow

There are several views on workflows that stress different aspects: flow of tasks, workflow objects or types of communication acts. Workflows can be seen as:

- *structured sequence of tasks*: workflows define events, conditions and actions which determine the sequence of tasks,
- *migrating objects*: workflows can be understood as subsequent changes to the status of an object, e.g., a document that can be created, amended, reviewed, corrected, published and archived, or
- *updated conversational types*: workflows can be seen as conversation nets where actors utter speech acts and other actors respond, e.g., an employee wishes to order a new computer and the department head rejects the purchase request.

Level of formality

Alternatively, worflows can be classified according to their structure:

- Highly structured workflows are called *standard workflows*. Exceptions to these standard workflows that occur seldom but more than once can also be subsumed in this category.

- *Semi-structured workflows* are workflows that partly consist of team-based tasks or a number of tasks that are loosely coupled and do not always occur in the same sequence.
- *Collaborative workflows* are either completely team-based or involve ad-hoc planning of interaction between individuals.

Ad-hoc workflows

Furthermore, workflows can be used to roughly structure a complex process into several weakly structured and thus still complex tasks so that employees performing the tasks can check whether all relevant parts of the process have been fulfilled (e.g., systematic evaluation of a product). One-time workflows are called ad-hoc workflows. In some cases, it is also reasonable to provide organization-wide templates with coarse granularity, e.g., for the proposal creation process of complex goods or services. Although most single tasks within this process would require further explanation and require a high amount of skills and experiences from the employee, it is still valuable to provide a framework that ensures that all parts of the process are completed in the right order and support the flow of electronic documents between the actors.

Both, ordinary and ad-hoc workflows can be managed with the help of workflow management systems (WfMS).

Definition of WfMS

A workflow management system defines, creates and manages the execution of workflows through the use of software, running on one or more workflow engines, which is able to interpret the process definition, interact with workflow participants and, where required, invoke the use of IT tools and applications (WfMC 2002).

Build-time vs. runtime

An important distinction that has to be made for workflow management systems is between the two modes build-time and runtime. Workflow models are designed during build time. At runtime, instances of the workflow are executed, e.g., a concrete document is submitted to be published in a document management system and a reviewer is notified of the new document to check quality and approve its publications.

Workflow life-cycle

Figure 3-11 shows the workflow life-cycle that starts with analysis and design of business processes, which results in often semi-formal business process models. These models or parts of it are then detailed, so that they contain sufficient information for workflow management systems to be able to process them. This happens during build-time. Afterwards, the implemented workflow model is executed by the workflow engine. The execution can be monitored and logged in an audit trail. Periodically, these audit trails should be analyzed to further improve the workflow. This "meta-process" can also be seen as a learning process which results in another iteration of business process analysis and therefore re-enters the life-cycle.

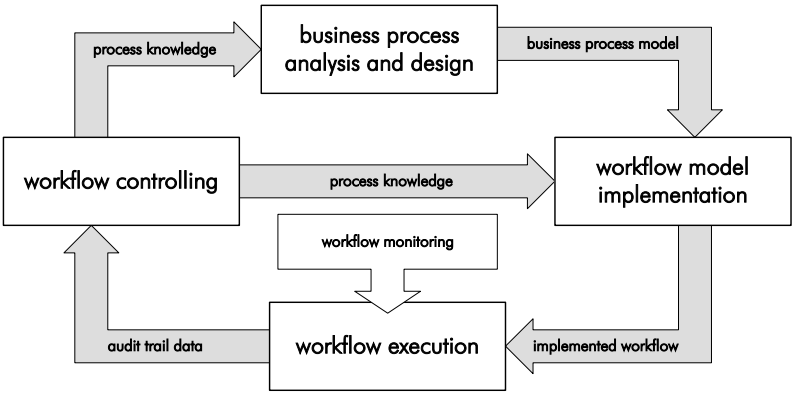

Figure 3-11. Workflow life-cycle (based on Mühlen/Hansmann 2003, 390)

Actions, events, actors

Workflows consist of *actions* (synonyms are tasks and functions) and events. *Actors*, which can be users of the WfMS or software applications for fully automated parts, are required to carry out the actions. *Events* can be seen as preconditions that need to be fulfilled so that an action can start or as results of actions. Boolean operators (and, or, xor), so-called *connectors* can be used to connect events with actions. Furthermore, objects of the workflow, e.g., forms or documents, are modeled as well as parameters required to invoke applications needed to perform individual steps of workflows. With these modeling elements, complex processes can be electronically reproduced, with actions that have multiple preconditions or different possible results. During runtime, the system watches for events that trigger a workflow. If all start conditions for the first action in the workflow are fulfilled, the system triggers the execution of the first action by either notifying the actor (often via email with a link to the workflow action in the WfMS) or directly invoking the function in the software application that can execute the action. Then, the actor carries out the action and reports the result back to the WfMS. The results then trigger one of the next actions and so on, until the workflow is finished.

Benefits

The usage of WfMS has several potential benefits that are more or less applicable in different organizational settings:

- *reducing waiting time*: the average time between the completion of one workflow action and the beginning of the next action can be reduced due to the notification the system sends,
- *monitoring process status*: the monitoring component enables real-time supervision of the states of all currently running processes,
- *prioritizing processes*: if a workflow needs to be completed as soon as possible due to an external signal, it can be prioritized in the WfMS and

therefore gets precedence over other processes that occupy the same resources,

- *distributing work efficiently*: task assignments are not bound to concrete persons, but to roles or groups, so that the workflow system can distribute work more efficiently using pre-defined rules,
- *improving flexibility*: application systems that use WfMS encapsulate functions and data required for coordinating tasks in a workflow that require multiple applications to be carried out and thus are more flexible regarding changing processes and software upgrades.

Design of workflows

Design of workflows at build-time is often supported graphically. Another common way to design workflows is to create XML files. The Workflow Management Coalition (WfMC) is a consortium of companies that have interest in development and standardization of workflow products and techniques. XPDL (XML Process Definition Language) is an XML-based standard developed by the WfMC that can be used to exchange workflow definitions across applications and platforms. The WfMC has also developed an ideal architecture for workflow management systems (Figure 3-12).

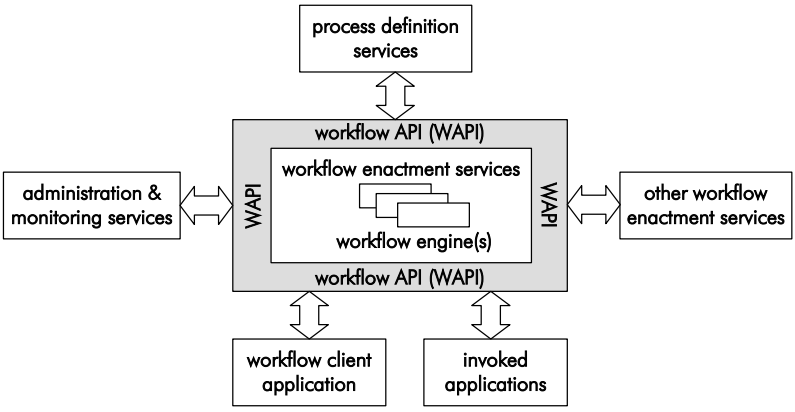

Figure 3-12. Architecture of workflow management systems (based on WfMC 1995, 20)

Worflow engines

The core component is a workflow engine. In order to make the engine independent from supporting tools, a workflow API (application programming interface) has been developed and standardized. Therefore, workflow clients, applications that are invoked by the workflow engine, monitoring and definition applications have to comply to this API and can then be used with any workflow engine that implements the standards interface. Today, most workflow engines are embedded in other applications, e.g., Intershop enfinity or IBM Websphere. Examples for stand-alone workflow

engines are enhydra shark, Fujitsu Interstage Business Process Manager (i-Flow), jBPM and WfMOpen.

3.4.3 Web Service Composition

A recent approach to increase flexibility of workflow management systems is based on Web services. The advantage of Web services compared to proprietary WfMS is that functions provided by the respective applications are described and invoked in a standardized way. Therefore, the limitation to a small set of supported applications that exists in all WfMS diminishes since a large number of developers of application system is expected to use Web services to describe the services offered by their applications.

Coordination. A first step towards process management based on Web services is to coordinate the execution of several Web services. Three standards that add coordination capabilities to Web services (WS) will be briefly described in the following: WS-Coordination, WS-Transaction and WS choreography interface (WSCI).

WS-Coordination

Web service coordination (WS-Coordination) is a framework for supporting coordination protocols. It can be seen as a meta-specification that governs concrete coordination protocols. The framework specifies an extension of the SOAP header called *coordination context* for exchanging unique identifiers between coordinated Web services, as well as two interfaces. The *registration interface* is used to exchange information about the WSDL port of the Web service. The *activation interface* supports exchanging information about the role of the Web service. One central or several distributed coordinators are responsible for coordinating Web services that participate in one workflow. During the activation of a Web service a coordination context is created and the following registration phase binds participating Web services to each other. However, the WS-Coordination standard does not describe how the actual coordination is implemented.

WS-Transaction

This is the goal of Web service transaction (WS-Transaction). Transactions that involve Web services are often long-running transactions in contrast to data base transactions. Therefore, it is no option to lock involved resources over the entire transaction time. In addition to that, it is not clear what basic actions are involved in a transaction and how to roll them back in case of an error (e.g., how sending of an email message can be rolled back). The result is that the ACID principle of data management which states that transactions should be atomic, consistent, isolated and durable, must be relaxed in the Web service context. Consequently, we speak of a *compensation operation* instead of a rollback of an operation. It is left to the implementation of a Web service how the compensation is performed. WS-Transaction specifies two protocols: *business activities* for long-run-

ning transactions as discussed above and *atomic transactions* for short transactions conforming to the ACID principle. Similar to `start transaction`, `commit` and `rollback` messages defined for data base transactions, the business activities protocol specifies a `start business agreement` message that initiates a transaction, `exited`, `completed` and `faulted` messages that Web services can use to report states of their operations to the coordinator and `close`, `complete`, `compensate` and `forget` messages which the coordinator can use to update Web services about the status of the whole transaction.

WS choreography interface

The Web service choreography interface (WSCI) basically defines four extensions to WSDL:

- *exception handling*: this is used to specify what should be done if an exceptional situation is encountered, e.g., a corrupt message is received.
- *transactions*: only the type of transaction (atomic or long-running) and the corresponding rollback or compensate actions are specified. WS-Transactions is used for other aspects of transactions.
- *correlators*: in contrast to the unique identifiers in the SOAP header that WS-Coordination specifies, correlators are used to point to an attribute in the message body that holds the id.
- *time constraints*: different types of time constraints can be specified, e.g., a time frame required between two consecutive invocations of a Web service.

Business process execution language. One abstraction level above these generic mechanisms to coordinate Web services are specifications that enable the description of compositions of Web services, e.g., (1) a request about document meta-data, (2) the following order to purchase and download the document, (3) the final confirmation and (4) delivery.

Composition challenges

Composition approaches have to provide answers for the following questions:

- What types of basic components are composed?
- How are they orchestrated?
- How is data specified and accessed?
- How are concrete services selected?
- How are transactions handled?
- How are exceptions handled?

There are several different proposals for Web service composition languages. The following paragraphs discuss how the specification with the biggest support by large IT companies answers these questions. Business process execution language for Web services (BPEL4WS or shorter: BPEL) is a specification proposed by IBM, BEA and Microsoft. It is based on XML and borrows many elements from earlier specifications such as the Web service flow language (WS-Flow) and XLANG.

Basic and structured activities

The basic components in BPEL are Web services defined in WSDL. These components are called basic activities in BPEL terminology. BPEL processes can also be reused as sub-processes in other BPEL processes by including them as so-called structured activities. BPEL distinguishes synchronous function calls represented by the `invoke` activity and asynchronous function calls represented by the `receive` and `reply` activities. Furthermore, `wait` and `assign` activities are intended for blocking the execution for a certain time frame and assigning values to variables.

Orchestration

BPEL provides five constructs to orchestrate components:

- A `sequence` executes activities in the defined sequential order.
- A `switch` evaluates conditions that are associated with activities and executes the first activity whose condition is true. Similar to the switch statement in C or Java, it is also possible to specify an activity that should be executed if no condition is true.
- A `pick` associates a set of events, such as the receipt of a message with activities and executes the respective activity if the event occurs.
- A `while` loop executes exactly one activity as long as the specified condition is true.
- A `flow` starts a group of activities in parallel and is considered completed if the last activity is completed.

Data types and access

In BPEL, it is possible to define variables similar to variables in programming languages. Variables have a name and a type and can be used as input and output parameters for invocations or as part of conditions. Variables can be of basic, simple or complex types as defined in XML-Schema. XPath can be used to reference basic values within a variable of complex type. The `assign` activity can be used to initialize a variable or set it to a defined value at some state of the process.

Service selection

Selection of services in BPEL uses three constructs to determine a concrete Web service that should be invoked. The *partner link type* is used to determine the roles involved in an invocation, the relationships between the roles and the WSDL port types the roles have to provide. The partner links then identify a so-called *endpoint reference* which is an abstraction of a concrete Web service. Endpoints can either be statically bound to a Web service by specifying a URL, or they can be dynamically bound by using a UDDI registry.

Transactions

Transactions are currently addressed by BPEL only in a limited way. An activity can be specified for compensation handlers that intend to undo the effects of a formerly successfully executed activity. Compensation handlers can also be specified for a scope. The default compensation handler for a scope simply calls the compensation handlers of involved activities in reverse order of usual execution. It is expected that future versions of BPEL will build on WS-Transactions to provide more powerful mechanisms for transaction handling.

Exceptions

BPEL relies on the proven "try, catch, throw" approach for exception handling which is known from several programming languages including C++, Java and C#. Fault handlers can be defined either for one activity or for several activities by defining a scope. Faults are either generated by a Web service using a WSDL fault message or by the execution engine, if a runtime error occurs. The throw activity allows to actively generate fault messages from within the process schedule. If a fault occurs, all currently running activities within the scope are stopped and the activity specified in the catch element of the fault handler is executed. In addition to the fault handler, an event handler can be specified that continuously monitors execution of activities and fires, if a certain event occurs.

Instance routing

For asynchronous message calls and long-running transactions, it is important that messages can be correlated to instances of processes, since it is possible that a process is instantiated several times at once. BPEL allows to specify so-called *correlation sets* that point to message contents used to identify correlated messages uniquely. For example, an ISBN number could be used to correlate a book request with its response.

There is still a lot of development going on in this area. The terms are not yet clarified (e.g., the terms choreography, composition and orchestration are often used as synonyms) and it is not yet clear whether BPEL will be the future standard for Web service composition, although it currently seems to be the most promising candidate. Similar standards like BPML (business process management language) specified by the BPMI consortium and XPDL from the WfMC are either less comprehensive or not equally well suited for Web services. Products that already support the BPEL specification are BEA WebLogic, IBM WebSphere, MS BizTalk Sever 2004, Oracle BPEL Process Manager and SAP NetWeaver.

3.4.4 Semantic Web Services

Semantic Web service is the keyword that denotes the vision to unify developments from the Semantic Web initiative and Web services, respectively composed Web services in order to achieve more automation in the ad-hoc identification and orchestration of Web services.

Standardization and adoption

Definition of standards as well as the implementation and adoption of technologies both succumb to a life-cycle. For standards one could use the organization of the world wide Web consortium (W3C) as an example. Firstly, there is a vision or a user need (e.g., Berners-Lee's vision about the Semantic Web). Then somebody expresses this vision in written form and submits it to the consortium. If the consortium can retrace the idea it publishes a note and creates a working group that develops a concrete technical concept. This concept is detailed and reworked over time. Versions of the concept are published as working draft, candidate recommendation, proposed recommendation and finally as recommendation. In parallel, companies implement the technology in prototypes and products. Publicly

available products might get more and more adopted by customers. At first, innovators and then "early adopters" buy into the technology. Later on, it is spread widely as the early and late majority adopt it. Finally, laggards implement the technology, often years later.

Maturity of Web services

Translated to Web services, Web service composition and semantic Web services, the technologies can be located in the following life-cycle phases: Web services are well standardized, several products that implement Web services are available and already in their second version. The early majority seems to be starting to apply these products in (parts of) their organizations.

Maturity of Web service composition

Web service composition, however, is still not as stable. There is no single standard, but several specifications from a couple of (important) software vendors, with BPEL being the most promising candidate for further standardization. First products have been available for several months and innovators or maybe even early adopters seem to use it already in pilot projects or first operational settings. However, it seems like there is still a long way to go until broad adoption is reached.

Maturity of semantic Web services

Semantic Web services are currently a vision. The need for an automated discovery and composition of Web services is identified. Although the term is used in an increasing number of publications (25,000 hits in google in October, 2004) and several working groups have been established, it seems to be still far from being ready for operational use. Especially the conjunction between OWL-S (OWL for services), a promising technology for semantic Web services, and Web service composition languages like BPEL that do not get obsolete with OWL-S are unclear. Products are not yet available on the market.

Integration developments

Figure 3-13 shows the development of data exchange and integration as well as of access to remote functions from the perspective of Web services and the Semantic Web. The goals of these approaches are ultimately that machines understand exchanged data and that services are universally interoperable. Steps towards these goals were taken under the headlines EAI (Enterprise Application Integration) and Semantic Web on the data comprehension axis as well as Web services and Web service composition on the service interoperability axis. Semantic Web services should integrate technologies from both development lines in the future.

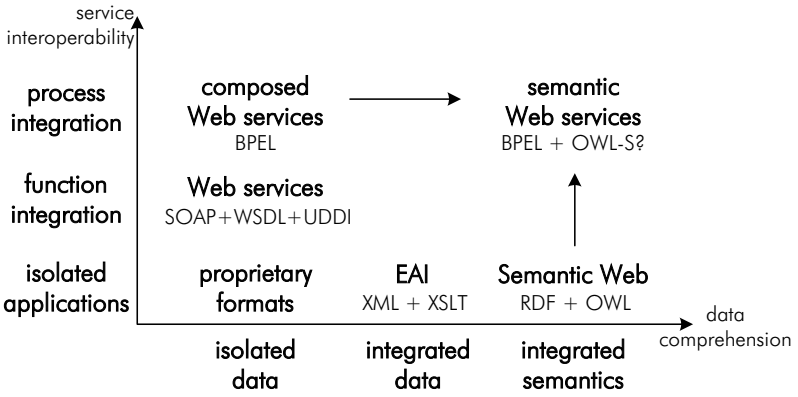

Figure 3-13. Semantic integration and function integration (based on Daconta et al. 2003, 7)

DAML-S, OWL-S

The most promising approach for this integration is coming from the DAML community (DARPA Agent Markup Language). The central specification is the DAML Services specification which was developed under the name DAML-S and was later translated to OWL-S because OWL succeeded DAML+OIL as ontology language for the Semantic Web. OWL-S uses OWL vocabulary to define four sets of data that semantically describe Web services: service, service profile, service model, and service grounding. *Service profile* describes what the service does, *service model* specifies how the service does it (internal flow) and the *service grounding* specifies how the service can be accessed (protocols). The service part then combines these three descriptions and must be referenced by the actual resource that offers the service (like the link to an RDF document describing an HTML page).

SWSL, PSL

There is also a lot of research undertaken on behalf of the Semantic Web services Language (SWSL) committee that tries to reuse the Process Specification Language (PSL) for Web services. PSL builds on first-order logic and is an ISO standard.

Outlook

All in all, there are a lot of open questions to be answered before the vision of automated discovery, invocation, composition, execution and monitoring of Web services can become reality. An additional obstacle for the development of semantic Web service standards is the lack of query language standards for higher level Semantic Web standards (RDQL the Query Language for RDF might be a candidate) and the connected slow adoption of these standards.

Questions and Exercises

1. XML files can be used as structured representation of initialization values or parameters for software applications. Get the file Vizx3D.ini from the supporting Web site[12] and create an XML representation of it.
2. Develop an XML Schema with local definitions for the following XML instance document:

```
<?xml version="1.0" encoding="UTF-8" ?>
<product id="P0815" title="Little Teddy">
 <descr>The teddy bear you love. Best seller!</descr>
 <category>toys</category>
 <price currency="Euro">
  <wholesale>9.99</wholesale>
  <retail>14.99</retail>
 </price>
 <category>stuffed animals</category>
</product>
```

Listing 3-21. XML instance document

3. Edit the result from question 2 so that the schema uses global definitions. Compare the two schema definitions and judge their suitability for this example.
4. Restructure the XSL transformation shown in Listing 3-6 so that it uses only one template and two for-each loops. Then restructure it again so that it uses three templates and no for-each loop.
5. XSLT is often used to convert XML instance documents from one XML file format into another in order to exchange data between organizations. Get the XML and XSD files orderCyberware and orderXTerra from the supporting Web site[12] and write an XSL transformation that converts the data from Cyberware format into the XTerra one.
6. XTerra sends delivery notes back to Cyberware to confirm that the ordered goods are on their way. Write an XSL transformation that converts the data from the XTerra into the Cyberware format. The source and target files can be found at the supporting Web site[12].
7. Michael Meyer is a student and interested in knowledge management (KM). He has written a diploma thesis "Success factors for the introduction of KM". This thesis is available at "http://www.michael-meyer.de/uni/km-paper.pdf". He participates in a community that deals with that topic and serves as a forum to discuss questions around KM.

[12] URL: http://www.wiwi.uni-halle.de/maier/EKI/

The community has a homepage at “http://www.brint.com/km”. The thesis was supervised by Prof. Dr. Reyam who has a homepage at `http://www.reyam.de`.

Create an XTM topic map that represents the facts stated above. Pay attention to the correct choice of classes and instances, topics and occurrences, associations between topics as well as the roles that topics play in associations.

8. Make yourself familiar with the elements defined in the Dublin Core Metadata Initiative. Write an RDF file that describes the accompanying Web site of this book, using the Dublin Core standard.
9. Use the foaf-a-matic, an online tool available at the FOAF project (Friend Of A Friend) homepage http://www.foaf-project.org/ to generate an RDF representation of the description of yourself and your friends. Examine the results.
10. Make a list of the three internet applications you use most frequently (e.g., online shops, mail accounts, etc.) and write down the data each of them holds about the user. How much data is stored redundantly in each of the three applications?
11. Download the Web service toolkits from Google or Amazon and analyze the WSDL files.

Further Reading

A large number of books and articles deal with integration from a variety of perspectives. The following list contains works providing an overview of integration in general or specifically an overview of one of the distinctive integration areas discussed in this chapter.

Hunter, D., Cagle, K., Dix, C. 2004: *Beginning XML*, 3rd edition, Wrox Press, Birmingham 2004

This book covers all details of XML, XML Schema and XSLT. It also deals with DTDs, the predecessor of XML Schema and the programmer's view on XML documents. It is well structured and easy to read. Despite its title it is not only for beginners, but also a reference for experienced programmers.

Berners-Lee, T., Hendler, J., Lassila, O. (2001): *The Semantic Web. A new form of Web content that is meaningful to computers will unleash a revolution of new possibilities*. In: Scientific American, May 17th, URL: http://www.sciam.com/print_version.cfm?articleID=00048144-10D2-1C70-84A9809EC588EF21, last access: 2005-02-04

Tim Berners-Lee, the founding father of the WWW, presents his vision of the next generation of the WWW that gives an impression of the opportunities that a systematic treatment of semantics in the Web context could create.

Daconta, M. C., Obrst, L. J., Smith, K. T. (2003): *The Semantic Web. A Guide to the Future of XML, Web services and Knowledge Management*, Indianapolis (IN), USA 2003

This book provides a comprehensive overview and discussion of the most important standards and technologies that are part of the Semantic Web initiative.

Fensel, D. (2004): Ontologies: *A Silver Bullet for Knowledge Management and Electronic Commerce*, 2nd edition, Berlin et al. 2004

Fensel provides an introduction into the concept of ontology and gives an overview of languages, tools and standards for the development of ontologies as well as selected examples for the application of ontologies in knowledge management, business transactions as well as enterprise application integration.

Alonso et al. (2004): *Web services - Concepts, Architectures and Applications*, Heidelberg 2004

The book describes not only the basic Web service standards WSDL, SOAP and UDDI, but draws a complete picture including preceding tech-

nologies, middleware and EAI concepts as well as extensions to Web services for security, routing, coordination and composition.

The DAML Services - Publication Archive, online available at http://www.daml.org/services/owl-s/pub-archive.html, last access 2004-10-26

The DAML Services - Publication Archive contains a wealth of technical papers describing OWL-S and its predecessor DAML-S. Especially worth reading are the papers “Automated Discovery, Interaction and Composition of Semantic Web services” and “Toward a Semantic Choreography of Web services”.

4 Knowledge Services

Overview

In addition to integrating a number of data and knowledge sources on the basis of infrastructure services, EKI offer advanced knowledge services that support knowledge processes. These services can be classified in accordance to the three types of media to which knowledge is bound, documents, people and social systems. Publication services target explication and documentation of knowledge. Collaboration services target joint development of knowledge in teams, communities and knowledge networks. Learning services primarily help people to internalize knowledge, but also foster learning circles and networks. Finally, discovery services allow for searching the entire knowledge base which is not only restricted to documented knowledge, but also contains links to holders of knowledge. These services are ideally integrated with each other on the basis of context provided by data and semantic integration as well as information about other participants, groups and knowledge networks provided by user integration. Finally, the various application systems offering services in the four domains can be integrated with the help of technologies of function and process integration (chapter 3, 147ff).

Section 4.1 discusses discovery services. These support the entire search process, consisting of preparation, search, visualization and exploration. Section 4.2 discusses document management systems (DMS) according to the document life-cycle as well as Web content management systems (WCMS) using the content life-cycle. Both classes of systems offer functions to handle content or documents respectively throughout the entire life-cycle of creation, capturing, storing, editing, publishing until archiving. Section 4.3 discusses asynchronous and synchronous communication tools as well as classes of systems to support cooperation, i.e. conferencing, shared information spaces and workgroup computing. Section 4.4 presents learning services and introduces basic concepts required to develop contents for learning, or reusable learning objects, and discusses functions of e-learning platforms and other tools required for the entire process of developing learning contents.

Learning objectives

On completion of this chapter, you should be able to

- give an overview of knowledge services and how they can support discovery, publication, collaboration and learning,
- explain techniques for preparing contents for efficient search and how search results are ranked and visualized,
- show differences between Internet and Intranet search,

- define document and document management system (DMS) and relate tasks of handling documents over their entire life-cycle to supporting DMS functions,
- give an overview of the most important laws and regulations affecting document management,
- define the term content and describe the content life-cycle and how corresponding tasks are supported by Web content management systems (WCMS),
- compare Web publishing with and without WCMS and name the advantages of publishing with WCMS,
- classify modes of communication and name information systems supporting them,
- understand motivation and goals of the discipline computer supported cooperative work (CSCW), describe the most important system classes and their functions to support cooperative work,
- delimit the views on learning of behaviorism, cognitivism and constructivsm and assess influences on systems to support learning,
- define the term e-learning platform and show which principles and standards guide the design of learning objects and units.

4.1 Discovery Services

As the amount of information quickly increases, it becomes more and more important to have effective methods for information retrieval (IR). Searching information bases has thus become a major issue in many organizations. Research on search algorithms, visualization of search results and (text) retrieval as well as vendors of software applications that package some of these methods have catered to the needs of these organizations. Today, numerous tools help to support search, presentation and navigation of information and knowledge bases which is termed discovery services here. Still, there remain several issues.

Challenges for Information Retrieval

The following list gives a brief overview of some of the challenges:

- understand the user's information need,
- quick results vs. extensive search algorithms,
- find information stored in different locations, source systems and formats in one search,
- deal with documents in different languages,
- understand structure and semantics of data and documents searched,
- present the right portion of information in the right level of detail.

Understand user need

First of all, discovery services have to understand the user needs and should hide implementation details from the user. Both demands pose substantial challenges to designers of discovery services. Users usually must choose between search modes that determine how to find the information they need. Three alternative types of searches have to be distinguished. First, users can browse through some kind of hierarchy or network of terms or categories to find a set of documents (browsing) or they can decide to specify search criteria and let a search engine find results (searching). The latter can be further subdivided into a keyword-based search and search based on meta-data. Both forms can be combined, e.g., users could search for documents containing `skill management` (keyword search) that were created in 2001 (meta-data search). In both search cases, users should be familiar with the search strategy of the search engine in order to express needs as required. The challenge for future research is to understand queries formulated in natural language and give concrete answers instead of presenting a number of sources that include some keywords.

Quick search results

Users do also want quick answers to their queries. Empirical studies showed that many users cancel queries that take longer than 3 seconds. That implies that some search strategies that would lead to better results can not be implemented due to computational complexity and the resulting time required for delivering results as user acceptance is too low. Certainly, user patience depends on the search situation, but the quest for quicker search results calls for time- and resource-consuming preparation of the search domain, e.g., indexing, and execution of searches on powerful machines.

Different source formats

Another challenge for search systems is to deal with a number of different sources. Relevant data could be stored on users' PC, on servers in the Intranet or the Internet. It could even be stored on PCs of colleagues and ideally should be found there as well (section 6.1, 352ff). The same is true for documents in different formats or from different source systems. Next to text documents in varying formats, email messages can be considered as an important source of information, as well as human experts that should be identified based on a skill management system. Sometimes, valuable information can also be stored in data base systems or other enterprise systems that store data in proprietary formats (section 1.5.2, 59ff). Search engines should handle this variety of formats and sources without requiring active user decisions where to search. However, the more source systems are considered, the more complex and time-consuming is the query.

Language dependency

As many advanced search strategies are based on linguistic models, the use of different languages in the documents is another challenge. The languages of both user query and documents should be detected and ideally queries formulated in one language should also find documents in different languages. This is especially difficult as most words do not have a single but multiple possible translations and even in one language there are

often several ways to express one idea. Context has to be considered, but it is missing in the query most of the time.

Semantic result processing

That leads to recognition of semantic information, which helps to identify relationships between terms, e.g., synonyms, subclasses, homonyms or antonyms. For example, the search engine should be able to differentiate between documents that use identical terms in different contexts (homonyms) and - if not able to determine the right context based on the user query - at least group the results according to the different contexts it can identify. The structure of a document which is often given in a table of contents and headlines can be valuable to determine the context and identify interesting sections of the document (e.g., author, or bibliography).

Number of results

Finally, users need the right information portion, e.g., the right level of detail or information corresponding to the user's level of skills. Sometimes, a whole document is too much and a chapter is more valuable whereas in other situations the user needs not only a single email message or contribution to a newsgroup, but the whole thread or series of replies.

Summing up, challenges can be described in computer- or user-centric views on the problem. The computer-centered view aims at building efficient indexes, processing user queries with high performance, and developing ranking algorithms which improve quality of answer sets. The human-centered view focuses on studying behavior of users, understanding their main needs, understanding semantics of objects in the search domain and determining how such understanding affects organization and operation of the retrieval system.

Search and preparation

Preparation of resources is a foundation for efficiently searching resources by creating an index which is accessed during the actual search process. Figure 4-1 gives an overview of the steps during these two phases and shows their relation. In the following, search phase and preparation phase will be explained in detail (sections 4.1.1 and 4.1.2). Then, we will discuss differences between Internet and Intranet search (section 4.1.3) and explain ways how to support explorative search as an important search mode besides search by query definition (section 4.1.4).

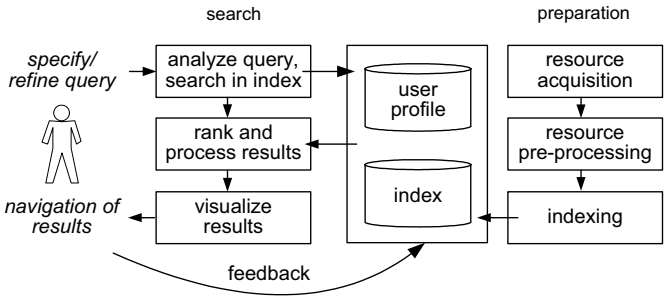

Figure 4-1. Steps in search and preparation phase

4.1.1 Preparation Phase

The preparation phase comprises the acquisition of information objects in the search domain and has to be completed before a search can be started. While online searches that scan documents directly after the user has started a search may still be appropriate in cases of limited search domains, e.g., for desktop systems, they can not be applied to searches in an organization's Intranet or the entire Internet. Preparation consists of determining the search domain, e.g., documents to be indexed, extracting and processing objects in the search domain and finally building an index.

Resource acquisition. The most common way to acquire resources for indexing is using a crawler. A crawler is a program that browses the Intranet or Internet in a systematic and automated manner. Synonyms for crawler are Web crawler, spider, Web robot and Web wanderer. Besides the different names, they all follow the same principles.

Function of a crawler

A crawler consists of gatherer, loader and checker components as well as a URL data base. Seven consecutive steps describe its function:

1. Starting point is a list of URLs, folders on file servers or other sources of information.
2. The loader assigns tasks to visit information sources to the gatherers based on the list.
3. The gatherer collects the data using http requests or other protocols and hands them over to the checker.
4. If the returned resource (e.g., document, html page) contains hyperlinks or the directory has subdirectories, the gatherer appends the new URLs to the URL data base.
5. If the URL is not pointing to a resource, it is deleted from the data base, otherwise it is marked as visited.
6. The checker examines the resource and decides whether it should be indexed. In order to accomplish this, it first checks the resource's format (e.g. html, pdf, doc, rtf) and looks for a corresponding filter that can extract plain text from the resource. Then, it checks if the resource has already been indexed (doublet-check).
7. If all checks are o.k., the content is handed over to the indexer.

Conditional GET

In the common case that HTTP is used as access protocol, a conditional GET request can be used to reduce network traffic. The conditional GET statement can contain parameters so that the requested resource is only returned by the Web server, if the condition is fulfilled. The most important condition is the modified date. As the indexing task has to run periodically (daily or weekly), the crawler can specify the date of the last indexing as parameter and the document is only transferred if it has been modified since then. Otherwise, the HTTP status code 304 (not modified) is sent back instead of 200 (ok).

Resource preprocessing. Once the contents of the search domain are determined, the resources must be prepared for indexing. In the following, we will briefly review this process for text documents which converts the document into a representation of (parts of) its contents.

Lexical analysis

The first step is lexical analysis. The purpose is to convert the stream of characters in the document into a stream of words. What seems to be an easy task as space characters delimit each word is quite difficult at closer inspection. One challenge is how to deal with digits. On the one hand, numbers usually are not useful search terms and therefore are often neglected during indexing. On the other hand, numbers that represent years, prices and sales numbers can be important for document retrieval. Another decision has to be taken concerning how hyphens should be treated. If the composed word is decomposed into its elements a part of the semantics can be lost. Dots usually separate sentences and can safely be removed if there is no further analysis on sentence level, but again a few exceptions as DNS names or variables in object-oriented programming can have dots as integral part of their names. Finally, the lexer (short for lexical analyzer) converts characters to either lower or upper case as the search is usually case-insensitive.

Language detection

The language of the document has to be determined before elimination of stop words can be applied. That can be done either by using linguistic analysis (generic) or based on comparison of a few randomly chosen words with dictionaries (one for each supported language). Language analysis can be done on the level of documents. However, this is not sufficient in cases where documents contain paragraphs or even single words in different languages.

Elimination of stop words

Before the document is being added to the index, some improvements have to be taken in order to make the index more effective. First of all, a list of stop words is processed and all words included in the list are removed from document representations. The reason for that is that words that are present in nearly all documents are not suitable as index terms. Examples for stop words are `is`, `a`, `and` and `to`. In general, words should be eliminated that are not discriminatory, i.e. are not suitable to distinguish different documents. Candidates for stop words are articles, prepositions and conjunctions. A second advantage of this method is that index size can be reduced by 40% on average.

Stemming

Another technique that improves index quality is called stemming. It reduces words to their stem by removing pre- and suffixes, so that nouns are all singular and verbs are all in their base form. Most stemming algorithms can only deal with regularly built forms, which is sufficient for most words, but can lead to unwanted results in special cases. Again, a trade-off between fast processing and best results has to be made.

Noun-groups

Finally, in some cases it makes sense to further reduce the number of index terms. Almost all keywords used regularly are nouns. That is because nouns are usually more distinct than verbs. An algorithm for auto-

matic index term selection can make use of this knowledge and searches the text for nouns and especially for nouns that are grouped together within a sentence. These noun-groups are identified based on their syntactic distance in the text, measured by the number of words or characters between them. So, noun-groups are used as index terms instead of single words.

Indexing. An index is a data structure that contains redundant data describing and/or pointing to text. It is used to speed up search, as otherwise online search of the whole document set has to be done for every query which is inefficient and not feasibly for texts larger than a few MB. Indexing is only a question of the most efficient form of storing index terms and their occurrences, as a lot of work has already been done in preprocessing.

Inverted list

The most commonly used index structure is an inverted list (also called inverted files). Index terms are sorted alphabetically (the vocabulary) and point to where these terms appear in the documents (the occurrences). The occurrences can be expressed in terms of characters, words or blocks of a certain size (e.g., 16 or 64 kB). The finer the occurrence is saved, the more memory it needs. Therefore, it is often accepted that a small portion of the text has to be inspected by users in order to determine the exact position. Indexes using blocks of 64 kB take up only 5% to 9% additional storage space for large text collections (e.g., 2 GB ASCII text) depending on the use of stop words (Baeza-Yates, Ribeiro-Neto 1999, 195). Vocabulary and occurrences are often stored in separate files in order to hold the whole vocabulary in memory.

Text compression

Text compression can be used to further reduce the size of the index. For example, the Huffmann and the Ziv-Lempel algorithms have good characteristics for using them on the indexes, e.g., in terms of compression ratio, speed, memory consumption, usability for pattern matching and random access. Another possibility is to build a hash table over the index and to hold only the hash table in memory.

Index merging

The size of the index is relevant not only for search time, but also for construction time. When the index is built, the vocabulary together with the occurrences is held in memory. Once the memory is full, the partial index is written to disk and a new partial index is developed. When all documents are indexed, the partial indexes have to be merged. Two index parts are examined, identical index terms in the vocabulary are merged, their occurrences appended and the new partial index is written to disk. This is done with all index parts of the first generation so that half as many parts exist after the first cycle. The procedure is repeated until only one index is left. Merging can take up a considerable amount of the entire indexing time (about 30%).

4.1.2 Search Process

Once the index is built, users can initiate search processes. The users' queries are analyzed and searches are executed before results are presented and can be navigated.

Query analysis and search. The types of queries a user can initiate depend heavily on the index, its structure and the search algorithm. The pre-processing that took place during index construction (stop word removal, stemming, term selection and compression) has to be applied to queries (search terms) as well.

Query types

The simplest form of search is a single-word query, but often user needs cannot be expressed in a single word. Keywords have to be connected with Boolean operators (AND, OR, NOT). Usually, search engines use the AND operator by default. As a result, search engines initiate single-word queries for every given keyword and merge the results using set operations. Therefore, Boolean search is classified as a set-theoretic search approach. It is noteworthy that search engines do not use a NOT operator in a set-theoretic sense as this might require time-consuming processing of a probably huge set of documents (all documents that do not contain the given keyword). Instead, the operator BUT was introduced that operates only on the result sets of the other single-word queries and selects those elements that do not contain the keyword specified in the NOT clause.

Fuzzy logic

Boolean searches have one important drawback because they are limited to simply distinguishing between documents fulfilling the Boolean search expression and documents that are not covered by the search expression. Thus, all documents selected are seen as equally fulfilling the criteria so that ranking is impossible in plain Boolean search systems. Fuzzy logic was introduced to overcome this limitation. With fuzzy logic, the result of checking whether a document fulfills a given search expression can not only be yes or no (1 or 0), but also something in between (floating point numbers between 1 and 0). That opens room for additional search criteria like term frequency and proximity of terms that represent the degree to which a document fulfills the user need.

Pattern matching

Another feature that many users expect from search engines is pattern matching, because they are used to it (at least in a basic form) e.g., from the search for filenames in the operating system. Several types of pattern matching can be distinguished:

- prefix search, e.g., `bus` would find `bus`, `busy` and `business`,
- suffix search, e.g., `ting` would find `interesting` and `resulting`,
- substring search, e.g., `tal` would find `talking`, `rental` and `metallic`.

Examples for more advanced techniques of pattern matching are allowing errors and regular expressions which will be briefly described in the following.

Allowing errors

Allowing errors means that misspelled words in queries or in documents (e.g., due to errors in character recognition or authors' mistakes) are matched with the correct version of the word. The technique is based on similarity between correct word and misspelled version. A common measure for similarity is the edit distance. Edit distance between two words is the minimum number of character insertions, replacements or deletions that are needed to convert one word into the other. If the space character is included in the search then also erroneous split-ups of words can be compensated. Sometimes, this kind of search technique is sold as phonetic search, although phonetic search must be based on more complex linguistic models instead of simple character replacements. An application area for phonetic search is the retrieval of persons based on their names, e.g., to find all persons called "Maier" no matter whether they are written with `e` or `a`, `i` or `y`.

Regular expressions

The last type of pattern matching is regular expressions that are usually combined with the use of extended patterns. Basically, regular expressions define three operations on the basis of substring search: union, concatenation and repetition. Union finds words containing either of the substrings (like the Boolean OR) and is written `(s1 | s2)`. Concatenation finds words that contain the first substrings immediately followed by the second and is written `(s1 s2)`. Repetition finds words with zero or more contiguous occurrences of the substring `(s1*)`. Most implementations of regular expressions also contain extended pattern matching capabilities like wildcards to replace one or more characters, conditional expressions or character classes (digits, letters, vowels and consonants).

Phrase search

Searches for phrases might sometimes return better results than searches for a combination of keywords. This is especially useful if the words contained in the phrase are not helpful as discriminators. Consider the phrase "to be or not to be" which is associated with William Shakespeare and Macbeth. None of the words would be a helpful query term, although the whole phrase will surely find documents dealing with Shakespeare and Macbeth. Searches for phrases demand different indexing strategies and full text search in a narrow sense as stop word elimination and stemming is counter-productive in this scenario.

Ranking and result processing. The ranking of documents is used to sort results so that hits that fulfill the user need most likely are presented first. The mechanisms consist of choosing and applying good criteria and a mathematical model to compare different documents based on that criteria.

Search models

Besides set-theoretic search models that were briefly introduced above, there are also algebraic and probabilistic models. The most prominent algebraic method is the vector model. Both, query and result document are represented as n-dimensional vectors of weighted terms. The similarity of these two vectors is then calculated. A measure for similarity is the mathe-

matical comparison of them that can be expressed e.g., in the cosine of the angle between the vectors.

Neural networks

Artificial neural networks are another technique that is used for pattern matching and ranking in search engines. Neural networks try to imitate the structure of human brains by using artificial neurons that are interconnected with weighted connections. Query terms thereby act as input, neurons consist of index terms and documents correspond to the output. Weights at index terms must be adjusted to get good results. The network is usually trained before it is used for searching, but can also be further improved using user relevance feedback to adjust weights during normal usage.

Probabilistic models

Probabilistic models calculate the probability that a set of documents matches the user need. It would be no problem to retrieve the relevant documents if the system would know the exact properties that describe the document set. The user query is only an incomplete and maybe even erroneous representation of the user need. Therefore, assumptions about the properties of the document set must be made that can be improved step by step using user relevance feedback. The user has to specify whether the retrieved documents are good or bad matches. The result is refined with the help of this additional information. A popular implementation for a probabilistic search method is a Bayesian network. Bayesian networks are directed acyclic graphs in which nodes are random variables. Arcs represent causal relationships between these variables. Strengths of these relationships are expressed by conditional probabilities (Baeza-Yates, Ribeiro-Neto 1999, p 48f). The advantage of Bayesian networks is that different ranking criteria like past queries, feedback and current query terms can be combined using a clear formalism.

Criteria for ranking

The key challenge for ranking is finding good criteria that allow to determine to what extend documents fulfill the user need. Some of these criteria are briefly described in the following paragraphs. Often, a combination of several criteria yields best ranking results. One criterion is term frequency. Documents that contain query terms more often are expected to be more relevant to the user. If no exact matching is being required, similarity of query terms and occurrences can also be a criterion. Exact matches are usually better than matches as substring or matches with errors. Proximity of the occurrence of different query terms is also a criterion. The closer together the terms occur (e.g., within a sentence or paragraph), the more relevant the document can be expected to be.

Recent developments in result ranking

Besides these basic techniques, some enhancements to ranking algorithms are discussed. One possibility is to compare data about the document's author and the searching user to determine better or worse matches. It can be argued that a domain expert regularly (but not always) might not want to retrieve documents that contain an introduction to the domain she has already mastered. Vice versa, learning theory suggests that novices might learn more effectively from (documents produced by) intermediates

compared to experts in the regarding domain. Thus, if skill profiles of both, user and author, are available, they can be compared to rank the results. Other criteria like document type (conference paper vs. presentation), reputation of authors or institution the authors work for (e.g., a renowned research institute vs. private, anonymous opinion contributed to a newsgroup), creation date (recent vs. out-dated) or access frequency of the documents (used daily vs. never retrieved since creation) can be used if the corresponding data is available and the use case permits to do so.

Choice of data to be displayed

Finally, once results are ranked it has to be determined which information should be presented to the user. The minimum is filename and location of the resource. Additional information help the user to decide whether the document should be inspected in more detail or can be discarded. Often, a percentage is presented that shows relevance of the result-resource for the given search term calculated by the search engine. Although often differing a lot from the user's subjective rating, it still helps for orientation, e.g., if there is a great lap between the first X results and the following ones. A summary of the document contents, especially the parts in which the query terms occur, can provide additional information that supports quick evaluation of the results. Creation date, size of the document in byte or (even better) in words or pages, authors, as well as time of last access or number of times accessed since creation are other examples of useful meta-data that should be presented together with the results. Meta-data primarily describes the context of creation, but could also be extended to cover the context of application, e.g., the skill level targeted or other descriptions of intended user groups that might be evaluated by the search system and then presented to the user. It is also important to not only deliver documents as results, but include information about domain experts, email messages and community home pages that are relevant. Finally, the result list could include information about related documents (e.g., based on the content, or user access "users who viewed this also viewed that") or the cluster, group or domain a document belongs to.

Visualization and navigation. There are basically two different approaches for the visualization of search results: a textual approach and a graphical one. The textual approach aims at presenting as much useful information about individual resources as appropriate to quickly evaluate relevance of a few top ranked search results, whereas the graphical approach presents relationships between results and gives an overview of the entire set of results. Both approaches could be combined, but most current implementations target either one of the two.

Textual result presentation

In the textual approach, colors can further improve quick comprehension of important aspects. For example, highlighting query terms in the summary of the document content gives the user additional confirmation that the result is relevant. If the results were clustered in some way, a diagram that shows the clusters and a brief description together with the tex-

tual results that are colored according to the cluster they belong could be a step towards integration of textual and graphical representations.

Graphical result presentation

Graphical approaches can be mainly distinguished according to the metaphor and visualization algorithm they use. There is a large number of visualization techniques that can be employed to present search results. Often-used metaphors are maps, trees and nets. A representative of the map metaphor is Kohonen's self-organizing map (Kohonen 1995). Kohonen is the inventor of the algorithm which automatically clusters input documents according to similarities between vectors of concepts they contain and forms the basis for visualization in the map. Each distinct topic or cluster of documents is visualized by a separate mountain or hill. Geographical proximity in the map represents close relation of the clusters or topics. Deep valleys between mountains visualize that the concepts or documents on the two mountains are considered semantically distinct. Figure 4-2 shows an example of a self-organizing map that represents news. The describing keywords are extracted using preparation techniques explained in section 4.1.1.

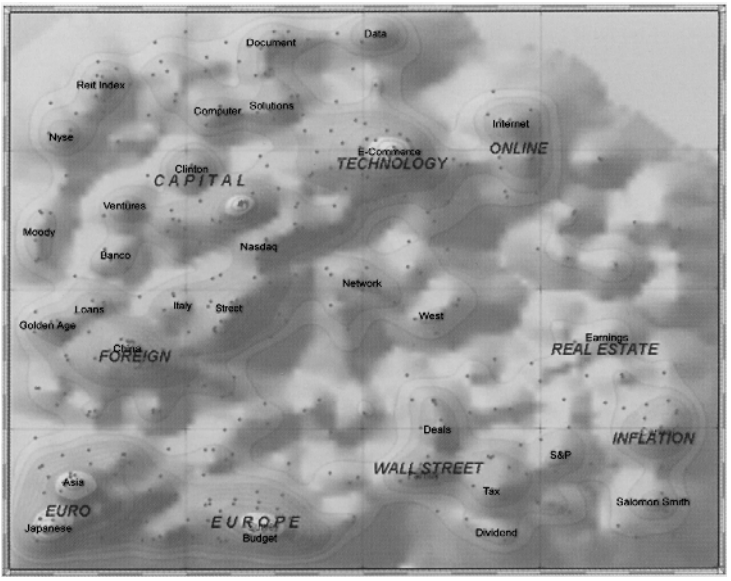

Figure 4-2. Self-organizing map showing articles about finance (Spence 2001, 183)

Hyperbolic tree

A special kind of tree visualization is the hyperbolic tree. It uses the root node in a hierarchy as the central focus and groups nodes that are directly related to the node in the focus around it. Each related node has other related nodes that are displayed further to the outside of the display. A kind of "fisheye view" can be used in order to utilize space on the screen

more efficiently. In this case, the few nodes in the center are displayed using a comparably large font and the many nodes at the outskirts of the graph are displayed using a smaller font. Figure 4-3 shows an example of a hyperbolic tree that is used as navigation structure for a Web site (generated with Inxight VizServer).

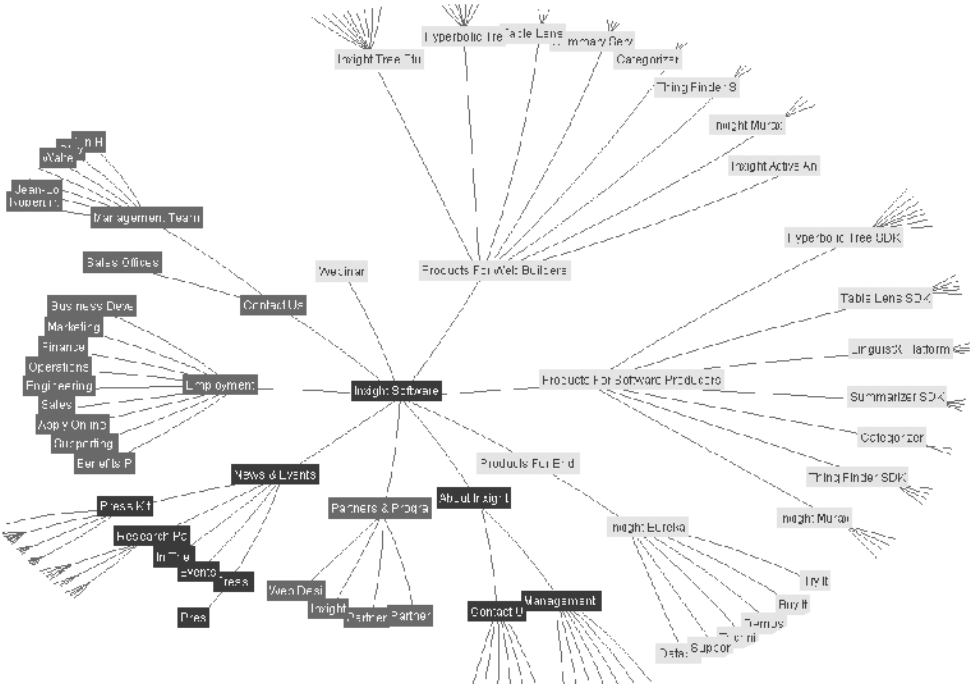

Figure 4-3. Hyperbolic tree representing the structure of a Web site[1]

Multi-dimensional result presentation

An innovative approach to visualize search results is the research prototype Infotop (Maier/Sametinger 2004). It copies the successful multi-dimensional view on structured data from the OLAP (OnLine Analytical Processing) approach and applies it to semi-structured data. Six general dimensions were designed that can be seen as classes of meta-data. The dimensions identified are time, topic, location, person, process and type. Not all dimensions are as important for all types of documents, but most of them apply in any case. Each dimension can have sub-categories for a given document type, e.g., an email message has a sender and a receiver, which both represent the person dimension. One, two or three of the dimensions can be chosen to span the display area. In Figure 4-4, an exam-

[1] URL: http://www.sapdesignguild.org/community/design/hierarchies4.asp, last access: 2004-12-04

ple is shown that uses person and process to form a two dimensional view on documents of type text. The pile metaphor is used to represent the number of results in the respective cell.

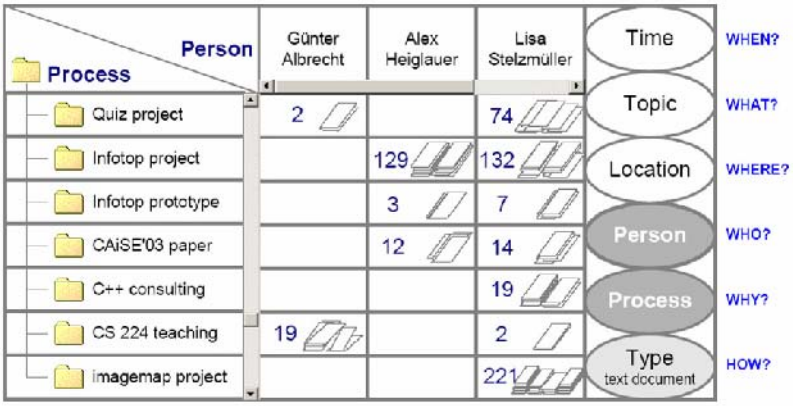

Figure 4-4. Multi-dimensional visualization with Infotop

Dynamic generation of result presentation

The dynamic generation of graphical views on results has gained momentum with the wide-spread use of Java Runtime Environments and the possibility to use Java applets for display. Macromedia Flash is another technique that can be used to visualize results, but the dynamic generation of Flash has only recently been enhanced as XML-data can now be sent to the client and used for visualization. Recent XML-based standards like SVG (Scalable Vector Graphics) and X3D (XML-based successor of VRML, Virtual Reality Markup Language) have also supported the trend to use auto-generated graphical views. XML-based graphics have the advantage that XSLT (XML Stylesheet Language, Transformations) can be used to transform numeric and textual data into graphical representations. However, it is difficult to create more complex visualizations than simple charts using XSLT so that Java and Flash will not be completely replaced in the near future.

Feedback and iteration. An optional step after result presentation is to refine the search. This can be accomplished by either reformulating the query, restraining the result set in some way or providing relevance feedback to the search engine.

Reformulation

Reformulation leaves improving the relevance of retrieved results completely to the user who has to find better query terms, often based on words found in the retrieved documents. Especially the exclusion of words that blur the results can increase result quality. This technique requires competent users and at least a basic understanding of the underlying search algorithms.

Restrainment to a cluster

Restraining the result set is often linked to categorization or clustering of results. Users can determine which group of documents suits best their needs and limit the results to those that are within this group. This is effective, if there is either a good clustering algorithm or a good manual classification of the documents. This kind of feedback is especially interesting if one or more of the query terms have different meanings in different contexts.

Relevance feedback

A technique that leaves improvement of results completely to the search engine is relevance feedback. The user only specifies which documents are close to his or her need and which are not. The search engine then analyzes the documents that were rated by the user and tries to extract discriminating terms, i.e. the properties that differentiate useful from useless results. Based on these terms, a new search is being initiated and new results are presented to the user. If there are documents in the document base that match the user's needs, then usually three iterations are sufficient to reach a good result set. This method is based on probabilistic search algorithms and is well suited for inexperienced users or non-text documents (see multimedia search in section 4.1.3). The feedback can also be saved in a user profile which can be used for further searches to enhance the initial assumptions about user needs.

4.1.3 Internet and Intranet Search Compared

Most of the facts discussed in the previous section apply to Internet search engines as well as to those that work in the Intranet. But there are some important differences between those two search domains.

Content acquisition

First of all, the phase of content acquisition is different. The best Internet search engines often stand out because they manage to index more of the immense document collection that exists in the World Wide Web. Estimations about what percentage of the WWW is indexed by Google which is regarded to have the most complete search index with over 4.3 billion Web pages indexed in spring 2004 and 8.0 billion in December 2004 range from 5 to 40 percent depending on the employed measures. On the contrary, content acquisition is usually not an equally challenging task in an organization’s Intranet. If it is well structured, then there should only be a comparably small number of known sources like CMS, DMS, Web or file server. Information about these sources can directly be provided to the search index, so that no crawling in a narrow sense has to take place.

Data sources

A next difference is that in the Intranet an important task is to not only index documents, but also consider emails, skill profiles and other information that exist in a proprietary format within company-specific applications (e.g. Groupware systems). The more sources of information are included by the search system, the more comprehensive the results and consequently the higher the potential benefits will be. In the Internet, it is no option to search in emails. The only distinction is in what format the

document is in. In addition to html and other plain text formats, most Internet search engines are also able to index a number of standard text document formats, e.g., pdf, rtf, and some widely-used application-specific formats, e.g., Word and PowerPoint documents.

Search engine spamming

An important problem for Internet search is the manipulation of documents to get higher ranked and therefore be more often accessed (spamming). On the one hand, the search algorithm should be transparent to users so that they can use it efficiently, on the other hand the more people know about how ranking works, the better they can manipulate the results. Examples for manipulation techniques reach from using meta tags to place keywords over texts displayed in background color or at a size of one pixel up to delivering manipulated Web pages to robots and letting other users access different pages. The latter can be done using one of the following techniques:

- Door pages are Web pages with redirects that hold content optimized for certain query terms.
- Cloaking is delivering different Web pages based on user agent identification (e.g., Mozilla for a browser and Googlebot for the Google Web crawler).
- IP delivering is the technique of delivering different Web pages based on the IP address of the requestor (IP addresses are relatively constant for the indexing servers).

Social responsibility

As Internet search engines have developed into major starting points for Internet activities of many users, they have to carefully watch what information they spread. In addition to laws regulating spreading of information on the Internet, search engines also have to act socially responsible as they are powerful instruments affecting information exchange on the Internet. For example, providers of Internet search engines try not to index and present results from Web pages containing illegal or offending contents. Web servers hosting such pages are either manually put on a black list or can be identified based on a combination of keywords and not indexed.

Internet search engines

Examples for well-known Internet search engines are AltaVista, Google and Yahoo. As different search engines index different parts of the Web (although there is a large overlap), it makes sense to search with more than one search engine. As this is inconvenient, a number of so-called meta search engines query several search engines automatically, merge the results more or less intelligently and present them to the user. Examples for meta search engines are MetaCrawler, Dogpile and Vivisimo.

Specialized search engines

There are also a number of innovative search engines. Some of them are specialized in multimedia search, such as finding audio files by humming the melody[2], finding pictures based on the rough distribution of colors[3] or

[2] URL:http://www.musicline.de/de/melodiesuche, last access: 2005-02-02
[3] URL: http://wwwqbic.almaden.ibm.com, last access: 2005-02-02

based on given images (query by example[4]) or finding 3D models using a number of 2D drawings[5]. Other search engines specialize in searching product information instead of the whole Web[6], present points of interest geographically in a map[7] or search in a certain domain (e.g., scientific publications[8]). There are also some promising developments regarding clustering[9] or graphical formatting of results[10].

Intranet search engines

Intranet search engines face different challenges. They can be customized to be further enhanced for special use cases. This is especially true for the inclusion of resources in proprietary formats. Also, if the structure of the documents in the document set is similar (e.g., conference or journal papers), some meta-data could be extracted automatically and be used to enrich traditional full-text search with meta-data based search techniques. Examples of well-established Intranet search engines are Autonomy Retrieval, Fast Data Search and Verity Ultraseek. Market leading products are currently under pressure from two sides. Innovative search engines with unique features like ontology based search (Ontoprise) or email search (Xtramind) are one threat. Big software companies which bundle their search engines with popular products (Microsoft, Oracle, IBM) are the other.

Example: Open Text Livelink

Figure 4-5 shows a screenshot of the results page of a search in the Open Text Livelink system. It displays the results with their rating, filename, folder, date, size and a summary in the main pane on the right-hand side. The query term `activity theory` is highlighted in the summaries. On the left-hand side, there is a listing of document authors, or more precisely people that have uploaded documents, an activity index that shows how often documents in the result set have been accessed, created or edited in the last quarter, half-year and year. There is also a list of the folders most of the documents are in as well as a list of topics that are associated with the documents. This additional information on the left pane can be used to restrain query results by checking the respective include check box and pushing the “refine results” button.

[4] URL: http://viper.unige.ch/demo, last access: 2005-02-02
[5] URL: http://shape.cs.princeton.edu/search.html, last access: 2005-02-02
[6] URL: http://froogle.google.com, last access: 2005-02-02
[7] URL: http://local.google.com, last access: 2005-02-02
[8] URL: http://scholar.google.com, last access: 2005-02-02
[9] URL: http://clusty.com, last access: 2005-02-02
[10] URL: http://www.kartoo.com, last access: 2005-02-02

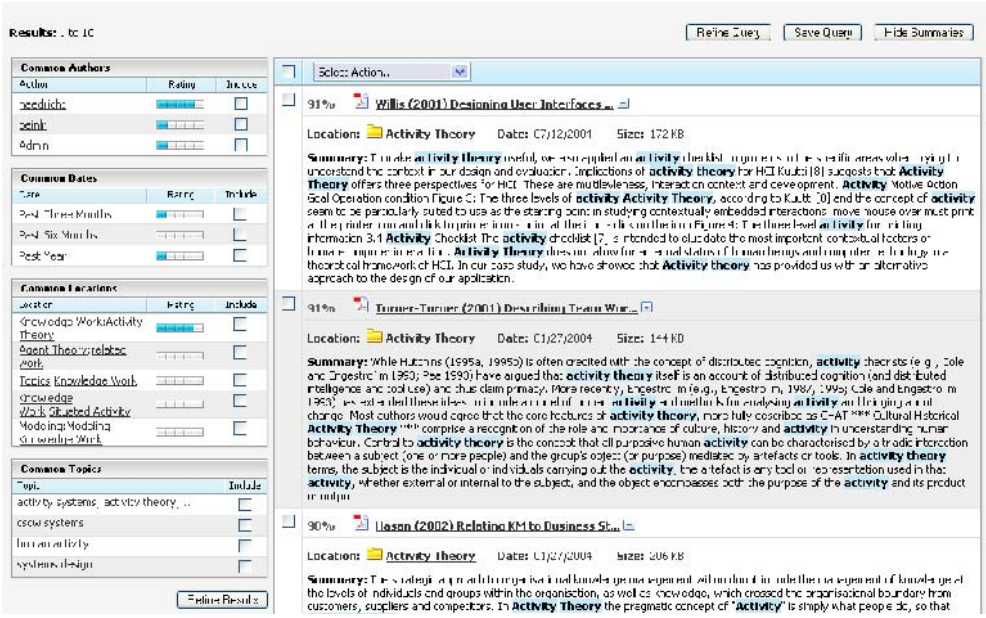

Figure 4-5. Livelink search

Desktop search engines

Recently, many companies target individual PCs with their search engine technologies, because over time it became more easy to locate a file in the Internet than to find it on one's own local hard disc. The Microsoft Indexing Service that ships with Windows 2000 and XP is only a small step to address user needs. Especially the increasing amount of data residing in emails is not found with that tool. Although most email clients offer search services, they are most often slow and limited to email contents. Thus, there are islands of search domains on one single machine which have to be connected with a comprehensive desktop search engine. It would be desirable to have local hard disc, the company Intranet and the Internet be searched simultaneously. Recent desktop search engines from Google, Microsoft and Yahoo address some of these issues. FileHand search, x-friend and blinkx are other product examples.

4.1.4 Explorative Search

In some cases and depending on user preferences, investigative search (search/retrieval in a narrow sense) might not be the best method to find the documents needed. Explorative search, i.e. browsing or navigating the whole document collection, is especially favorable if users lacks knowledge about the domain or user needs can not be expressed precisely. Some of the techniques presented in section 4.1.2 can be used, e.g., a tree, net or

map view on document collections. Static views that result from an underlying (physical) structure like a hierarchical file system are distinguished from dynamic views that are generated based on meta-data and allow to switch between different viewpoints. Of special importance for knowledge infrastructures are knowledge maps.

Knowledge map

Knowledge maps are graphical representations of knowledge and its relation to organizational concepts. Several types of knowledge maps can be distinguished depending on what kind of elements are mapped to the knowledge domain or topic:

Knowledge source maps

Knowledge source maps help to visualize the location of knowledge, either people (sometimes also called knowledge carrier maps) or information systems and their relation to knowledge domains or topics. They can be further classified into knowledge topographies to identify gaps, competence maps to find experts and pointer systems that directly link from challenges within a business process, e.g., to a contact that can assist. Knowledge source maps are used if not only people with knowledge in the desired domain are listed, but also all forms of codified knowledge (sections 4.1.1 to 4.1.3) that are relevant.

Asset maps

Knowledge asset maps are further enhancements of knowledge source maps as they visualize not only that there is knowledge in a document or person, but also the amount and complexity.

Structure maps

Knowledge structure maps show relationships between different knowledge domains or topics and should not only visualize that there is a relationship, but also explain the type of relationship (e.g., subclass, synonym, antonym).

Application maps

Knowledge application maps are combinations of process models and knowledge carrier maps as they describe who should be contacted for help in a certain situation, e.g., at a certain step in a business process.

Development maps

Knowledge development maps visualize learning paths that are recommended to acquire a certain skill by individuals or a certain competence by teams or other organizational units.

Procedure for creating knowledge maps

The procedure to create knowledge maps is a five step process that can briefly be described as follows (Eppler 2003, 202):

- identify knowledge-intensive processes, tasks or issues,
- determine relevant knowledge sources, assets or elements,
- codify these elements, build categories of expertise,
- integrate codified reference information on expertise or documents in a navigation and/or search system that is connected to the work environment of the target group,
- provide means of updating the knowledge map, especially enabling decentralized update mechanisms so that every employee can (re-)position himself continuously within a knowledge map.

There is no standard that describes how knowledge maps should be visualized. Thus, the development of knowledge maps provides a great

deal of freedom for both the determination of what elements and relationships should be part of the models and how they should be visualized. Figure 4-6, Figure 4-7 and Figure 4-8 give examples of knowledge maps and show the variety of approaches to their design.

Knowledge asset map

Figure 4-6 maps (1) central areas of competence in an IT consulting organization and (2) employees according to (3) their level of expertise. The bars indicate whether an employee holds basic knowledge, expert knowledge or is a leader in the corresponding area of competence. The map shows the importance of Mr. Tinner and Mr. Ehrler for the organization because they seem to be competent in (almost) all relevant areas of competence.

Consultants	IT	Strategy	M&A	Accounting	Marketing
Tinner, Jeff					
Borer, André					
Brenner, Carl					
Deller, Max					
Ehrler, Andi					
Gross, Peter					
…					

expert knowledge basic knowledge leadership

Figure 4-6. Example for a knowledge asset map (Eppler 2003, 196)

Knowledge source map

Figure 4-7 shows a portion of the knowledge source map of a multimedia company that develops Web sites, CD ROMs and stand-alone multimedia terminals.

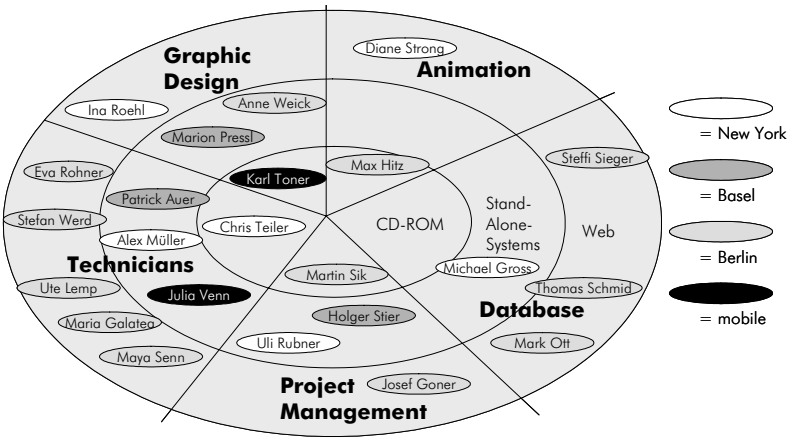

Figure 4-7. Example for a knowledge source map (Eppler 2003a, 195)

The map supports staffing of multimedia projects. It visualizes what experts are available for the company's five areas of competence animation, data base, graphic design, project management and technology know-how and the three product lines Web systems, stand-alone systems, CD-ROMs, at the company's three main locations Basel, Berlin and New York. Additionally, two employees are not located in a single office, but float between the three locations.

Knowledge structure map

Figure 4-8 shows a portion of the main knowledge structure used by the authors' work group as the central access structure to a knowledge workspace implemented in the knowledge management system Open Text Livelink.

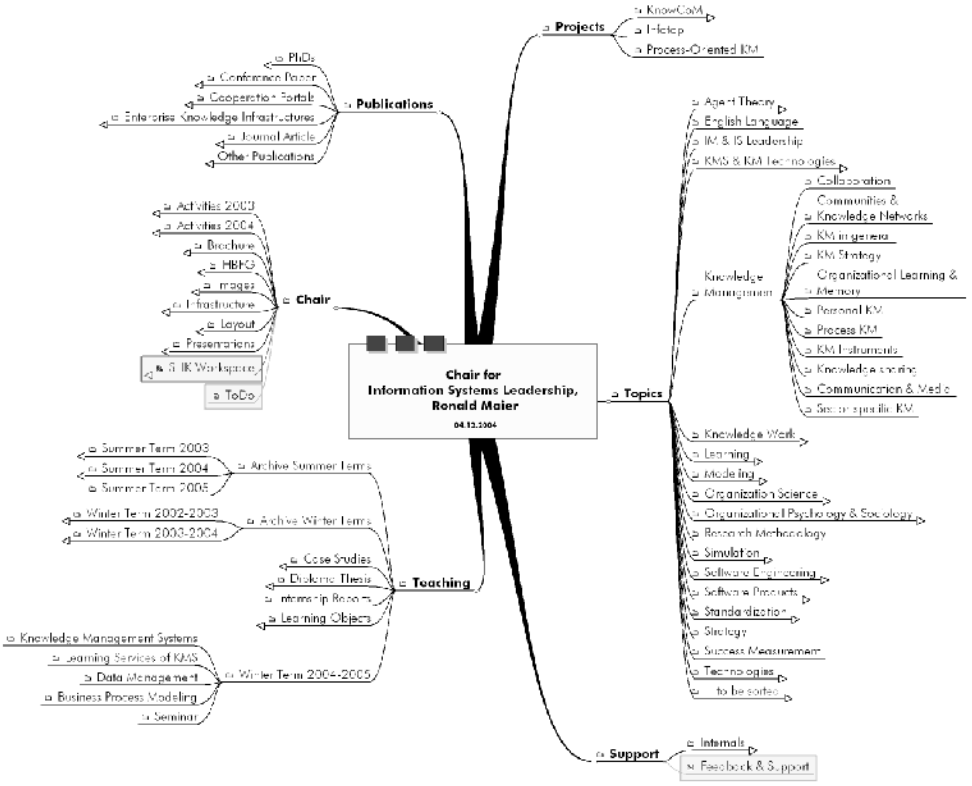

Figure 4-8. Knowledge map of the structure of a knowledge workspace

The first level of the knowledge structure consists of the terms chair, projects, publications, support, teaching and topics. Thus, it reflects the two core processes of a University department, research and teaching. In the *publications* branch, there are a number of workspaces to support research and publications in domains the work group is engaged in. This

includes the Ph.D. workspaces of the research assistants and cooperation portals for publications with external authors. *Teaching* contains workspaces for each individual course or seminar. Students have access to a portion of the material in the workspaces of the courses that they are enrolled in. Moreover, they can contribute to the workspaces and share knowledge with their colleagues. *Projects* represent units of funded thematic research and of cooperations with other institutions. *Topics* are the primary structure to organize e.g., electronic research articles, news, contributions to newsgroups or empirical data that has been collected by the members of the work group. *Chair* reflects internal projects and collaboration workspaces for the work group's teaching assistants. *Support* is a category in which the work with the KMS is supported and reflected. Arrowheads at the end of the branches represent collapsed hierarchies that are not visualized in the map.

Automated knowledge map generation

The map can be automatically generated by a script that exports Livelink's structure, imports it into MindManager[11] and serves as an alternate way to access the knowledge elements stored in Livelink. Each branch in the map contains a hyperlink that directly links to the corresponding object in Livelink.

Knowledge structure maps differ widely between organizations. The maps usually represent primary instruments to structure the organizations' knowledge objects and thus are important navigation aids.

4.2 Publication Services

In this section, we will introduce services of an EKI that aim at supporting explication and publication of knowledge, i.e. efficient presentation, storing, processing and output of objects that represent explicit knowledge.

Two akin aspects can be distinguished in practice: document management and content management. Both are supported by corresponding system classes: document management systems (DMS) and Web content management systems (WCMS). The focus of document management is on handling documents (e.g., purchase orders, protocols, meeting notes) along the entire life-cycle including capturing paper-based documents and archiving them electronically. Content management in a narrow sense deals with publishing content (text, image, audio and video files) with the help of Internet technology and thus is often referred to as *Web* content management. We will define the terms document and content later more thoroughly. It is important to note that the differences lie in the goals and

[11] URL: http://www.mindjet.de, last access: 2005-02-04

can not be fully attributed to the distinction between documents and contents as the boundaries between those terms are blurry.

Sometimes, content management is conceptualized in a broader view as comprising document and Web content management and referred to as enterprise content management (ECM). At first sight, this may be confusing, but it points into the right direction. The management of documents and of contents have much in common and the functions of document and content management systems partly overlap. This is reflected by developments in the marked of standard software for DMS and CMS. Many software vendors integrate content management functionality into their document management systems and vice versa, often by acquiring other software companies and incorporating their products.

An EKI must provide both, integrated support of handling of (formerly paper-based) documents as well as publishing Web contents. However, the distinction between document and content management is still apparent in literature and practice since different challenges and topics are discussed in each discipline. Thus, we will explain the handling of documents and contents separately according to the document and content life-cycle and use the term content management in its narrow sense as Web content management.

4.2.1 Document Life-Cycle

Organizations still store much of their information on paper-based documents. The often-cited vision of the "paperless office" has not been turned into reality. This regularly deters efficient storage, search, access, and archiving of documents. Altogether, the handling of paper-based documents is costly and often inflexible. Document management systems (DMS) promise to overcome these challenges by providing functions for managing documents electronically, not limited to documents that were formerly paper-based, but also including electronic documents in multiple formats. Examples for documents are briefing notes, correspondence, email-messages, memorandums, spreadsheets, and studies as well as multimedia documents such as audio and video files.

Advantages of DMS

Possible advantages of DMS are

- decentralized and simultaneous access by multiple users,
- more flexible categorization and filing of documents,
- higher consistency through centralized data management,
- easier backup through digital copies and
- prevention of unnecessary media changes.

Quantifiable advantages in terms of costs and time are

- faster access, archiving, exchange, search and retrieval of documents,
- improved productivity of document-related tasks, and

- less personnel costs as well as general cost reductions (copies, stocking, etc.).

Definition of document

A document is a (1) legally sanctioned record (e.g., purchase order) or a transitory record (e.g., informal meeting notes) of a business transaction or decision (2) that can be viewed as a single organized unit. (3) It is composed of a grouping of formatted information objects (4) that can be accessed and used by a person and (5) are usually stored on media such as paper (one or set of papers), microfiche or electronic.

Description

(1) The term *record* denotes that the document's context relates to some kind of business transaction or decision which the document represents. There are legal requirements regulating the handling of many documents in organizations (e.g., time period for archival). The term transitory reflects the requirement that many documents are developed step-by-step so that a DMS has to provide functions for versioning. (2) Documents are collections of information objects bound by the document's purpose. (3) These information objects are formatted in many cases. DMS have to conserve the original form of the entire document. (4) Documents are accessed as a whole because they group related information with respect to the (most common) user needs. (5) Last, but not least documents can be stored either electronically or non-electronically. An important challenge for DMS is to handle both storage media effectively.

Document life-cycle

Every document passes through the life-cycle phases of creation, filing, (re-)use, change, and deletion. Ideally, DMS provide an integrated set of functions to support tasks during those phases. These are capture, structure, distribute, retrieve, output, access, edit and archive (Figure 4-9).

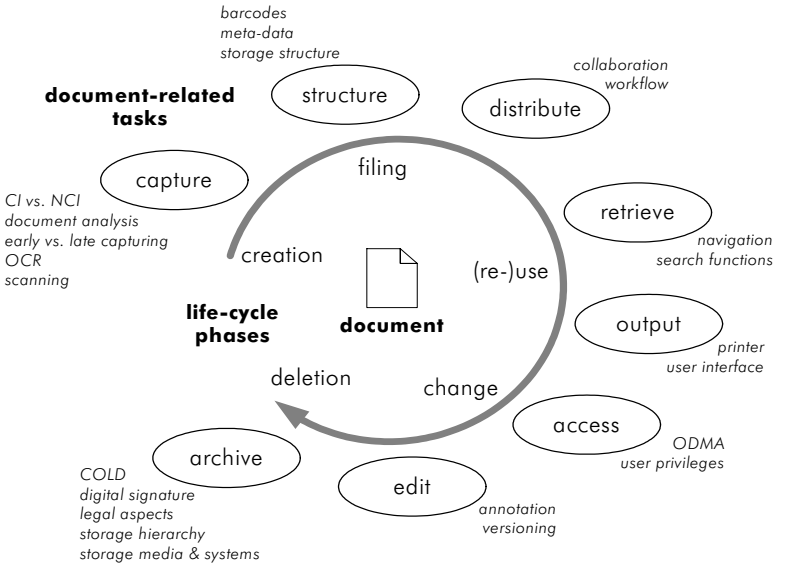

Figure 4-9. Overview of document life-cycle

Definition of DMS

Document management systems (DMS) provide functions to support all tasks related to the management of electronic documents, i.e. to capture, structure, distribute, retrieve, output, access, edit and archive documents over their entire life-cycle of creation, filing, (re-)use, change and deletion.

Products

Examples for DMS vendors are documentum, Saperion and Hummingbird that all offer extensive platforms with different strengths with respect to DMS functions, for integration with other products (e.g., SAP R/3), archiving and compliance with legal regulations.

In this section, we will focus on tasks directly related to the publication of documents. Distribution of documents is supported by workflow systems described in section 3.4.2, 210ff. Retrieving documents is supported by discovery services discussed in section 4.1, 226ff.

Capture. Capturing documents comprises digitizing and storing them in the repository of a DMS. This happens by the following ways:

Ways to capture documents

- *entering* documents manually, e.g., in free-text or with the help of forms, i.e. structured screen masks,
- *sampling* of paper-based documents or microfilms with scanners, if necessary coupled with automatic transformation on the basis of optical character recognition,
- *photographing* or filming documents with cameras,

- *digitizing* audio with soundcards,
- *capturing* and transforming video sequences with video boards,
- *drawing* maps, technical plans or engineering drawings with specialized digitizers,
- *migrating* electronic documents from file servers, hard disks, etc.

Document analysis

Regarding paper-based documents that are candidates for inclusion in a DMS, the appropriateness of capturing and storing them in the system depends on

1. *importance* of a document, e.g., documents crucial for tasks in business processes,
2. *how often* a document is accessed, and
3. whether *legal regulations* prescribe archiving of the document.

Classes of documents

These criteria can be used to shape classes of documents (Table 4-1). Class A documents definitely should be archived, whereas class E documents should be shelved outside the DMS. For documents that fall into the classes B, C and D, efforts for capturing them need to be balanced with possible benefits and with eventually existing legal obligations that prohibit destroying the paper-based documents. This classification scheme is especially useful during the introduction of a DMS. At this stage, regularly large amounts of paper-based documents and micro-films need to be captured and archived. It is often economically most feasible to engage specialized service companies for this task. A part of the paper-based documents needs to be shelved in addition to storing them in the DMS due to legal restrictions (e.g., signed contracts or certificates). Here, the role of the DMS is to provide easy access to the documents rather than replacing the paper-based document archive.

Table 4-1. Document classification for storage in a DMS

class	characteristics	decision
A	documents with high importance and high access rates	store within DMS
B	high importance, low access rates	balance legal restrictions, necessity of safe-keeping outside DMS (paper-based archive), possibility to destroy original documents, and efforts for capturing
C	low importance, high access rates	
D	low importance, low access rates, but safe-keeping legally prescribed	
E	low importance, low access rates, no legal restrictions	shelve outside DMS

Paper documents suitable for capturing

Suitability of paper-based documents for capturing depends on (1) whether the quality of the source document is sufficient to fulfill the requirements for output on screen, on paper and eventually for optical character recognition, (2) on time and effort that is needed for capturing, (3) durability, (4) costs and workload of hardware for capturing and (5) how carefully the paper-based documents should be treated. Typical hardware for capturing are high performance scanners that are able to process hundreds of pages per minute.

Early vs. late capturing

Documents can either be *captured early* when they are created or arrive in the organization, e.g., at the time when they get in by mail. In this case they are scanned, indexed and stored in the DMS directly after arrival. All tasks and changes like annotations, editing and creation of signatures happen electronically with DMS functions. The other alternative is *late capturing*, i.e. scanning, indexing and storing documents not until they are completely processed. Thus, documents at the end of their life-cycle solely need to be archived. During capturing, conformity of electronic documents to paper-based documents is certified by means of digital signatures, especially in the case that sensitive information is captured and stored.

Coded information (CI)

Directly after digitizing documents with scanners, it is necessary to decide how they should be technically represented in the DMS. Depending on how single symbols or characters are internally used to represent document contents, documents can either be represented as coded information (CI) or as non-coded information (NCI). Coded information is structured in a way that enables computers to directly interpret single symbols as units of the information. Interpretation in this case means that characters, words and to a certain degree the document structure are directly accessible and processable for the computer. Examples are plain text, postscript, pdf, HTML, XML-documents and file formats of office suites. Advantages are that it usually requires comparably little space to store CI (about 2,5 -30 kB per document page) and that it is directly and easily processable for computers e.g., for full text search or for calculations. A challenge is that CI may only be accessible with the original application that determines the internal document format. Thus, the application possibly also needs to be archived. For instance, if a document was written and stored with Lotus AmiPro in the year 1997, eight years later in 2005 it is unlikely that it can be easily accessed since there might be no computer available with this application installed. Another disadvantage is that identity with original documents can not be guaranteed as CI not necessarily resembles a direct image of NCI (due to e.g., automatic reformatting). However, standard formats such as the portable document format (pdf), help to alleviate these problems because on the one hand many applications can deal with this formats and on the other hand, it aims at capturing the entire formatting of a document.

Non-coded information (NCI)

In contrast, non-coded information (NCI) is not directly interpretable by computers. This does not mean that no format or code is used to store the information. The difference is that the basic signs that compose NCI documents do not directly resemble the signs (e.g., single characters, words) of the stored information. Examples are documents that are scanned and stored as halftone images (e.g., Bitmaps, TIFF, GIF or PNG). NCI documents are only directly interpretable by humans, at least further transformations like optical character recognition are necessary to transform NCI into CI. Moreover, NCI requires more space for storage (about 0,5 - 8,3 MByte per document page, without compression). Advantages are that NCI documents can easily be captured and that compliance with original paper-based documents can be guaranteed with electronic signatures.

Optical character recognition (OCR)

The term optical character recognition (OCR) was coined for methods to transform documents from half-tone images to characters and therefore from non-coded information (NCI) into coded information (CI). In document management, OCR is applied in two ways:

One alternative is to recognize single document attributes like customer ID, invoice ID, numbers that identify the document type or sequence numbers of workflows. Best recognition rates are achieved if documents have a fixed layout with attributes always in the same position or if attributes are constricted with respect to the range of possible values so that later validations are supported. This is usually the case for capturing standardized forms.

The other alternative is to transform the whole document into CI. This approach is usually chosen if documents later have to be edited or automatically processed. OCR software also recognizes layout of the pages and transforms tables, text with multiple columns and position of pictures that may be included in the document as halftone images. Usually, manual control and reworking is required to transform documents into CI.

Recognition rate

From an economic perspective, the recognition rate has to be as high as possible so that amount of rework is minimized and especially does not exceed the effort to enter the whole document manually. A recognition rate of 99,8 percent might at first glance appear to be very good, but leads to 4 to 8 errors on a page with 2000 to 4000 characters. It depends on the quality of the original paper-based documents that is determined, e.g. quality and color of the paper, font types and sizes, usage of special characters as well as distances between characters, words and lines. Quality of recognition can be substantially improved by applying advanced methods for pattern matching, advanced heuristics, fuzzy-logic and spell checkers that extend the OCR software.

Meta-data

Structure. After documents are captured, additional meta-data is assigned (section 3.2.1, 173ff), which is often called indexing or attribution of documents. Meta-data usually is stored in a data base together with a link to

the document that is transferred to a separate storage medium or system. Typical examples for document meta-data are identifiers for invoices, records or drawings, creation date and the type of document or form. Especially when large amounts of documents need to be captured, it is necessary to automate this task as much as possible. Indexing information can be located and extracted automatically by applying OCR techniques in the case the document is suited for this process. Nevertheless, data often needs to be entered manually, which usually is a time-consuming and laborious task.

Barcodes

Barcodes are one solution to reduce the effort of indexing documents. They are used to store IDs on paper-based documents that indicate e.g., the document type, a customer ID or a workflow step and are especially suited for documents printed by the organization which need to be captured and stored after some changes on the paper-based document. An example is a customer questionnaire with a barcode carrying an ID that links the document to a workflow in the customer support. This document is sent out to the customer, returns after some time, is scanned, automatically linked to the workflow and by this way appears in the inbox of the responsible person.

Access and edit. A key requirement is to manage access to documents as many of them contain confidential and sensitive information that is not open to every employee.

Access privileges

Thus, DMS allow to assign privileges on the level of single objects, object classes or parts of the storage structure for users or groups of users. Typical privileges are to search, access, change, print or delete documents, access and change meta-data or to administer parts of the DMS. If these basic privileges are not sufficient, then documents should be additionally encrypted (section 2.3.4, 124ff). This is especially relevant

- when documents are transferred over electronic networks, especially public networks,
- in case copies of documents are stored outside the DMS where unauthorized users could get access to them, e.g., on hard disks of notebooks, and
- when documents are stored in an archive since even there it can not be completely excluded that unauthorized persons get physical access to confidential information.

Editing and annotation

Ways to edit documents are to change their contents, to change related meta-data or to annotate them. DMS provide several possibilities to annotate documents, e.g, by highlighting fragments of the document or attaching notes to them. An important way of annotation is to electronically sign documents, e.g., for approval in validation steps of workflows. Usually, the content of the document remains unchanged and annotations are stored separately.

Versioning

Versioning mechanisms support handling of different versions of the same document. This is typically necessary to back up older versions of documents, to track changes and to synchronize document versions that evolve when multiple users work together. Basic implementations store every version as a full copy of the document which is identified by a distinct version number. More advanced methods identify and store the differences to older versions which saves storage space and allows roll-back of changes. The latter is only possible if documents are available as coded information and in appropriate formats (e.g., plain text).

Storage structure

The user can access documents either by full-text or meta-data search (section 4.1.2, 232ff) or by navigating through the structure of the DMS repository. The structure usually uses metaphors like “locker”, “folder” or “file”, which are hierarchically organized. This is suited for simple structures that do not contain too many sub-structures or sub-trees and especially when a clear and stable hierarchical structure already exists (e.g., in library systems). The structure should be durable, self-describing, without redundancy, consistent and accepted by users. If more complex structures need to be implemented, documents are accessed by querying their meta-data and selecting them from result lists.

Interfaces and integration

From the user’s perspective, DMS can provide either dedicated interfaces and clients for specialized tasks, e.g., for capturing documents or administering the DMS, or it can be integrated into the user's desktop with the help of the WebDAV protocol, with TIFF / PDF printers (section 5.2.4, 326ff) or by standardized interfaces that allow for a tight integration into end-user applications.

Open document API

In the context of DMS, a typical example with a focus on integration into office suites is the ODMA standard (open document management API). It was defined by the ODMA group, an international association of about 50 DMS vendors and establishes a set of software functions that a DMS needs to provide in order to allow direct access from within end-user applications, e.g., opening and saving a document from within a textprocessor. The main advantage of ODMA compared to proprietary solutions is its independence from concrete DMS products.

Archive. The DMS archive is implemented using a variety of storage media and systems (section 2.3.1, 109ff). Selection of appropriate storage media to implement the archive has to balance access times, costs and durability of media. Usually, integrated storage systems such as RAID arrays, jukeboxes and hierarchical storage management systems are used to provide storage space according to the DMS users’ needs.

Storage hierarchy

Figure 4-10 depicts the storage hierarchy that roughly classifies storage space provided by different types of media according to the criteria availability, access time, capacity and costs into three groups. The importance of a document determines the group of media used for storage. (1) Especially documents with high access rates, information facilitating accom-

plishment of tasks in the context of business processes of the organization and whose provision is time-critical can be considered very important to an organization. Examples are best practices, guidelines, templates and documents of current projects. They should be available online on fast storage media. (2) Documents with medium importance which are accessed more or less regularly can be stored nearline on reasonably priced storage space. Examples documents of recently completed projects or construction plans used for reference. (3) Documents with low importance and especially those that are archived and very seldom if at all accessed can be stored (3) offline on cheap storage space. Examples are documents archived in the DMS only due to legal regulations such as bills or contracts as well as documents of projects completed a long time ago.

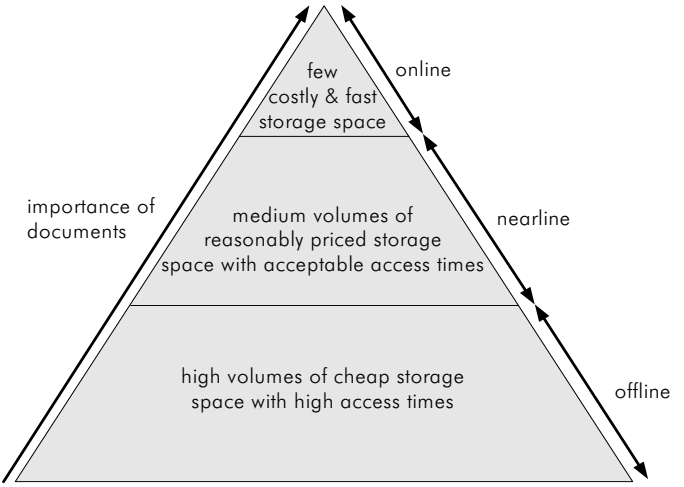

Figure 4-10. Storage hierarchy

Caching

Caching aims at lowering access times while better utilizing expensive storage space. Data that is likely to be accessed twice or more times within a short time period is held in a separate caching memory. It is usually much faster than the original storage media and implemented with fast RAM memory or hard disks. Caching is especially helpful if many users simultaneously access the same data or documents and if it can be forecasted which data elements will be requested from storage media. The amount of necessary caching memory depends on acceptable waiting time and on the amount of data that is processed.

COLD

COLD (computer output to laser disc) denotes the output of electronically created documents on optical storage media. It is used for information that needs to be archived in parallel to its print-out, e.g. to fulfill obligations to store special document classes for pre-defined amounts of time. Examples for document classes that are archived by COLD are correspon-

dence, invoices and lists handled by accounting or human resource management that are usually generated by mainframe systems as large print or report jobs. The typical procedure to archive COLD data starts with transferring the data from application servers to the COLD system which can either be a component of the DMS application or a separate archiving or COLD server. Then, the data stream is decomposed into single logical units, e.g., single documents, and indexing information is automatically extracted for document classification. The indexing information is then stored in a structured data base. The document content is transferred to the archive and saved on appropriate media.

A main challenge is to detect and extract indexing information from the print job. Reasons for this are manifold document and data structures. The DMS should provide sufficient possibilities to analyze and process print jobs produced by different applications. Changes on the side of the source applications often cause high efforts or are not economically feasible at all. Document contents (e.g., values of a form) are often separated and stored independently from their layout and structure (the form itself) as coded information (CI) in order to save storage space. For output on screen or printer, the original appearance of the document is recreated by merging content and layout.

Emails

A recent trend is to include emails since they include more and more legally relevant data. To implement this, DMS vendors include modules for integration with messaging servers like Microsoft Exchange or Lotus Notes.

Legal aspects. Archiving is affected and regulated by a number of laws that aim at orderly, secure and traceable storing of documents that contain relevant data about business transactions. Most regulations are nation-specific, though the European Union strives for harmonization of national laws in Europe. As an example, we will outline the laws that are significant in Germany (based on Glubins et al. 2002).

Trade and tax law

Trade and tax law (in German: Handelsgesetzbuch (HGB), Grundsätze ordnungsgemäßer Buchführung (GoB), Abgabenordnung (AO)) specify general standards and procedures of bookkeeping. They regulate which types of documents have to be archived for what time period. Storage on electronic data carriers is explicitly permitted, though the term “data carrier” is not clearly defined. Additionally, general requirements for accounting need to be considered in document management. These are accurate and complete recording of all accounting transactions that underlie the legal obligation to keep records, timely recording, ordered representation and safekeeping of the data during the whole prescribed time periods. For audits, an electronic archiving system always must be able to present all necessary records with adequate effort, promptly, in the required sequence and anytime over the period of time the organization is obliged to be able to show supporting documents.

Civil and criminal law, signature law

Civil and criminal law (in German: Zivilprozessordnung (ZPO), Bürgerliches Gesetzbuch (BGB), Strafgesetzbuch (StGB), Strafprozessordnung (StPO)) affect document management with respect to the code of civil procedure and especially with respect to validity of electronic documents for evidence. Law explicitly demands original documents, e.g., financial statements, annual accounts and invoices since only these documents are suited as valid evidence. Thus, document management has to secure and to make verifiable that (a) the electronic document matches the original paper-based document and (b) changes on the electronic document are always detectable. Digital signatures are one way to implement this and to fulfill these requirements (section 2.3.4, 124ff). Signature law (in German: Signaturgesetz) aims at creating the basic conditions for electronic signatures and defines organizational and technical requirements for certification authorities (CAs) that issue qualified electronic signatures and provide the constituting public key infrastructure (PKI). Part of a certificate are the name of the signature key's owner, the assigned public key, the name of the signature algorithm and hash method, certificate number, begin and end date of the period in which the certificate is valid, name of the issuing certification authority, further data about cases in which the certificate may be used, specifications defining the security level of the signature as well as optional data about the owner of the certificate.

In 2001, the German ministry of finance detailed requirements with regard to the verifiability of digital records (Grundsätze zum Datenzugriff und zur Prüfbarkeit digitaler Daten im Unternehmen, GDPdU). Auditors are explicitly granted access to tax-relevant data and are entitled to get electronic copies of them, e.g., to analyze and process them automatically. Electronic originals like electronic invoices have to be archived and relevant data must always be extractable from accounting systems and electronic archives.

Data protection law, copyright law

Other important regulations are data protection laws (in German: Bundesdatenschutzgesetz (BDSG)) that regulate data security and protection of personal data, copyright law (in German: Urheberrecht, UrhG) for documents secured by copyright, the works constitutions act (in German: Betriebsverfassungsgesetz) for co-determination during the introduction of a DMS as well as additional laws dependent on specific industries (e.g., emission laws regulating required documentation of machinery in plants).

Basel II

An international regulation that influences archiving is Basel II. It will be effective in 2006 and regulates the conditions of borrowing capital from financial institutions. For that, organizations need to undergo a rating that is generated on the basis of key financial data. This data is stored in accounting systems, management information systems and DMS. A recent trend in the DMS market thus is to make DMS GDPdU and Basel II conformant.

Code of practice

The association of organizational information systems (Verband Organisations- und Informationssysteme, VOI), an alliance of leading document

management vendors, has developed guidelines for archiving conforming to tax and trade laws. Ten maxims guide proper handling of documents according to the principles of electronic accountancy systems, which are called *code of practice*. These principles clarify the fundamentals that have to be regarded in relation to legal laws and regulations when implementing a DMS:

1. Every document that is archived has to remain unchanged.
2. Every document must not get lost on its way to the archive or in the archive.
3. Every document must be locatable with adequate retrieval techniques.
4. Exactly that document must be found that has been searched for.
5. A document must not to get destroyed over its entire life time.
6. Every document needs to be displayed and printed in exactly the same format it has been captured.
7. Every document has to be recovered promptly.
8. All actions that change the organization and structure of the archive have to be logged so that its original state can be reconstructed.
9. The construction of electronic archives has to enable a migration to new platforms, media, software versions and components without loss of information.
10. The system has to provide functions that secure compliance to laws as well as organizational regulations regarding data security and life time of the archive.

4.2.2 Content Life-Cycle

In this section, we will discuss another aspect of publishing - the publishing of Web content that is supported by Web content management systems (WCMS). Today, WCMS are widely used and many different products and tools are available on the market. Based on the definitions of content, asset and WCMS we explain the main functions of WCMS according to the content life-cycle. Then, four generations of WCMS are discussed with the fourth generation giving an outlook on future developments. The section is concluded with a case example of introduction and concrete technical implementation of a WCMS at the company E-Plus.

Life-cycle phases. Figure 4-11 gives an overview of the content life-cycle and the services that a WCMS offers to support activities regarding collection, publication and management of Web content.

Definition of content

Content is a set of meaningfully arranged electronic data that is separated from its layout which describes how it is rendered during reproduction.

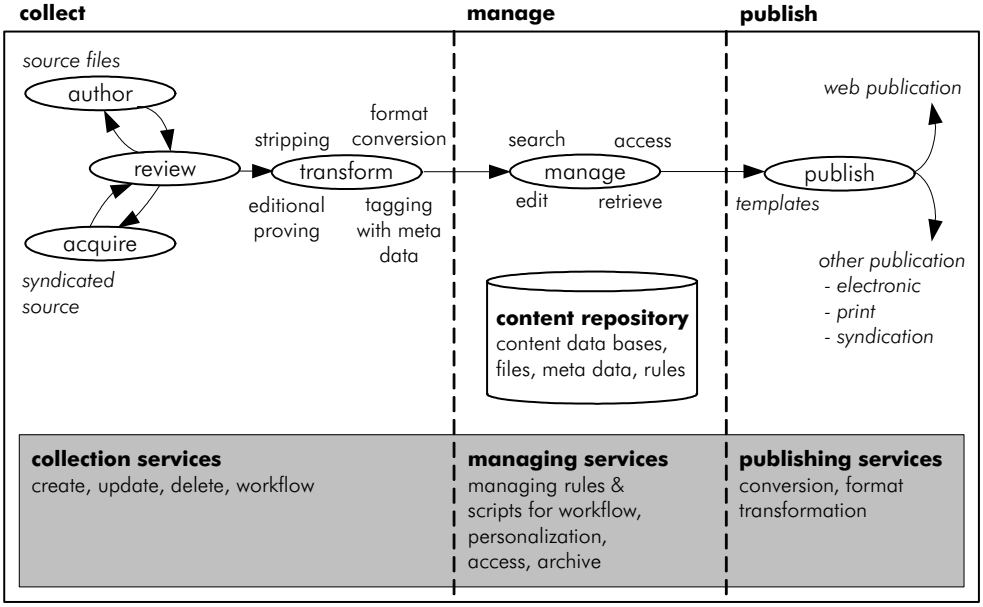

Figure 4-11. Web content life-cycle (based on Boiko 2002)

Collecting content

Content is either authored from scratch with tools like office applications, editors or extensions of WCMS, or it is gathered from an existing source, e.g., content that is commercially available content syndications or other source files from within or outside the company. To assure the quality of contents, it is reviewed by, e.g., supervisors of the authors or experts in a special domain. If changes are necessary, it is transmitted back to the authors. Eventually, multiple iterations of the authoring/acquiring and review cycle are necessary for the content to meet the requirements. Then, content is transformed in a format that can be handled by the WCMS and stored in an electronic repository. This comprises stripping of unnecessary parts of contents (e.g., headers, footers), file format conversion, customizing of the structure, editorial processing to improve readability and consistency of the content, annotation with meta-data (section 3.2.1, 173ff), segmentation to divide the content into useful chunks and creation of versions, e.g., of pictures of differing size and quality for different channels. This is a major step required to transform raw contents into reusable assets.

Definition of asset

Asset denotes a subset of content that is annotated with meta-data (context) and thus can be readily retrieved and (re-)used. The term emphasizes the value that it represents for the organization.

Managing content

Managing content comprises searching, accessing, retrieval and editing content as well as the definition of rules for management and publication. A content repository contains structured content data bases (e.g., within relational or XML object data bases), files (e.g., media files), control and configuration files (e.g., log files, indexing files), meta-data, templates and rules (e.g., access rules, workflow definitions). The repository is the core of a WCMS and a foundation to publish content. The term repository is used here in a broad sense to denote the WCMS inventory of both, content and meta-data describing the content and its appearance. It must not be confused with repositories used for integration of catalogues of organization-wide meta-data.

Publishing content

Contents are created by collecting them together with other resources from the content repository and automatically creating different types of publications. Templates are a central element for this process as they contain rules and information about how to transform unformatted content into the right formats, e.g., to create HTML out of the structured data base for publishing Web content. Content management is not necessarily limited to creating Web content. Any other format can be generated from the content in the repository. This can be print publications (e.g., generation of Adobe Acrobat PDF files) or other electronic publications (e.g., CD ROMs, Microsoft Help files, PDA formats, syndications).

Definition of WCMS

Web content management systems (WCMS) are applied to efficiently handle all electronic resources required to design, run and maintain a Web site and support all tasks related to authoring, acquiring, reviewing, transforming, storing, publishing and delivering contents in Web formats.

Comparison of content and document life-cycle

Content and document life-cycle are similar as they comprise processes of creation, access and archiving of documents and contents. In principle, it would be possible to create a life-cycle that integrates both, the life-cycle of documents and of contents. However, a major difference is that content management deals with processing, customizing and publishing of semi-structured information that are arranged and formatted so that they are delivered to and navigated by selected target groups, e.g., employees, customer segments or partners, while document management originally aims at capturing, handling and archiving documents internally in an organization. The focus of Web content management is on workflows of editing and publishing, on formatting and presentation compliant to corporate design as well as on structuring of, navigation of and interaction with Web sites. WCMS aim at making publication of Web contents according to predefined, often complex layouts and styles possible without technical knowledge, in a timely manner and with low effort. In contrast, DMS focus on capturing, describing, versioning, accessing and archiving of digital documents compliant with legal regulations (section 4.2.1, 247ff).

Thus, different challenges and issues are predominant and both life-cycles are discussed separately.

Publishing with and without WCMS. Function and advantages of WCMS become more evident when traditional Web publishing without WCMS and Web publishing with WCMS are compared (Table 4-2).

Table 4-2. Web publishing with and without WCMS compared (Zschau et al. 2002)

	traditional Web publishing	**Web publishing with WCMS**
format	content and layout mixed	content and layout separated
publishers	only Web team or Web master publish content, danger of bottleneck	all employees are able to publish content
effort	publishing effort is high and quickly grows with the size of the Web site	initial effort for introduction of CMS is high, publishing effort is low and slowly grows with the size of the Web site
publishing workflow	manual publishing workflow, e.g. based on emails	automated publishing workflow supported by WCMS
tools	multiple different tools needed to manage Web site	CMS provides integrated set of functions, no additional tools necessary
redesign	redesign of Web site is a costly reprogramming process	easy redesign by changing templates

Traditional Web publishing

Contents that need to be published on an organization's Web site are usually authored and illustrated by many different persons. Most of them do not (and do not need to) possess skills to create complex HTML pages formatted according to the organization's corporate Web design, embed resources, place them on a Web server and link them to other pages of the organizations Web presence. In the case of traditional Web publishing, Web masters are responsible for technical management of an organization's Web site which often includes administration of software as well as hardware. They or specialized Web publishers take over the task of transforming contents to be published into the right format and a consistent layout. The danger is that these employees become a bottleneck if many changes have to be made or much new content needs to be published. In order to cope with this situation, authors wanting their contents to appear as soon as possible thus are inclined to acquire technical skills to create content on their own which usually breaks the uniform appearance of the Web site because they will knowingly or not-knowingly develop their own ideas about layout and formats. The management of the Web site involves many different tools, e.g., converters for image formats, link-checkers, content validators and so on. Publishing Web content is an error-prone process, especially if the Web presence grows up to many hundred pages.

Major changes, especially applying a new layout to refresh the appearance of the Web site are costly and time-consuming and regularly have to be accomplished by specialized companies.

Web publishing with CMS

In the case of Web publishing with a CMS, a centralized system enables every author to publish content directly without the help of a Web master or Web designers. Thus, employees with skills to transform and publish the content are not any more a bottleneck. Layout information is separated from content and shifted to so-called templates. Changing the site's appearance is merely a matter of altering layout templates. Consistency of the Web site's layout is higher and less errors (e.g., broken links) occur since the system has built-in functions to check this. WCMS support the publishing process and automate tasks involved in publishing content on the Web (e.g. transforming content to different formats, link-checking). Thus, the effort to maintain even large Web sites remains low. Publishing and access to contents is coordinated by workflows and user privileges. Overall, time to publish contents on the Web site is substantially shortened.

Components of WCMS. We will discuss the core components of a WCMS according to three content life-cycle phases: collect, manage and publish. In addition, WCMS have functions to support workflows as well as administrative tasks throughout all three phases.

Functions for collecting

A common way to support authoring are HTML-based forms where authors can type in text, upload images and add meta-data in a structured way. They are usually structured according to pre-defined content types (e.g., menus, news, articles) and allow for tight control of the authored content. Each content type has a defined appearance, e.g., in terms of size, position and format. Some systems present the resulting Web pages in an so-called *author view* with edit-buttons on each structure element so that authors are able to directly choose items on the page that have to be changed and directly "type into" the Web site. The content is then stored together with its type and other meta-data (e.g., author, creation date, expiration date) in the structured content repository. An alternative way is to create content outside the WCMS with other applications such as text processors. In this case, the content is firstly structured with the help of templates (e.g., Microsoft Word .dot-files) that can be complex with fill-in fields, validation macros and direct connections to the WCMS repository. Then, contents are transformed to be stored in the structured repository. The WCMS can be integrated into the external applications, e.g., by using WebDAV or other ways of integration so that in a user's perspective, the contents can be directly stored within the WCMS.

Syndication

Syndications are sets of contents that are published for distribution and reuse in other publications, regularly by specialized content providers. An example inclusion of news from specialized news companies. The customer of a syndication service usually includes statements into the Web

site that request pre-defined content elements from the syndication service supported by the content provider. Most useful and widely spread are XML-based markup languages such as RSS (RDF site summary, older versions are sometimes called rich site summary), NewsML and ICE (information and content exchange) which support communication between the requesting site and the syndication service. Content elements are usually is transformed and edited before they are published.

Listing 4-1 shows an example for two simple JavaScript statements for syndication to include content from the German research journal Wirtschaftsinformatik in a Web site. Type of content is specified by the parameter `op` (`article` for announcements regarding publications in recent issues of the journal and `news` for latest news), number of entries is defined by `count` and the maximum length in lines by `maxl`.

```
<B>News from WI-Online:</B>
<!--
  <SCRIPT language=javascript src="http://62.52.24.38/wi/
  wi_syndicate.php?op=news&count=10&maxl=100"></SCRIPT>

 <SCRIPT language=javascript src="http://62.52.24.38/wi/
   wi_syndicate.php?op=article&count=5&maxl=150"></SCRIPT>
-->
```

Listing 4-1. Example for syndication statement

Functions for managing

The content repository is the core of a WCMS and stores all necessary elements in a structured content data base. Additionally, it provides interfaces for other systems to import structured data directly into the repository, e.g., from other relational data bases (bulk loading), to include multimedia files, enable content syndications, etc. in order to transform and process data from transaction processing systems, the WCMS can offer APIs or application server can be included into the overall WCMS architecture, e.g., to connect systems to e-business platforms with standards like CORBA, COM, EJB or JSP (section 2.4.2, 139ff).

Templates

Templates govern the process of transforming raw content into the desired layout or format of publication. Two types of templates can be distinguished. Input templates support authoring of content and publication templates are required to produce output pages. A publication template consists of layout elements (e.g, HTML tags) and program instructions that control where and how the structured content is rendered which is selected from the content repository. Templates are responsible for rendering defined layout areas of the Web page, e.g., menus, tables, navigation elements. A Web page is usually rendered with the help of multiple templates in a cascading process top down from a main template that defines

the overall structure of the page to templates that render atomic layout elements.

Asset management

As noted earlier, the repository contains all services that support managing content. This is often denoted as asset management to emphasize the value of the structured content for the organization. Besides services for versioning, check-in/check-out, control of access privileges, WCMS can provide services for personalization (section 5.1.2, 316ff) and for handling contents in multiple languages.

Functions for publishing and staging

Two server architectures can be distinguished with regard to how changes affect the published Web content. Either the WCMS installed on a server where all changes are directly published which is especially suited for fast changing and content. Many WCMS render the content dynamically in the moment it is accessed by readers. This leads to the danger of producing high loads on the server and may lead to long response times. The other alternative is to prepare contents on a WCMS installed on a separate staging server. Within defined time intervals, contents are automatically exported from the staging server to another server that provides the Web pages. Thus, the collection and managing of contents (content management application) is separated from the actual publishing of contents (content delivery application). Advantages are that the WCMS can be operated, managed and accessed in a secure environment while the dedicated Web server can be accessed by external users and thus is more exposed to security threats.

Administration

Administration comprises management of user roles and privileges, maintaining meta-data and content structures, backup, archiving, defining workflows and last but not least ensuring that all hardware and software is advisable for publication and access to the Web site.

Workflows

Workflows are used to automate chains of tasks during collection, managing and publishing (for details about workflows see section 3.4.2, 210ff). Concerning collection, control creation, review and approval tasks. Concerning managing content in the content repository as well as administration, workflows are applied to automate tasks such as backup and archiving, synthesis of content, notifications or managing connections between WCMS and other systems. Concerning publication, workflows are used for quality assurance and update tasks.

Generations of WCMS. The wide-spread use of the WWW as information platform can be dated back to the early 1990s and soon it became clear that publication of Web content needs to be supported by appropriate tools. Table 4-3 gives an overview over the four generations of WCMS that can be identified.

Table 4-3. Generations of WCMS (based on Zschau et al. 2002)

	first generation (mid 90s)	**second generation** (early '00s)	**third generation** (mid '00s)	**fourth generation** (outlook)
	Web site administration	*Web publishing*	*dynamic Web publishing*	*enterprise content management*
target group	technical experts manage Web site	non-technical authors are able to publish Web contents	WCMS open to all employees	WCMS open for internal and external information providers and sources
meta-data	no meta-data	pre-defined meta-data	customizable meta-data	semantic meta-data
navigation	manual structuring of contents	automatic generation of navigation paths	basic personalization and profiling	context-dependent, dynamic navigation
rights management	no user privileges	basic user privileges	standardized user management	integration with organization-wide identity management
workflow	manual workflow	pre-defined workflows	customizable workflows	ad-hoc workflows and collaboration functions
flexibility	static contents	basic functions to manage flexible contents	advanced functions to support dynamic content	integration of multiple internal and external information sources
integration	no integration with existing systems	application integration with APIs	connectors to most important Web applications	WCMS part of an enterprise knowledge infrastructure

First generation

Strictly speaking, the first generation includes the precursors of WCMS which are tools to make Web site administration and structuring more efficient. The focus is easing publication of larger amounts of contents and on introducing regulations and functions for structuring Web sites. Relational data bases were filled with content and used to automatically generate HTML pages. Basic Web site management tools supported administration of the structure of a Web site. Authors needed to write pages in HTML, layout was not standardized or separated from content. Systems were usually implemented in server-side script languages like Perl. Access to Web contents was administered with Web server functions and there were usually no functions to control privileges of authors or to manage additional meta-data.

Second generation

Second generation WCMS introduced many functions that aimed at making the publishing and transformation processes of Web contents more efficient, e.g., functions to automatically transform content to Web formats, generate elements for navigation through a Web site (e.g., table of contents, site maps, etc.) and to guarantee consistency of the Web site (e.g., link checking). Through separation of layout and content, technical Web publishing skills were not necessarily needed any more and more authors were enabled to create and publish Web content. Functions to administer user privileges and built-in publishing workflows to make publishing more effective were supported by WCMS. In addition to contents, pre-defined meta-data could be stored to structure Web site content and automate handling of content, e.g., by storing information about the publishing period and automatic removal after expiration.

Third generation

The third generation represents the current technical state of advanced WCMS. Customizable meta-data and workflows more flexibly support publication processes and handling of dynamic contents. WCMS provide interfaces for integration with other systems and with user desktops. Customization (e.g., of the appearance of the Web site) and personalization and user profiles are supported (section 5.1.2, 316ff) and allow visitors of a Web site to individualize appearance of the page and the contents presented. Through better usability and advanced user management, Web publishing is more and more open for all employees of the organization, not only for designated authors.

Fourth generation

The fourth generation gives an impression of how WCMS might develop in the near future. A major trend is integration with other systems that support publishing and collaboration around contents. In particular, WCMS are more and more integrated with DMS to facilitate handling of other formats, content with various life times and archiving. They need to provide standardized interfaces for integration with other systems and multiple information sources from inside and outside the organization. Support of Semantic Web standards (e.g., RDF(S), OWL, section 3.2, 171ff) and enriched meta-data is a foundation for semantic links to other Web contents and to flexibly handle large amounts of contents. Advanced functions for user profiling as well as enhanced categorization and personalization enable context-dependent and dynamic navigation. WCMS are used to create Internet sites of organizations, E-Business market places and Community spaces. Recently, two specialized forms of WCMS are widely discussed: wiki books and Web logs (shortly called blogs). Wiki books aim at collaborative and fast publishing of Web content, while Web logs aim at implementing Web-accessible personal “diarys” with the time bar as the main structuring unit.

Case example: Web content management at e-plus[12]. E-plus operates of telecommunication networks for cellular phones with 8,7 million cus-

tomers in 2004. A considerable share of turnover is generated with non-voice services like SMS, MMS and i-mode.

E-plus needs to supply information over multiple channels, especially Voice, WWW, SMS and imode. The information channels should amend each other, e.g., SMS for notifications about a topic and Voice or WWW for more detailed information. Data from content providers like Onvista (market information) or Interactive Media (sports news) need to be integrated (content syndication) and flexibly managed, e.g., to deliver specialized content for events like pop concerts or sports events. Customers should be able to personalize e-plus' offerings by selecting information channels and using a personal calender as well as email access. To supply information over multiple channels, content, functions and layout need to be clearly separated.

In 2002, e-plus applied the system Vignette[13] to support editorial processes and the conversion of XML data that was delivered from content providers. At this stage, employees were able to easily change and add contents. Authored and syndicated content was stored in separate Oracle relational data bases. In the second stage, WCMS and Oracle data bases were merged, but personalization still was implemented through rules and instructions within additional Java Server Pages (JSP). In the third stage, integration between WCMS and LDAP server was improved to control and edit personalization information as well as communication between Web server and WCMS to allow the creation of user profiles. Moreover, JSP were now directly managed within the CMS supported by workflows.

Judging the benefits of this WCMS initiative is not an easy task as they are often qualitative and thus not directly measurable. Examples for qualitative benefits are convenient publication of Web contents, efficient information distribution, better information quality, and more consistent Internet sites. Nevertheless, benefits such as less time to introduce major changes of the site or to publish content can be quantified. Incurred costs were licensing costs for software, costs for hardware and to the biggest share costs for external consultants that accompanied introduction and deployment of the WCMS.

4.3 Collaboration Services

As apposed to publication services focusing the documentation of already explicated knowledge, collaboration services focus creation of knowledge and directly exchanging knowledge between people. Collaboration is com-

[12] based on Christ (2002)
[13] URL: http://www.vignette.com, last access: 2005-02-04

monly divided into communication, coordination and cooperation. However, as this book focuses weakly structured processes and coordination focuses well-structured processes these are only briefly touched in section 3.4.2, 210ff.

Section 4.3.1 discusses theoretical models to structure communication as much as possible. However, applicability of these models is limited with respect to communication processes taking place in knowledge work. Concrete communication tools are classified according to time/place. Section 4.3.2 discusses basic foundations of computer-supported cooperative work and systems that support cooperation between knowledge seekers and knowledge providers.

4.3.1 Communication

Generally, communication is the bidirectional exchange of data. Communication between humans can be mediated by technical systems with the goal to bridge distances or enable for more effective communication (e.g., of large amounts of data).

Formalizing communication. To support communication with the help of electronic systems, it is necessary to understand how communication happens, how it is structured and which special issues need to be regarded. We will present two approaches that provide insights in these areas: The general model of communication by Shannon/Weaver helps understanding how information basically is transmitted. Speech act theory is a way to structure and model communication processes between humans.

Communication model of Shannon/Weaver

The roots of the general model of communication (Shannon/Weaver 1949) can be traced back to information theory and cybernetics. It was initially essentially mathematical and intended to be applied to technical problems under clearly defined conditions. Later, it was generalized to human communication. The model proposes that communication always must include six elements (Figure 4-12):

- a *source* that is an entity with a purpose, a reason for engaging in communication,
- an *encoder* that expresses the sources purpose in form of a code contained in a message,
- a *message* that is transferred with the help of
- a *channel* that transfers the message and can be interfered by physical noise like damping in an electric cable,
- a *decoder* that retranslates the code,
- and a *receiver* to whom the message is addressed.

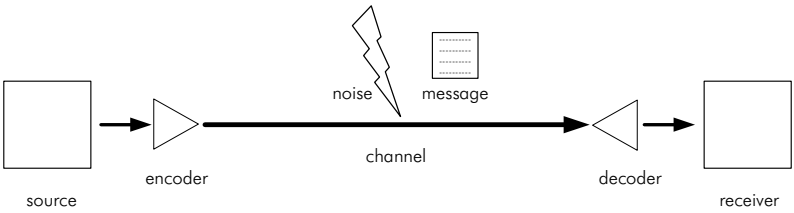

Figure 4-12. Components of the Shannon/Weaver model

One of the initial purposes of the model was to research the role and effects of physical or mechanical noise. It was not primarily intended to study the way how meaning is represented in and transferred by messages, which explains why the model says nothing about how meaning is created by the sender and interpreted by the receiver. Many other approaches and models are based on it. Its value is that it provides a framework and terms to structure and to describe communication processes, e.g., the possibility of a mismatch between encoding and decoding devices.

Speech act theory

Speech act theory was developed in the 1960s to model human language. It analyses speech as meaningful acts by communication partners in situations of shared activity, how speech affects the future of speakers and listeners and especially which actions are caused by words. It emphasizes the activities that are related to speaking in contrast to questions about how messages are transferred and disturbed by noise in Shannon/Weaver model, or how words transport meaning in traditional linguistic theories.

Speech act theory divides human communication into speech acts that are verbal units that can be a word, a sentence, a paragraph, or even an entire document (Austin 1962, Searle 1969). It investigates the meaning speech acts can carry and the types of actions related to them. A speech act can be divided into three parts: (a) *sentence content* that specifies the issue of communication, (b) *category* that classifies the intention of the speaker, and (c) *presentation* that specifies how the verbal units are presented to the listeners (e.g., polite, aggressive, determined).

The category classifies the intention of the speech act as

- *assertive*, i.e. the speaker makes a statement about something (e.g., "Vulcans are characterized by pointed ears"),
- *directive*, i.e. the speaker wants to make the listener do something (e.g., "Could you please hand me the pencil?"),
- *commissive*, i.e. the speaker makes a commitment about something by the speech act (e.g., "I will drive you at home"),
- *declarative*, i.e. the speaker declares concordance between sentence content and reality (e.g., "I hereby declare you husband and wife"), and
- *expressive*, i.e. the speaker describes a current psychological state (e.g., "You did a good job").

The articulation of a verbal unit is referred to as *locutional act*, its intention as *illocutional act* and the effect of the communication on the behavior of the listener as *perlocutional act*.

Conversation network

Winograd and Flores (1986) applied speech act theory to short conversations with the goal being support with ICT. Structures of conversations are described in a conversation network by interlinked speech acts. Arrows represent speech acts and circles indicate the state of the conversation. Conversation networks thus are a state transition diagrams. Conversation networks are modelled on the type level, i.e. the actual content of utterance is not prescribed and depends on the actual context. It can have one or more final states where no further acts are expected by the conversation partners. All acts are linguistic, i.e. they represent a communicative action. The conversation network does not specify what a person should do or what consequences a speech act may have.

Figure 4-13 shows an example of a conversation between a customer and an employee at the cash desk of a supermarket. The customer (A) comes to the cash desk with the intention to pay for her shopping. The cashier's (B) answer depends on the actual situation. The customer could either tell the amount of money she needs to pay, that the cash desk is closed or that she has forgotten to weigh the vegetables and needs to return. If the cash desk is closed, the customer can either ask the cashier to allow her to pay anyway at this desk or she can decide to go to another desk. When the customer knows how much she has to pay, she can hand over the money or realize that she has not enough money. If the cashier has received the money, he thanks and says goodbye to the customer.

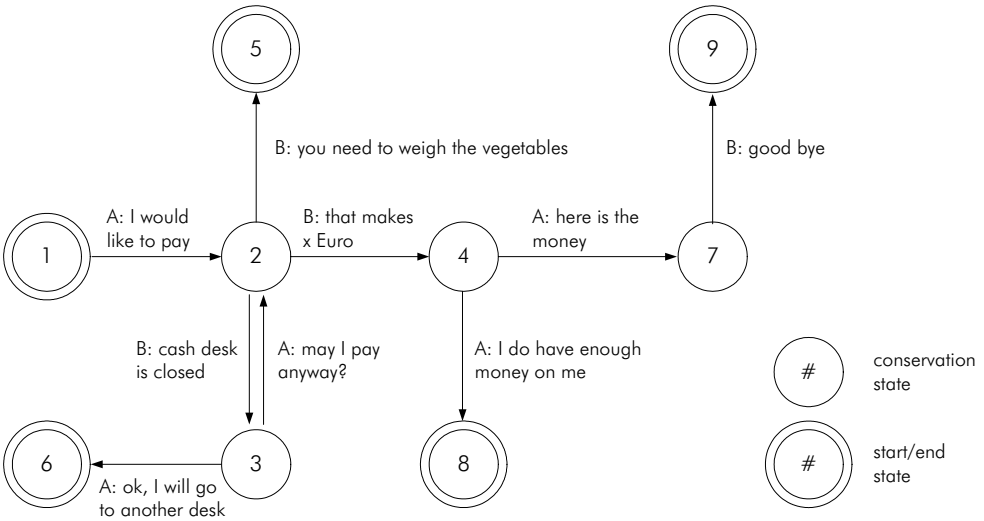

Figure 4-13. Example for a conversation network

Conversation in groups consist of multiple sub-conversations between its members and thus of multiple possibly interwoven conversation networks. Conversation networks can be used to structure and organize communication processes in groups, e.g., with the help of structured workflows or software functions like group calendars that implement a predefined conversation network. An example for a standard language that defines how speech acts on a syntactic level is KQML (knowledge query and manipulation language). It is used to structure communication between software agents (section 1.5.1, 49ff).

Communication modes. Communication can be classified into different communication modes according to criteria such as time, place and number of involved participants. Each mode requires different tools to be supported.

Time/place taxonomy

A common way to classify communication modes is the time/place taxonomy. It distinguishes four modes of communication depending on whether participants are located at same or at different places and whether communication happens synchronously at the same time or asynchronously at different times. Table 4-4 shows these modes together with an example.

Table 4-4. Time/place matrix of communication modes

	same time	**different time**
same place	(I) same place / same time, example: direct conversation	(II) same place / different time, example: blackboard
different place	(III) different place / same time, example: phone call	(IV) different place and time, example: (electronic) mail

Number of communication partners

Additionally, communication modes can be further distinguished according to how many communication partners are involved on each side into one-to-one (e.g., conversation), one-to-many (e.g., broadcast message) and many-to-many (e.g., meeting) communication.

Information richness and hardness. To support knowledge work with EKI, not only the mode of communication mode and the number of communication partners is of interest, but also richness and hardness of the communicated information.

Information richness

Information can be classified into different categories of richness according to its ability to transport the sender's context (Table 4-5). In an organizational setting, the need for lean vs. rich information depends on the actual tasks that need to be accomplished. Complex, multifaceted problems regularly need richer information that clearly structured and

recurring tasks. For example, executives regularly need context-rich information to interpret situations and avoid misinterpretations. Thus, they often prefer face-to-face and phone communications to reach decisions.

Table 4-5. Information richness

richest				**leanest**
face-to-face	phone	personal documents (letters, memos)	impersonal written documents	numeric documents

Information hardness

Table 4-6 shows a classification of information according to their hardness and compares them with the hardness of minerals. Depending on the hardness level of the information that needs to be communicated, humans prefer different ways, systems and media for communication.

Table 4-6. Information hardness (based on Watson 2003)

mineral	**scale**	**example**
talc	1	unidentified source-rumors, gossip, and hearsay
gypsum	2	identified non-expert source - opinions, feelings, ideas
calcite	3	identified expert source - predictions, speculations, forecasts, estimates
fluorite	4	unsworn testimony - explanations, justifications, assessments, interpretations
apatite	5	sworn testimony - explanations, justifications, assessments, interpretations
orthoclase	6	budgets, formal plans
quartz	7	news reports, non-financial data, industry statistics, survey data
topaz	8	unaudited financial statements, government statistics
corundum	9	audited financial statements, government statistics
diamond	10	stock exchange and commodity market data

Traditional dispositive systems deliver information and explicit knowledge with a low level of richness and a high level of hardness. However, decision-making regularly needs additional, soft information and implicit knowledge since predictions, explanations and even opinions or speculations are important sources to assess the value of information, to reduce complexity of decision processes and to develop assessments. Effective decision making thus needs to be supported by multiple types of information, lean/hard information as well as rich/soft information and knowledge

that are prepared for specific work situations and contexts of information and knowledge work.

Systems to support synchronous communication. Synchronous communication can be supported by a variety of media. At this point, we give a short overview of the most important communication systems that are part of an EKI.

Phone

Phones are still the most important communication media. for one-to-one voice communication. Visual phones today have no considerable market share, though their technical implementation has been solved years ago and they have been offered on the market for quite some time. Recently, IP telephony has gained increasing attention in many organizations. It denotes a set of facilities for synchronous exchange of voice, fax and other forms of communication using the Internet Protocol (IP, section 2.1.3, 88ff) that have traditionally been carried over circuit-switched connections of commercial or government-owned telephone networks (PSTN). Advantages of IP telephony are that it avoids the tolls charged by private or public owned telephone companies and that it can be run on the same network infrastructure as other network technologies. Disadvantages are that IP telephony over public packet-switched networks may have a low quality as it is not ensured that data packets arrive in an acceptable time-frame. Better service is possible with private networks managed by an enterprise or by an service provider.

Teleconferencing

Teleconferences connect multiple participants. Usually, this happens either by specialized conferencing systems that all participants connect to by dialing a conference number or by one participant that step wise calls up all participants and connects them to the conference. Communication needs to be conducted in a disciplined way since there are no visual clues that otherwise are used for coordination of communication in, e.g., face-to-face meetings. Teleconferences are often supported by application sharing systems where all participants access an application over a common interface (e.g., text processing documents, spreadsheets), by electronic whiteboards for joint drawings, electronic storage spaces for distribution of documents as well as tools for moderation and to coordinate communication (e.g., by a system that allows issuing the right to speak). The term teleconferencing covers text-based and, audio and video conferencing.

Video conferencing

Video conferencing systems enable interactive, bidirectional communication based on voice and video signals. It provides a rich communication channel which is expected to be a foundation for people to form impressions of others and to communicate more effectively. However, up to now the technology could not live up to users' expectations. Reasons may be high costs and nevertheless limited nonverbal and social communication. Professional video conferencing systems transmit voice and video with high quality, but usually cause high costs. Desktop video conferencing is a less costly alternative where personal computers extended by a Web cam,

a microphone and a software client are applied. Voice and video quality are substantially lower due to limited network bandwidths of public networks. Standards that define network communication protocols for video conferencing are H.320 (for conferences over ISDN) and H.323 (for IP-based networks). An example for a software product is Microsoft NetMeeting.

Chat

Chat enables multiple persons to communicate in text-based dialogues that take place in so-called chat rooms or channels. These channels are usually dedicated to selected topics. Users log-on anonymously using a nickname they are free to choose. Chat systems offer many additional functions around text-based communication, e.g., one-to-one communication outside chat rooms, file transfer, encrypted data transfer and listings of basic data about users. The channels are hosted at a centralized chat server or on a network of multiple interconnected servers. Users need client software (e.g., Java applet that is executed within the user's browser or client application) to connect to those servers. Many public chat networks like Efnet, Undernet, IRCnet, DALnet or NewNet are based on the Internet Relay Chat (IRC) protocol.

Instant messaging

Instant messaging (IM) is focused on one-to-one communication via text-based messages and best suited for short, informal communication. Users need to install a (often freely available) IM client on their local computer and create an IM account on one of the servers operated by an IM provider. Many IM-networks are implemented based on an assisted peer-to-peer architecture: centralized servers support lookup of peers and their current status. Exchange of data happens directly between peers. One reason for the popularity of IM are awareness functions that notify about the online/offline status of communication partners enlisted on the users individual contact list. Thus, they allow for monitoring current online/offline status of communication partners and enable ad-hoc, informal communication. There are multiple competing IM standards, clients and IM networks from different companies like AOL, Mirabilis (ICQ), Microsoft and Yahoo. Recently, steps are undertaken to unite these standards. XMPP (Extensible Messaging and Presence Protocol) is an XML-based instant messaging standard and might be a first step to an uniform Internet standard for IM.

Functions of operating systems

Operating systems usually offer basic functions to exchange one-to-one or broadcast messages (e.g., a warning that a server needs to be restarted). Examples are "net send" on Windows platforms or "talk" on Unix-based operating systems.

Systems to support asynchronous communication. EKI can include a rich variety of systems for asynchronous communication. The most common and important are email, list servers and discussions/newsgroups. Other ways to support asynchronous communication are specific Group-

ware functions like shared information spaces which we will describe later in section 4.3.2.

Email

Email is one of the most wide-spread used applications of the Internet and supports one-to-one or one-to-many communication. Today, email client programs are usually based on POP3 and SMTP (section 2.3.2, 119ff). IMAP4 (internet messaging access protocol, version 4) is a newer and more powerful standard, but still not supported by many email clients. Today, email substitutes paper-based mail and many phone calls. Compared to the traditional mail system, email is much cheaper, faster with respect to delivery times and creation of messages, less formal and more straightforward. Recent challenges are undesirable mass postings of advertising mails (so-called spam or junk email) and spreading of viruses and trojans via email. Technical countermeasures are functions for email filtering (either on organization-wide servers or within individual email clients) and virus scanners.

List server

List servers support one-to-many broadcasting of messages by managing mail subscription lists and distributing emails to the addresses on these lists. Users usually are free to subscribe or unsubscribe via email (e.g., by a message to listserv@organization with the instruction "subscribe newsletter" in the subject line or mail body). Common fields of application are distribution of newsletters and communication within communities. An example is the ISWorld mailing list that connects several thousand MIS researchers all over the world and is an important medium e.g., to announce conferences, call for papers, ask questions or discuss topics with colleagues about research and teaching in IS. Listserv, Mailman and Majordomo are widely-spread software tools that implement listservers.

Discussions, newsgroups

Discussions (sometimes called blackboards) are forums that allow for many-to-many text-based communication about a selected topic. Messages are structured in threads consisting of messages and replies sorted according to time of their creation. Discussions are one of the key technologies to support communication within communities. Today, messages usually are displayed on and created with standard Web browsers. Since the early days of the Internet, the Usenet based on the NNTP protocol (section 2.3.2, 119ff) is widely used for discussions within so-called newsgroups. The Usenet is hierarchically structured according to topics of discussion. For example, the top level consists of the categories biz (business), comp (computer), humanities, misc (miscellaneous), news (about Usenet news), rec (recreation), sci (applied science), soc (social issues, culture), talk (current issues and debates) as well as alt (alternative). Access to news groups today usually is integrated in other Internet applications like email clients or Web browsers.

4.3.2 Cooperation

Computer supported cooperative work (CSCW). What CSCW is about can be best explained going step-by-step backwards through the single parts of the term "computer supported cooperative work":

Leavitt diamond

- *Work*: To successfully support work and introduce changes, different perspectives have to be taken into account. A classical model to clarify this is the Leavitt diamond (Leavitt 1965). It visualizes four highly interrelated perspectives: people, tasks, structure and technology (Figure 4-14). All perspectives need to be regarded and aligned when designing and introducing technologies. Changes in one perspective will influence all other perspectives. Their inter-connectedness is the reason for the need of an interdisciplinary approach when supporting work in groups.

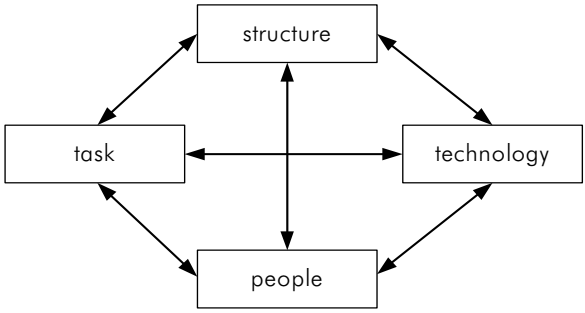

Figure 4-14. Leavitt diamond

Levels of coordination

- *Cooperative work*: CSCW deals with settings where multiple persons cooperate to reach common goals and accomplish shared tasks. Their individual tasks need to be coordinated. Coordination can happen on different levels of interaction: (a) *Informing* is the lowest level of interaction with no or only few communication acts between parties. There are no shared tasks or resources. (b) *Coordination* denotes sporadic communication to share information, structure activities and use common resources. (c) *Collaboration* means that participants work together as a team to reach common goals. Communication is more intensive. Activities, resources and information are shared, but team members do not work together as closely as in the case of (d) *cooperation*, which is the highest level of interaction. Cooperation is characterized by common goals, a shared plan and processing of shared data. The team is responsible as a whole for its outcomes.

ICT-support of cooperative work

- *Supported cooperative work*: Support can happen at content and process level. Thus, it either can be focused on fostering exchange of knowledge, e.g., in shared workspaces, with communication media or

via data bases and knowledge repositories, or on structuring cooperative work processes, e.g., by controlling communication processes (e.g., for scheduling appointments), structuring decision processes (e.g., with decision support systems) and by providing creative environments that spark exchange of ideas.

- *Computer supported cooperative work* thus stands for an interdisciplinary research field that explores how computers can be applied to make cooperation between individuals more efficient. In the following, we will discuss system classes and functions of systems targeted at this goal.

CSCW system classes. Results of CSCW research are incorporated into software prototypes and products on the market as well as into specialized hardware. Since making work in groups more efficient is a wide field, many different terms and system classes have been developed with the boundaries between them being blurry. We will firstly explain the predominant system classes and then provide two classification schemes for orientation.

Group decision support systems

A group decision support system (GDSS), sometimes called group support system (GSS) or (electronic) meeting support systems, is an interactive system that combines communication, computer and decision technologies to support formulation and solution of unstructured problems in group meetings. Participants usually are located at the same place, but may also be geographically distributed. Meeting rooms are extended with hardware like electronic whiteboards and computer screens for each individual participant. GDSS integrate technologies to support communication in groups, structuring of processes by which groups interact (e.g., agenda setting, facilitation) and information processing (e.g., aggregating, evaluating or structuring information). GSS can be classified according to the level of support in *level 1 GSS* which remove communication barriers, *level 2 GSS* that provide decision modeling and group decision techniques and *level 3 GSS* which provide expert advice in selecting and arranging rules in a meeting and thus lead to machine-induced group communication patterns. An example for a well-known system is GroupSystems (DeSanctis/Gallupe 1987, Zigurs/Buckland 1998).

Groupware

Groupware provides general support for collecting, organizing and sharing information within (distributed) collectives of people, such as work groups and project teams over corporate networks as well as the Internet. In contrast to GDSS, Groupware focuses unstructured, flexible processes in workgroups and is a platform for communication and information sharing in groups. Typical functions are email support, group calendars, task lists, directory services and shared places to store structured and semi-structured information. Some Groupware systems feature replication to enable each participant working offline on local copies and to later synchronize the changes with the other team members' data bases.

Examples for well-known products are IBM Lotus Notes and Microsoft Exchange.

Workflow management systems

Table 4-7 compares Groupware and workflow systems, which are discussed in section 3.4.2, 210ff. Workflow management systems focus recurring and structured or at least semi-structured tasks which can be automated easily or with little effort. The focus is on describing, controlling and automating task flows. In contrast, Groupware is seen as a general platform to support collaborative tasks and dynamic processes that can not be structured and described in advance. Nevertheless, actual Groupware platforms often incorporate workflow functionality to support structured (sub) tasks like scheduling meetings or revising documents. In this case, workflows are usually ad-hoc and less complex than in traditional workflow systems.

Table 4-7. Groupware and workflow systems compared

	Groupware	**workflow systems**
goals	support of group and project tasks, efficient decisions	coordinate complex processes efficiently
characteristics	provide an infrastructure to support coordination and cooperation	structure tasks and integrate systems to automate workflows
focused objects	individuals, groups	tasks, workflows
type of tasks	unstructured ad-hoc tasks with many exceptions	structured tasks that can be formalized with rules
source of activity	user	system
source of control	user	system
IT architecture	decentralized	centralized

Shared information spaces

Shared information spaces are virtual places where common resources like documents or links can be stored and manipulated. Additionally, they offer functions for interaction and to collaboratively create and work with documented knowledge. Ideally, shared information spaces are workspaces that integrate all functions to store, retrieve and jointly create information resources as well as to coordinate activities and manage group processes within geographically dispersed groups. A shared information space can be seen as a metaphor that is used to combine a set of Groupware functions for sharing and communicating documented knowledge.

Classification of CSCW systems. In the following, we will discuss two classifications for CSCW systems that are suited to give an overview of

common functions and application areas of CSCW technologies. These are the time/place taxonomy and the 3C model.

Place/time taxonomy

Like the time/place taxonomy for communication modes (section 4.3.1), this classification groups CSCW systems into four classes based on whether users work at the same time or at different times and whether they are at the same place or geographically dispersed (Table 4-8). Systems within quadrant (I) focus on supporting processes and tasks within group meetings, e.g., creative problem solving, presentation and placing decisions. Quadrant (II) is somewhat unusual with respect to the fact that users are on the same place, but use the system at different times. This may be the case when employees of different shifts use a system to exchange messages or work on a joint task. Most communication systems naturally can be classified into the "different place" category and thus into quadrants (III) and (IV). Comprehensive groupware platforms usually provide many functions within quadrant (IV) as their focus is on geographically separated groups.

Table 4-8. Time/place taxonomy of CSCW systems

	same time	**different time**
same place	(I) brainstorming, presentation, Group Decision Support System	(II) bulletin board systems, shared applications
different place	(III) teleconference support system (e.g., chat, video conferencing), shared whiteboards, co-browsing	(IV) group calendar, newsgroups, co-authoring, listserver, workflow management system

The time/place taxonomy leads to a rough classification of CSCW systems, but has its limits. The way and type of support (e.g., structuring decision processes, focus on communication vs. focus on documented knowledge) is no criterion in this classification which leads to the result that different systems belong to one and the same category, e.g., group calendar and workflow management systems. Moreover, the classification is not clear-cut. For example, group calendars can be used by groups in the same place as well as in different places.

3C model

The 3C model distinguishes three orientations of CSCW systems: (a) *Communication-oriented* CSCW systems are targeted at synchronous and asynchronous information exchange between participants. Examples are video conferencing or messaging systems. (b) *Coordination-oriented* systems focus structuring and control of group tasks and task flows, e.g. through workflow functionality. (c) *Collaboration-oriented* systems such as shared information spaces and group editors aim at supporting participants that work on joint tasks.

System classes

Figure 4-15 shows CSCW system classes and their primary orientations. A categorization of systems according to these dimensions leads to (at least) four classes: (a) *Conferencing systems* primarily deal with supporting synchronous communication between persons. (b) *Workflow management* systems are oriented towards coordination of group activities. (c) Sometimes, the term *workgroup computing* is used as a general term to denote the shared creation, collection, processing and storing of information. In a narrow sense, workgroup computing combines functions to make working on joint tasks more efficient, without focusing communication and coordination tasks. (d) *Shared information spaces* are somewhere between all three dimensions, as they may combine different functions dependent on the application scenario.

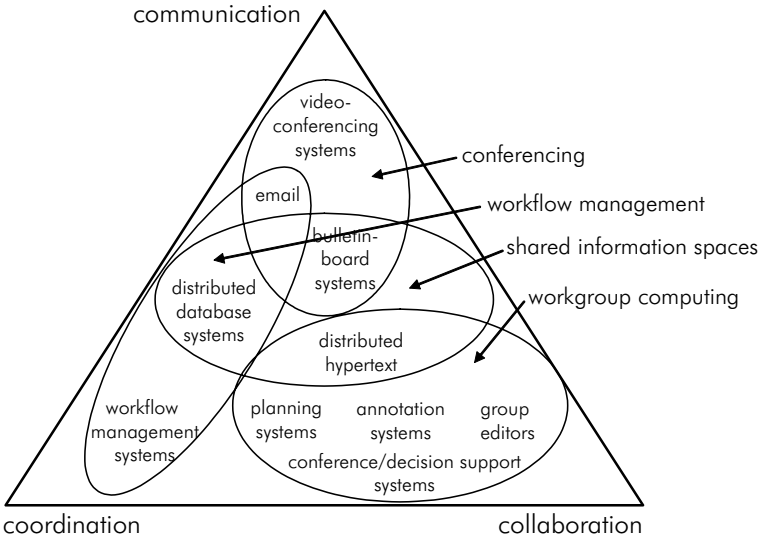

Figure 4-15. 3C model for classification of CSCW systems

Challenges

It should be noted that these classifications are useful to provide an overview and to analyze CSCW technologies, but do not regard how they are applied by individuals. These continually switch between different styles of cooperation, synchronous and asynchronous communication, between formal and informal work styles, working jointly or are separate from each other. Many systems focus on one combination of these characteristics. Thus, it can be seen as the main challenge to create an infrastructure of seamless systems and services. Future trends in this direction are summarized under headings like “any time, any place Groupware” and “seamless systems”. These trends and requirements are closely related to the characteristics of the EKI, which aims at providing seamless and integrated support of knowledge work.

Advanced functions of groupware. Groupware combines functions that can be characterized to be basic and widely-known like email, chat, blackboards or task lists. A major benefit is the combination of these functions into one comprehensive platform. Groupware can provide advanced functions targeted at areas expected to substantially influence the performance of computer-supported groups. As examples, we will explain joint authoring, group awareness, collaborative filtering and group calendars.

Joint authoring

Joint or collaborative authoring (in short: co-authoring) denotes jointly creating and exchanging documents synchronously or asynchronously by using multi-user editors. This allows for integrating isolated writing processes into a joint writing process in which authors' actions more directly influence each other. To avoid conflicting accesses, parts of a document can be reserved on different levels of granularity (check-out). Changes on the document are marked and can be approved or rejected by other authors. Comments for co-workers can be placed within the document. Notification mechanisms help authors to stay informed about activities. Usually, the co-authoring process is backed by additional communication media that allow for synchronous communication to share opinions, reach consensus and coordinate activities.

Wiki

A special form of collaborative publishing that has recently received much attention is wiki. "Wikiwiki" is a Hawaiian word, turns up in many contexts and usually means "quick" or "informal". A wiki (short form for WikiWikiWeb server) is a freely expandable collection of modifiable interlinked Web pages (Leuf/Cunningham 2001). Every user is allowed to create and edit pages and is involved in an ongoing process of collaborative content creation. In the pure wiki-approach, there are no user privileges or access limitations. Every user possesses the same rights and is able to access and edit Web pages within a standard Web browser. Wikis are a special class of Web content management systems (WCMS, section 4.2.2, 258ff) with the main difference being that they focus fast collaborative content authoring by everyone who is inclined to participate whereas WCMS rely on privileges for authors. Today, there are many wiki Web sites with contents that have surprisingly high quality (e.g., http://www.wikipedia.com).

Group awareness

Group awareness refers to the perception of what other members of a group are currently doing or have done in the past hours, days or week. It is defined as an understanding of the activities of others which provides a context for own activities (Dourish/Belotti 1992). One of the first notions of awareness was a study that observed work in a London Subway control room. It researched the question of how people in a group know when to do what. It showed the importance of implicit cues about the activities and the role of subtle communication like mimics and gestures which was coined "periphal awareness". Users of Groupware are physically separated

from each other which prevents direct awareness. Awareness functions are thus seen as a way to make coordination more efficient.

Example BSCW

Figure 4-16 shows an example screenshot of OrbiTeam BSCW, a Web-based collaboration system. Icons visualizing user events are assigned to items within the system (e.g., folder, document). If a user clicks on one of these icons, she gets more detailed information, e.g., a list of users who recently accessed or read the item.

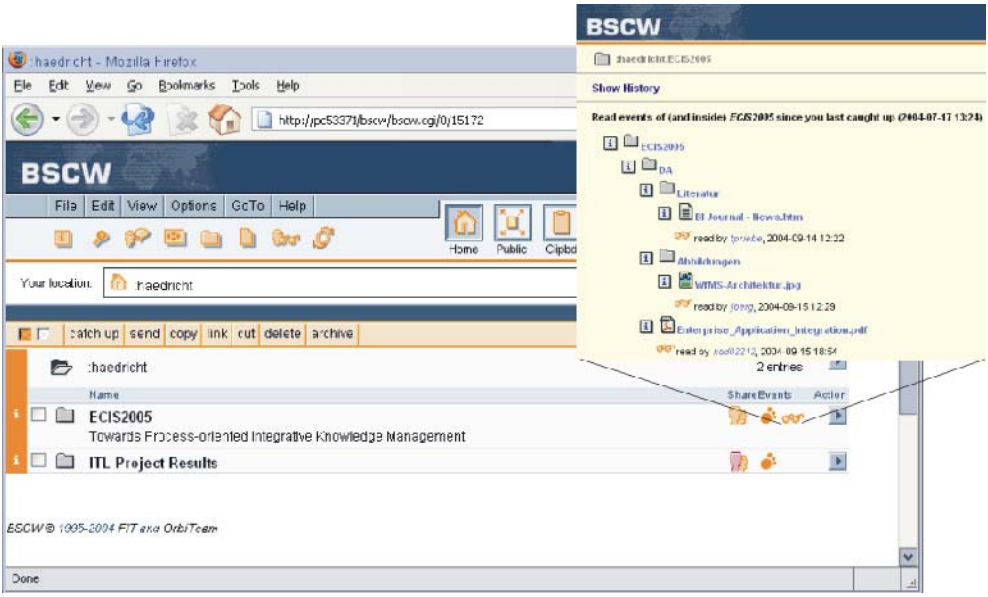

Figure 4-16. Awareness within OrbiTeam BSCW

Modes of awareness

Different modes of awareness can be distinguished according to whether it is focused on past, present or to which future events in a system and whether the events, to which their attention can be directed are tightly or loosely coupled to the actual work context. Table 4-9 shows example questions for each of the resulting mode of awareness. Users will need more detailed awareness information for tight coupling. Information for loosely coupled areas of interest need to be provided as unobtrusively as possible ("peripheralness" of visualizations).

Table 4-9. Group awareness modes

	past	**present**	**future**
tightly coupled	What changes happened on the draft that I uploaded last week?	Which member of my project team is currently online?	Who will read my project report?
loosely coupled	How many documents were added last month?	What are the activities of other participants in the system?	How will other workspaces evolve?

Positive results of group awareness are that it encourages spontaneous, informal communication as well as creation of a shared culture since group members are better informed about activities, state and workload of other group members. Moreover, all participants stay informed about the current state of collective team activities. Therefore, members are able to judge whether individual contributions are relevant to the group activity and evaluate their own actions with respect to the group's goals and progress. Awareness thus facilitates planning of group processes and collaboration within groups.

On the downside, awareness information may clutter the user interface with a lot of information not directly necessary or useful. It must be decided thoroughly when users need what type of information. Awareness functions rely heavily on capturing data about user behavior (e.g., accessed functions or documents) which may limit privacy. It should always be transparent for users which information is recorded. Analysis and publishing of a single user's behavior to be prevented.

Collaborative filtering

Filtering deals with identifying and presenting information according to user information needs. Commonly filtering is applied for individualizing results of search queries. Furthermore, two types of advanced filtering techniques can be distinguished: (a) *Content-based* filtering analyzes and selects contents with regard to content-related criteria like pre-defined key words, similarity to other contents the user has accessed or more advanced methods like neural networks that represent evolving interest profiles of users. Content-based filtering selects items because they are similar to items a user liked in the past. In contrast, (b) *collaborative filtering* identifies and selects (groups of) users whose tastes seem to be similar to those of the given user and then recommends him what these users liked. Examples are suggestions like "customers who bought this book also bought these titles / titles by these authors" by Amazon.com. As a consequence, it is possible to select content that is dissimilar to content that a user has seen in the past or to pre-defined search terms. This approach is effective when the number of users is large compared to the amount of content in a system. If the number of users is small in relation to the volume of informa-

tion, there is the danger of the coverage of ratings becoming sparse, thinning the collection of recommendable items.

Group calendar

Group calendars integrate individual schedules of group or team members and today are an important tool to manage joint appointments. They facilitate awareness of other team members' appointments and planning of suitable meeting times. Group calendars usually access individual schedules and present all appointments of selected persons, e.g., participants of a project meeting to be scheduled, in an integrated view. This allows to identify time slots where all participants are able to attend the meeting. Group calendars usually offer pre-defined workflows to invite participants to an appointment and to handle acknowledging or rescheduling appointments.

Electronic calenders need to support the definition of access privileges, since they usually contain sensitive information. This comprises at least determining who has access and which calendar items are not accessible at all. Electronic calendars always are in danger of being misused to control employees. Their introduction thus needs to consider privacy issues and workers' participation.

Group calendars are only useful if all team members maintain their individual schedules electronically. In the past, this regularly was a challenge since electronic calendars were no replacement for their paper-based pendants due to their limited mobility and usability. The wide-spread use of personal digital assistants (PDAs) and their ability to synchronize with electronic calendars lowered this barrier (section 5.3.3, 343ff).

Example: Groove. The product Groove from Groove Networks targets collaboration in small dispersed groups and is based on a peer-to-peer architecture (see "Peer-to-peer architecture" on page 53). Thus, Groove peers communicate directly with each other as opposed to conventional Groupware platforms which dispatch all communication by a central server.

Collaboration takes place in workspaces that are accessed with a MS Windows client (called transceiver, Figure 4-17) and contain a number of tools to manage contents like basic text, documents, calendar items or images.

Tools for collaboration are group calendar, contact list, blackboard, meeting minutes and task list. A sketchpad (whiteboard) and an outline tool (structured list) offer basic support for brainstorming sessions. A group of users can jointly browse Internet/Intranet-pages with the help of co-browsing functionality. A "navigate together" option synchronizes the interface of the workspace. Awareness services provide information about current activities of other users, e.g., the workspace and the tools they currently access. Co-edit functions allow users to simultaneously work on MS Word and MS PowerPoint documents. Files can be stored in a basic hierarchical folder structure in the files tool.

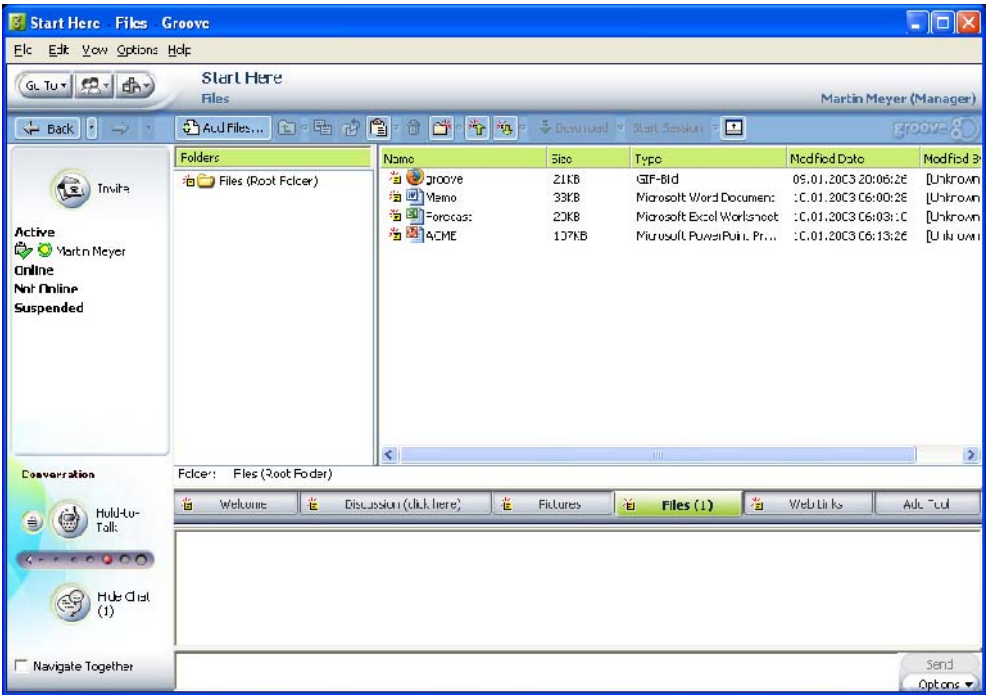

Figure 4-17. Groove transceiver

All data resides in XML stores on the local hard disks of the peers. If the peer is online, changes in workspaces are continuously transmitted to all peers. Otherwise, the differentials are synchronized when he switches back online. Local data and messages to other peers are encrypted by a security service. Peer connection services determine IP addresses of other peers and handle communication using the proprietary simple symmetrical transmission protocol (SSTP). Device presence services handle the detection of other peers and their online/offline status.

Groove functions without centralized servers, though they can be additionally deployed for administrative tasks such as managing licences, distribution of software updates, monitoring system and network usage, to offer directory services, a public key infrastructure (PKI) and a basic account management as well as to ensure the synchronization of peers with relay services.

4.4 Learning Services

Learning services help participants to build individual and team knowledge and develop skills. Although learning services generally can be applied for all types of knowledge, they are primary tools to facilitate communication and to transfer implicit, personal knowledge. They include functions that enable creation of learning modules, address multiple information channels and also support collaboration between learners.

EKI especially need to support informal learning that is a process taking place in everyday experience, often on subconscious levels. Research supports introducing informal learning not as a replacement for formal learning activities, but as a complement to them. Workplace learning is formal and informal learning at users' work environments. In professional organizations, a mentoring relationship between experienced individuals and novices is often an important source of informal learning for both, the expert and the novice.

Section overview

At the beginning of this section, we will introduce some foundations that are important for understanding different views on learning as well as different types of learners and how this influences decisions about which techniques are best to support and enable learning (section 4.4.1). Then, services that aim at supporting the design of learning modules and related challenges and issues are discussed (section 4.4.2). Learning objects are presented as examples of what such learning units or modules could look like in the context of an EKI. Finally, we will deal with services and functions of e-learning suites and with functions that aim at supporting examinations and tests to give feedback about learning progress (section 4.4.3).

4.4.1 Foundation

Learning paradigms. Due to different views and perspectives on what exactly learning is, how it happens and by which means and ways it can be supported, it is not easy to give a commonly accepted definition of learning. A general definition is that learning comprises all changes of behavior caused by experiences. We will shortly outline and compare three different learning paradigms that result in different views on how learning can be supported and how e-learning software should be designed.

Behaviorism

Behaviorism regards learning as reflex under certain conditions. Stimuli evoke corresponding responses of the learner. Ultimately, behaviorism focuses on creation of conditioned reflexes and is often referred to as classic conditioning. It is based on the experiments of Pawlow and was later extended by Skinner's reinforcement theory which postulates that behavior is more likely to be repeated when it is rewarded. In 1958, Skinner developed the concept of programmed instruction that applies behavioristic assumptions to technical media. It led to the construction of simple

mechanic learning machines and later was transferred to learning software. Questions are sequentially presented to learners who have to formulate or choose correct answers. Goal is to generate as many correct answers as possible since only correct behavior reinforces learning. Critiques to behaviorism and his learning machines that build on this view of learning state that learners are only motivated to "push the right button" and to reproduce memorized fact knowledge. Thus, behavioristic methods only seem suited to teach basic fact knowledge.

Cognitivism

Cognitivism focuses on internal thinking processes of learners and therefore differs fundamentally from behaviorism. The approach states that external stimuli are processed autonomously and individuals can not be directly controlled by them. The brain is seen as information processing unit that is based on internal dynamic patterns. On an abstract level, brain and computer are compared in this view in that they are both information processors. Cognitivism focuses on problem solving, it is not just about finding the right response to a certain stimuli, but to find methods and ways to find good answers to a given problem. Critique states that cognitivism focuses too much on given problems, neglecting the fact that problems often first need to be recognized. Additionally, it is critically pointed out that the human body plays only a minor role in cognitivism so that this view has difficulties with explaining bodily skills.

Constructivism

The core assumption of constructivism is that there is not only one objective representation of the world, but that knowledge is constructed autonomously by individuals based on their concepts and ideas. Thus, many different meanings and representations are principally possible for one and the same concept. One "situation" might appear differently for different human actors. Learning is seen as an active process in which knowledge is linked to previous experiences and human actors recognize and solve problems in context-dependent situations. Education thus needs to consider the present knowledge of learners and should involve and activate them. Communication and cooperation play an important role because social interactions are an important means to acquire knowledge.

Table 4-10 summarizes the described learning paradigms. The table also characterizes some implications on learning software that considers these paradigms

Table 4-10. Learning paradigms compared (Baumgartner/Payr 1994 cited from Röder 2003)

	behaviorism	**cognitivism**	**constructivism**
metaphor of brain	passive container	information processing unit	informationally closed system
knowledge	stored	processed	constructed
learning goals	correct answers	good methods to find answers	to master complex situations
learning paradigm	stimulus-response	problem solving	construction
strategy	instruction	observing and helping	cooperation
role of teacher	authority	tutor	coach, trainer
feedback	externally given	externally modelled	internally modelled
interaction	pre-defined and fixed	dynamically dependent on teaching model	self-referring, circular, determined by structure, autonomously
characteristics of software	fixed sequences, quantitative time/answer statistics	dynamically controlled sequences, predefined problems, analysis of answers	dynamic, complex networked systems, no predefined problems
software paradigm	teaching machine	artificial intelligence	social-technical environments
ideal software classes	drill & practice	adaptive systems	simulations, microworlds

Types of learners. Besides different theories and views on learning in general, individuals differ with respect to the way they learn. Each individual prefers a way of learning and for effective knowledge acquisition.

Factors for distinguishing types of learners

This depends on a variety of factors (Schrader 1994) such as motivation to learn, different ways of working with texts, ways of processing information, confidence of the learner in success vs. fear of failure, reactions to difficulties during learning, planning of time and work, attitude towards examinations as well as preferred channels of learning. These factors can be used to define types of learners. Ideally, learning contents are prepared for alternative types of learners so that each learner is able to access learning materials in the format that best fits her needs. However, e-learning must not impose too much additional efforts on designers of contents and thus regularly pragmatic approaches are chosen.

Intrinsic vs. extrinsic motivation

Basically, learners can be distinguished to be intrinsically or extrinsically motivated. Intrinsically motivated persons learn because they are interested in the learning subject or enjoy dealing with it. Presentation of learning contents that support intrinsically motivated learners should abide

curiosity and leave control over learning paths to learners. Ideally, learners should be able to dive into a "learning world" with abundant information. Tests and examinations have no primary role and should be presented on demand of the learners.

In contrast, extrinsically motivated individuals learn to reach defined goals like in order to earn certificates or diplomas, secure their current workplace or get on their career or raise recognition. Extrinsic motivation leads to different needs with regard to the design of learning materials. They have to be clearly structured into manageable and consistently presented learning units. Motivational measures, tests after presentation of materials, feedback about learning progress and consideration of breaks are more important issues.

Active, self-motivated learning is supposed to be more effective than passive consumption of learning contents. Active learning typically is the result of intrinsic motivation which thus should be fostered as much as possible.

Preferred media and senses

A straightforward way to distinguish learner types is to categorize learners according to the media and senses they prefer for learning:

- *Auditory user types* prefer acoustic presentations,
- *visual user types* prefer figurative presentations,
- *abstract user types* prefer textual presentations, and
- *audio-visual user types* prefer video-based presentations.

These groups may directly be regarded when designing learning materials, but do not take into account other important factors like existent knowledge, motivation to learn and learning behavior. Thus, categorizations like this one are very limited with regard to their applicability for individualizing contents.

Behavior and way of generating new insights

Kolb classifies learners according to two dimensions: (a) according to their preferred learning behavior and (b) according to the way how learners generate new insights (Kolb 1976). He separates two classes of learning behavior. Learners can either prefer to actively experiment or to observe things reflectively. New insights can either be generated by forming abstract concepts or by practical experiences which is close to the differentiation between *induction* (inference of a generalized conclusion from particular instances) and *deduction* (conversely to induction the inference in which the conclusion about particulars follows necessarily from general or universal premises). Using these two dimensions, Kolb's *learning style inventory* (Table 4-11) distinguishes between four types of learners:

Divergers

Divergers rely on observance and on practical experience, they excel at viewing subjects from different perspectives. They are often very creative and imaginative individuals. Emotions play an important role when dealing with ideas.

Assimilators

Assimilators prefer to observe reflectively and to form abstract concepts. They like to deal with things and theoretical models rather than with

human beings. They are best at constructing coherent theoretical models whose practical applicability is not in the center of interest.

Convergers

Convergers like to experiment actively and to form abstract concepts. A main difference to assimilators is that they strive for practical application of ideas and theories. They focus on manageable situations and clear-cut solutions and are oriented rather technically than emotionally.

Accomodators

Accomodators learn through active experiments and practical experiences. They are good at adapting to new situations and inclined to accept risks. They create plans and theories, but fast discard them if they do not fit with new perceptions and insights as forming abstract concepts is not their way of learning.

This categorization should be applied with caution, it can only give a rough orientation. Especially in connecting disciplines this model leads to assertions contradicting each other.

Table 4-11. Kolb's types of learners (Kolb 1976)

		preferred learning behavior	
		experiment actively	*observe reflectively*
way of generating new insights	*forming abstract concepts*	converger	assimilator
	practical experiences	accomodator	diverger

Expert level

A central factor that determines didactic methods and the way to design learning content is the learner's expert level of learners which. According to Dreyfus/Dreyfus, the level of experience can be classified into *novice*, *advanced beginner*, *competent*, *skillful master* and *expert* (see "Expertise" on page 15). In contrast to novices that rely on descriptive knowledge, expert knowledge and thinking is organized differently and strongly based on experience and intuition. Each level of becoming an expert requires other didactic strategies which are contrasted in Table 4-12.

Novices need structured learning material that is presented clearly and manageably. Tests and repetition are best to deepen fact knowledge in a certain domain. Advanced beginners can be supported by tutors that answer detailed questions and give feedback about the learning progress. Competents need to discuss and learn about actionable knowledge in groups by jointly working on a task, deepening practical knowledge and opposing different views and perspectives. Complex simulations that can be used individually or in moderated group sessions are a way to support learning of skillful masters that already possess knowledge about a variety of typical situations and do not rely on the analysis of single attributes of a situation. Experts only can acquire new knowledge by researching a subject. There are no pre-defined didactic methods as research can take place

in a variety of forms. Sharing knowledge and experiences with other experts is centrally important.

Table 4-12. Didactic strategies and expert levels (based on Röder 2003)

expert level	didactic strategy
novice	drill & practice
advanced beginner	support by tutors
competent	work in groups, communication of actionable knowledge
skillful master	complex simulation
expert	research, share knowledge with peers

History and trends of computer-supported learning. Distance learning and e-learning are no completely new topics. The roots of distance learning can be traced back to 1865 where first letters with instructions for learning languages were distributed. This was called the "method Toussaint-Langenscheidt" named after it's inventors.

Distance learning, telelearning, and e-learning

Distance education denotes a process of communicating knowledge between geographically separated teachers and learners. The term distance learning is used synonymously, sometimes to emphasize the interaction between teachers and learners. If communication is supported by electronic media, this activity is characterized as telelearning. The term synchronous telelearning is applied when teachers and learners communicate at the same time over electronic media, e.g., by means of videoconferencing. In the other case, when teachers and learners are geographically and temporally separated, one speaks of asynchronous telelearning. In this case, usually material created and stored on a storage medium (e.g., a CD ROM) is accessed. E-learning is a much younger term and emerged at the end of the 1990s together with the wide-spread use of the Internet and other such terms like e-business or e-government. Our definition of e-learning emphasizes that multimedia contents need to be provided online and together with functions that enable interaction, though e-learning is often used in a broader sense as comprising other forms of electronically supported learning.

Definition of e-learning

E-learning is ICT-supported learning with the help of multimedia or hypermedia contents that are online accessible for the learner backed by functions that enable communication between learners and teachers as well as among learners.

Programmed instruction

A first attempt to support learning with ICT was programmed instruction in the 1960s, where answers of learners were automatically interpreted. The goal was to infer the competency level of learners and conse-

quently to individualize learning contents. Learners should be given the possibility to learn autonomously with a system that guides them through the content. Due to restricted access to computer systems in these times, programmed instruction primarily was applied to teach knowledge about programming languages. This approach later was continued under the topic *intelligent tutorial systems*. This class of systems class is characterized by complex models of learners and regularly incorporates techniques of artificial intelligence to intelligently individualize the contents. Programmed instruction was criticized for restricting learners too much to pre-defined learning paths as well as for the underlying behavioristic assumption that learning is solely enforced by positive feedback, which was falsified in scientific experiments.

Hypertext and hypermedia systems

An alternative approach called *explorative learning* emerged in the 1980s when hypertext and hypermedia systems offered the possibility to freely navigate through contents. Hypertext is any text that contains links to other documents. Words or phrases in the document are linked to other documents that can be easily retrieved and displayed to the reader. Hypermedia extends hypertext with multimedia objects, including sound, motion video, and interactive 3D animations.

Computer-based training

When computer systems became technically more advanced, it was possible to create contents and learning systems that needed more storage space and high-performance computers. At that time, computer-based training emerged which stands for multimedia learning units that provide text, sound and video as well as complex interactions between learners and the computer system and was usually distributed on CD-ROM. In the first phases, separation of programmers and authors of learning contents in large programming projects to create learning modules led to problems. Often, a well-designed technical implementation was emphasized over didactically sophisticated solutions. Later on, authoring tools eased the creation of learning units, though still it remains a time-consuming and expensive process.

Web-based training

At the time when Internet and Intranets were used more widely, Internet technologies were applied to design learning units. *Computer-based training* became *Web-based training*. The design of optically appealing user interfaces and well-designed contents regularly was emphasized. Unfortunately, the potentials of Web technology to enable communication between learners and teacher as well as among learners often was not utilized.

Blended learning

Developers of e-learning systems and its predecessors aim at supporting learning solely through the use of ICT. In contrast, blended learning nominates an integrated, harmonized mix of methods for online, offline and face-to-face learning. For example, participants of university courses prepare at home using online course material and additionally meet in seminar sessions. Blended learning is the youngest term in this area (approximately since 2000). Learning with new media is still in focus, though the

amount of non-ICT-supported learning sessions and ICT-supported sessions may vary depending on the context. For example, a university could use computer technologies to enhance its pre-existing courses but still rely mainly on seminar sessions. A company that previously offered only virtual courses, could enrich its services by some face-to-face learning sessions. One reason why blended learning is seen as a promising approach is that it combines advantages of traditional face-to-face learning and of new technologies to make learning more effective. Nevertheless, up to now, it does not substantially lower the costs of learning (e.g., by systematically reusing courses or bigger target groups) or by establishing bigger audiences.

Advantages and disadvantages of e-learning. Many hopes are connected to the application of new media and especially to Internet technologies to support learning. Especially the ability to span geographical and temporal barriers and powerful methods to design contents regularly propel trust in the potenials of e-learning.

Advantages

A major advantage is that learners can freely choose time and place for learning. In countries where it is necessary to travel large distances to the next school or university, the hope is that e-learning could substantially reduce travel costs. Some countries, e.g., Australia, have a long tradition in offering distance educational services. E-learning could be a possibility for universities to offer their services in a large region and to reduce their costs substantially by reusing contents and improving efficiency of creation and distribution of learning contents as well as of teaching and examinations. Cooperation can be fostered by educational institutions in different places and countries. Other advantages are that the new media offer new possibilities to comprehensibly explain complex issues with the help of animations, simulations as well as audio and video content and to individualize learning. Connections to external data bases, library catalogues or Web resources enable learners to retrieve additional material and to autonomously deepen their knowledge. Collaboration and cooperation between learners are possible and it is easy to directly contact experts in a special domain which could motivate learners and support their creativity. Altogether, new media are seen as a way to involve learners more actively. As noted earlier, this is seen as supporting for effective learning.

Disadvantages

However, e-learning may have disadvantages. Social interaction between learners over electronic media like chat, email and discussions is not of the same quality as direct face-to-face interaction. Thus, e-learning is criticized of isolating learners rather than fostering cooperation. Many forms of learning (e.g., discussing case studies in group sessions) and some ways of teaching (e.g., demonstrative experiments in physics) can not be substituted by electronic learning contents. The easiness with which teachers can effectively publish contents in an e-learning system is a core success factor of e-learning systems.

4.4.2 Design

Design of learning modules comprises authoring of multimedia contents and especially preparing learning materials to enable learners for effective knowledge acquisition. Thus, decisions how to structure contents and how to present them need to be considered thoroughly in advance.

Learning goals and structuring of contents. Before learning contents can be designed, some preliminary steps that deal with the general planning of the presentation of learning contents need to be undertaken.

Definition of learning goals

Learning goals need to be defined that describe the knowledge that the learner should possess or a behavior the learner should show after a successful learning process. Learning goals are widely discussed. Critics note that if they are open to the learner, they may be constricting because the learner focuses too much on the learning goals when working the material. Nevertheless, they enable the learner to form expectations and give feedback with regard to the success of learning. Learning goals are a foundation for creating examinations and tests and last but not least, are an important instrument to plan learning units.

Determining the learning strategy

Learning strategy determines which role the learner has. Two paradigms can be distinguished: instructional paradigm and problem-solving paradigm. Design following the instructional paradigm represents a behaviorist view. The task of learners is to absorb contents chosen by a tutor or by the learning system. After completion of a learning unit, learners then get positive feedback to be motivated to continue. In contrast, the problem-solving paradigm takes a constructivist view and regards learning as an active and dynamic process. Learners autonomously work through the content and relate it to their cognitive structure. The learning environment needs to flexible and motivating for the learner to acquire contents, e.g., in the sequence he prefers most.

Segmenting and sequencing contents

Segmenting denotes dividing the whole body of learning contents into single learning units. Sequencing stands for determining the order in which learning units are to be presented. This is not an easy task as there usually are multiple alternatives to divide and sequence contents, e.g., foundations can be presented separately at the beginning or “on demand” not until they are necessary for understanding. The “logic” of the contents itself usually is not sufficient to give an orientation. When presenting two or more different subjects, two basic types of sequencing can be distinguished: linear and spiral sequencing. Linear sequencing discusses a subject until the desired competency level is reached and then changes to the next. Advantages are that learners can concentrate on one subject and materials can be organized more easily. Spiral sequencing presents every topic in multiple cycles. At first, the foundations of the different subjects are presented. Then, all sub-topics for all subjects are detailed successively (Figure 4-18).

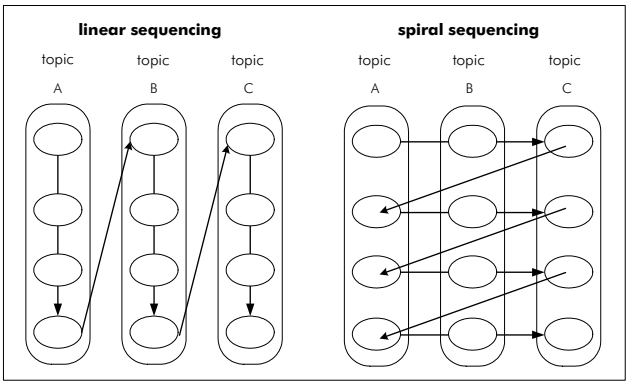

Figure 4-18. Linear and spiral sequencing

Three generic phases

A basic scheme for the structure of single learning units is their separation into three basic phases which is applicable for both learning strategies presented above. (1) In the *preparation phase*, learners get an overview of the course and its contents. Existing knowledge is activated and related to the new knowledge to be learned. Examples for activities during this phase are explanation of learning goals, motivating and emotionally involving users or participants, testing existing knowledge through a short examination and kick-off workshops. (2) *Acquiring* is the next phase. Here, learning contents are presented or learners individually work through them. (3) The last phase, *reinforcement,* has the goal to secure results and check how much progress has been made. This can happen by means of small interspersed questions, recapitulation of contents, giving hints for further materials (e.g., links, literature) or offering opportunities for communication.

Using special content types. In EKIs, in addition to more traditional content types like text or figures, other content types like audio, animation and video are regarded to be well-suited to design and enrich learning contents.

Audio

Three general types of audio can be distinguished: speech, sound effects and music. Speech can be applied to mediate information, to substitute text or to control attention (e.g., by explaining a picture). Music and sound effects are well-suited to stimulate emotions and can have the functions to constitute a setting (e.g., digeridoo sounds that accompany an Australian landscape) or a certain period of time (e.g., music of the 1980s), identify recurring persons or situations (e.g., by repeating themes), to prepare the learner for changes in the presentation (e.g., by effects that mark the transition to the next sequence) or to control speed of a presentation.

Animations

Animations are widely used, unfortunately often more to impress rather than to teach. It depends on the background of target group and learning goals whether animations are applicable or not. They can be applied to

decorate the learning environment (should be applied with caution because they may distract learners from the actual learning contents), to control attention or motivate the learner (e.g., by giving feedback in the form of an animation) and present things that otherwise could not be observed (e.g., dynamic processes in a chemical reactor, ecological systems, circulation of blood). In the last case, animations play an important role in building mental models.

Simulations

A simulation is a model of a dynamic system in the perceived reality. Traditionally, complex simulations are applied in fields where faults of unexperienced learners can lead to dangerous results (e.g., steering an airplane or operating a power plant). Simulations can be fruitfully applied in e-learning as they support explorative learning by enabling the learner to discover relationships in systems by manipulating parameters. Additionally, they allow for experimenting in areas where this otherwise would principally be impossible, e.g., simulation of breedings in biology (genetics) or physical experiments in an environment with different conditions than found on earth.

Video

Video is often applied in e-learning environments. The use of video usually causes high efforts, but has a number of advantages. Information density is high and the real-world scenario may be accurately reproduced. It is especially useful for illustrating complex processes and social interactions (e.g., movements in sports, sightseeing tour, visualization of rooms and buildings, product presentation, social dialogues), descriptive and authentic and thus fosters memorizing knowledge. Moreover, dynamic and visual media produce emotional reactions regarded as well-suited to gain the learner's attention. However, emotions may limit the information processing capacity of humans. Video sequences that cause strong emotions should thus be used with caution.

Principles for designing multimedia contents. Developers of learning units need to consider some design guidelines and patterns that aim at fostering learning. As an example, we will present principles based on the SOI model (Selection, Organization, Integration) for multimedia learning (Mayer 2001).

Assumptions of the SOI model for multimedia learning

The SOI model builds on three assumptions about human cognition: (1) Humans possess two different channels for receiving information, one for visual, nonverbal information and another one for auditive, verbal information. Changing between these channels demand high cognitive resources and thus switching between contents that address different channels should be applied cautiously. (2) The information processing capability is limited in that information to be processed must be held in a working memory that can only store up to seven (+/- two) so-called *chunks* of information. Thus, learners need to repeatedly go through e.g., text, figures or animations until they have fully acquired the contents. 3) To memorize a subject lastingly, humans need to construct a coherent mental model of it.

This happens by a number of active cognitive processes of inference, comparison, generalization, enumeration and classification.

Table 4-13 summarizes six general principles for designing multimedia. *Six design guidelines*

Table 4-13. Six multimedia design principles (Clark/Mayer 2002)

principle	description
1. multimedia principle	use words and graphics rather than words alone
2. contiguity principle	place corresponding text and graphics near each other
3. modality principle	present words as audio narration rather than onscreen text
4. redundancy principle	presenting words in both text and audio narration can hurt learning
5. coherence principle	adding overly attention-catching material can hurt learning
6. personalization principle	use conversational style and virtual coaches

The multimedia principle encourages not only to enrich and decorate text with graphics, but rather to meaningfully relate text and graphics by visualizing its organization and structure or clarifying relations between things (e.g., with the help of diagrams). Ideally, learners should be motivated to actively deal with the subject. *Multimedia principle*

The contiguity principle refers to the arrangement of text and graphics. Searching and connecting text and graphics needs additional cognitive resources and thus it must be secured that text and graphics should always be simultaneously visible on screen, that words and explanations are located closely to corresponding graphics. Pop-up texts that appear when the mouse pointer is located over an item are an example of how to implement the contiguity principle. *Contiguity principle*

The modality principle refers to the partitioning into an auditive and a visual channel for information perception. If the load on one of these channels is too high, the learner can not process the whole information presented. By presenting information as speech, the visual channel is disburdened. This is especially useful if a figure or animation needs to be explained. *Modality principle*

The redundancy principle states that the same content should not be presented simultaneously as written text and speech as both channels may affect each other negatively in this case. *Redundancy principle*

To motivate learners, e-learning content is often enriched by background music, entertaining elements like animations or stories that are not directly related to the subject. However, results of research on cognition argue for a reduction of contents not directly related to the subject of learn- *Coherence principle*

ing. Reasons are that learner's attention is attracted to unimportant information. The learning process is interrupted and information that is actually unimportant may be used to internally structure and organize the subject.

Personalization principle

Personalization principle calls for a personal language style that directly addresses the learner. This stands in contrast to the wide-spread view that an objective style of writing is most appropriate. This principle is based on the assumption that the computer to a certain degree acts as a partner for social interaction. Agents that guide the learner through the contents can help to mediate more closeness between learner and the learning material. Nevertheless, the appropriateness of agents and personal style largely depends on the target group.

Authoring tools. Creation of e-learning contents involves a rich variety of authoring tools to create learning units and multimedia contents. They can be divided into three classes: professional authoring systems, rapid content development tools and other tools to create content.

Professional authoring systems

The first class of tools aims at developing professional multimedia e-learning courses that allow for complex interactions. They provide extensive functions and high degrees of freedom to create different courses. Authors need to be experienced content developers, usually with programming skills. Examples are Macromedia Authorware, SumTotal Toolbook and Trivantis Lectora.

Rapid content development tools

To support easier and faster creation of learning contents, rapid content development tools provide more standardized and pre-structured ways to create, use and manage learning units. Authors do not need to have equally advanced technical skills. However, these systems provide less flexible ways for creating desired results. Examples are ISM EasyGenerator, Macromedia Breeze and Dynamic Media Dynamic Powertrainer.

Other tools

The third category comprises a variety of tools applied to create (multimedia) content. These are software products to create audio and video content, programming tools with a focus on programming languages predominant in the Web area like Java, JavaScript or ASP as well as editors to create HTML pages. Examples are Macromedia Flash for animations, the integrated development environment Eclipse for programming environments as well as Microsoft FrontPage for the class of HTML editors.

E-learning standards. There are numerous standards that aim at supporting exchanging, reusing and integrating learning contents and differ with respect to extensiveness, complexity and implementation into existing platforms and tools. Predominant aim is to secure interoperability between learning systems. We will briefly discuss the most wide-spread specifications.

LOM (learning object meta-data)

The LOM (learning object meta-data) standard specifies meta-data for certain types of digital or non-digital learning objects[14] that can be used, re-used or referenced during any learning activity such as teaching pro-

grams, time schedules, learning objectives, slides, lecture notes and Web pages.

Meta-data includes educational or pedagogical features like pedagogical type (e.g., active or expositive), courseware genre (e.g., hypertext or exercise), approach (e.g., inductive, deductive, explanatory), granularity (e.g., course, unit, lesson) as well as data about the educational use of the resource like roles that apply the resource, prerequisites, educational objectives, target audience, difficulty and estimated time to work through the contents. Moreover, the meta-data descriptions contain data about the life-cycle of the resource, technical features of the resource like format, information for rights management and relation to other resources.

The LOM standard is increasingly implemented in current learning platforms and was defined by the LTSC (Learning Technology Standards Committee) of the ISO IEEE (ISO Institute of Electrical and Electronics Engineers).

Educational modeling language

Educational modeling language (EML) is a comprehensive notational system to describe units of study e.g., courses, course components and study programs. It consists of

- learning model (learning context and educational approach),
- unit of study model (e.g., learner characteristics, learning objectives),
- domain model (characteristics of the subject), and
- theories of learning and instruction (information about perspective on learning, e.g., behaviorist, cognitivist).

EML is the most powerful, but also the most complex standard and thus more demanding for authors, tutors and programmers which could hinder its wide-spread use.

Public and private information

PAPI (public and private information) of the IEEE LTSC is a specification of a learner model, i.e. for data that characterizes a learner and her knowledge or abilities. The model includes personal data, data about relations to other users, security credentials, learning preferences, performance-orientation, and portfolio information (collection of a learner's works that illustrate his abilities).

Computer managed instruction systems

CMI (computer managed instruction systems) is a standard originally defined by the AICC (aviation industry computer-based training committee) and targets interoperability of CMI systems. It includes extensive information to enable the integration of trainings in learning management systems.

Sharable content object reference model

SCORM (shareable content object reference model) is a collection of specifications aiming at interoperability, accessibility and reusability of Web-based learning content. It is a recommendation of the advanced dis-

[14] These learning objects should not be confused with the learning objects described below.

tributed learning initiative (ADL) and is used as a reference model to define requirements for e.g, the LOM specification.

Learning objects. An important success factor for e-learning in organizations is to provide ways to efficiently design and update learning units that aim at transferring knowledge that is highly relevant for daily operations on the workplace.

Lifetime

In contrast to learning units that aim at transferring knowledge about fundamentals that are stable over a long period of time, this knowledge can be characterized as fast changing. Thus, extensions, updates and enhancements of learning units need to be brought to learners as quickly and efficiently as possible. This is the main aim of the concept of learning object.

Definition of learning object

A learning object is an ICT-supported structured collection of different types of contents that was designed with the goal to facilitate learning about a clearly focused topic. It provides functions to support flexible extension and enhancement and ideally is a part of the organizational knowledge base.

Support by EKI

Learning often happens ad-hoc, bound to specific situations and on the job, i.e. during daily tasks. The boundaries between using EKI for searching, retrieving and applying knowledge elements for business tasks and learning about new topics are blurry. Thus, learning objects represent the vision of flexibly incorporating knowledge elements that are created as by-products of daily operations, especially when exceptions are handles. These knowledge elements are targeted specifically at learning supported by an integrated ICT platform. In the following, we will discuss the characteristics of learning objects.

Flexible

The content of learning objects must be presented in a flexible way to learners. The EKI must provide means to regards previous knowledge, skill level and other personal characteristics (e.g., interests in specific domains, learning goals) as well as the actual work context. The contents of a learning object thus need to be manually or automatically enriched by meta-data and stored in a repository to flexibly transform, process and thus personalize and publish it for access over different information channels.

Manageable topics

Learning objects should be focused on a manageable topic that allows to include meaningful and concrete elements and to make them reusable. If the topic of a learning object is chosen too general, there is the danger that contents remain too general or the object becomes too big so that learners have no possibility to overview and get a grasp of the topic. The view on what level of granularity is suited depends on the perspective of the actual users. Examples for topics covered by learning objects are “success factors and barriers of KM”, “creating skill profiles for KM”, “maintaining electronic community spaces for KM” and “success of KM”.

Life-cycle

Learning objects are created, grow, mature, might be frozen or split into more detailed learning topics. This inherently depends on their topics, learners and their relevancy for the organization.

Structured

Learning objects have a generic structure that can be changed and especially detailed depending on the needs of the actual topic. Parts that explain theoretic concepts need to be related with parts that provide concrete examples and case studies and especially links to experts, examples or products of the organization. Moreover, learning objects should enable learners to test their knowledge by providing questions for self-evaluation exercises and case studies. To sum up, a learning object should be structured as follows:

- Introduction: motivation and challenges,
- contents: approaches, concepts, methods, theories, empirical results,
- tools,
- examples: case examples, implementations, prototypes,
- case studies,
- experts and institutions,
- market overview,
- questions for self-evaluation,
- exercises,
- sources like references to literature, links to other sources, URLs and
- current projects, dissertations and studies.

ICT-supported

Learning objects are necessarily implemented with the help of ICT since they are the foundation to flexibly change, update and transform them and to present contents flexibly, personalized and context-dependent to the learner.

Example: learning objects in Open Text Livelink. An easy and straightforward way to implement a learning object is to set up appropriate structures and rules within a document-oriented KMS, e.g., in the system Open Text Livelink. The basic structure is implemented by creating folders and sub-folders. Permissions are assigned on object level and thus detailed rights to access, change or add contents or structures can be assigned to groups of learners, tutors and teachers. Every object can be described by customizable meta-data that allows to link it to expertise levels, skills or other topics. Pre-defined search queries (e.g., “show all elements containing a general introduction”, “show all case studies”, “what’s new this week”) allow to access the objects independently of the storage structure. The learning object can contain links to other knowledge elements of the system not specifically designed for learning, e.g., documentations of products, scientific papers, guidelines and forms. Meta-data and content can be exported in XML markup and transformed with XSLT (section 3.1.5, 162ff) to generate alternate views or maps of the learning object.

The learning object can be enriched with collaborative functions like recommendations of objects to other users, discussions, task lists, etc. A full-text search engine allows to search the content based on keywords and meta-data.

Altogether, learning objects might be a promising way since implementation of learning objects is easy and straightforward. However, the system does not support functions specialized on learning like definition of dynamic learning paths, learner models or examinations.

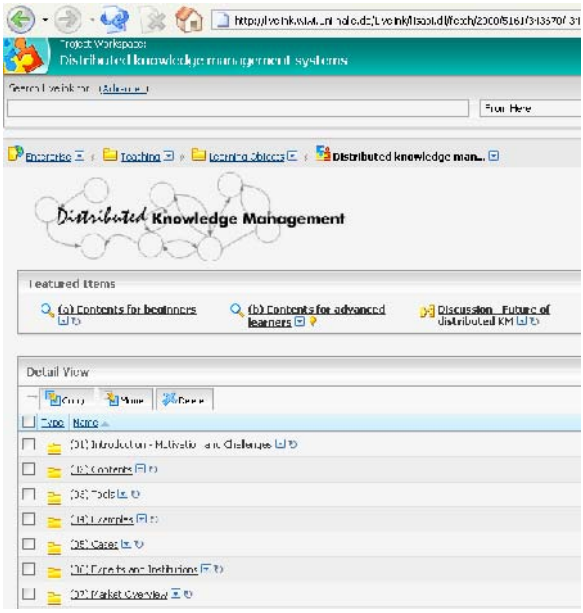

Figure 4-19. Screenshot: Learning object implemented in Livelink

4.4.3 Learning

Related system classes

There are multiple system classes for computer-based support of learning:

- *Learning management systems* emphasize the administration of courses, learners and course materials of the learning process rather than creation and use of learning contents, e.g., by providing calendars, news and workspaces for distribution of electronic documents for courses held on-site at university.
- *Assessment tools* allow for designing and deploying tests, quizzes and questions. They can be integrated into other learning systems to complement their functionality.
- *Collaboration tools* aim at supporting collaboration between teachers and learners and among learners. Their functions are discussed in section 4.3, 267ff.

- *Platforms for synchronous learning* support simultaneous communication supported by whiteboards, application sharing, ad hoc-questions, co-browsing and file sharing.
- *Web-based platforms* are closely related to Web content management systems (WCMS). In fact, they can be regarded as WCMS specialized for the domain of learning and sometimes are called learning content management systems.

We define e-learning platforms broadly and as being characterized by five functional areas. In the following, these areas guide the discussion of functions provided by an e-learning platform.

Definition of e-learning platform

E-learning platforms or suites provide an integrated environment for the organization-wide management of learning modules and for computer-supported cooperative learning. They provide functions for (1) creating learning contents, (2) presenting and using them, (3) for communication and collaboration between participants, (4) to evaluate learners and (5) to manage learners, courses, contents and learning processes.

Creating learning contents

(1) Creating learning contents can happen either with support of different classes of authoring tools or with the help of functions that the e-learning platform provides. This depends entirely on the way how e-learning should be implemented, e.g., either as a complex application that displaces learning on site in a class or as a system targeted at supporting only specialized tasks of the learning process like organization of courses in the context of a blended learning approach. Many vendors of e-learning platforms acquired smaller specialized companies to enrich their platform with authoring functions. Creating learning units comprises planning and structuring of courses, creating electronic learning units, planning resources and promoting the learning unit.

Presenting and using contents

(2) Many e-learning platforms are based on Web technologies. Participants access the system with an Internet connection and a standard Web browser. Offline learning without a direct connection to the learning system requires special software client to synchronize contents as well as other data like results of learner evaluations or profiles gained by observing the learners' actions. Teaching and learning usually happens with the help of virtual classrooms. Additionally, learners have individual workspaces to maintain their personal information like bookmarks, contact lists, course entries and documents. Interfaces to other systems allow to access and import data like news, literature references and contact information.

Communication

(3) Communication functions allow for synchronous or asynchronous communication between participants like email, blackboards, whiteboards, chat, audio or video conferences. Advanced functions aim at relating communication to the contents presented, e.g., to jointly solve given tasks and

create an appropriate solution. Awareness functions give feedback about actions and status of other participants, e.g., degree of activity in a blackboard, users currently online, changed contents, etc. Communication functions should be regarded as an important way to spark involvement and activities of learners.

Evaluation

(4) Evaluation of learners aims at feedback about their learning progress. Descriptive feedback and showing learners ways how they could improve their skills should be preferred in place of short feedback that just enumerates failures. Naturally, this is far more complex to implement. Examinations go one step further by incorporating the goal to certify learners. Online examinations head for a computer-based realization of examinations with the goal of making the evaluation process more efficient for educators. Most straightforward ways for implementation is a direct evaluation of learners, e.g., after they worked through a learning unit.

This can be realized by

- single-choice questions,
- multiple choice questions,
- free-text fields for short statements,
- free-text fields for long statements and essays

as well as a rich variety of tests like

- clozes,
- tasks to relate and
- tasks to sort items.

More advanced ways for evaluation and especially to personalize learning contents and provide feedback during the learning process are complex internal models of the learner that are used to generate suggestions for contents to be presented and the next steps on the learning path.

Administration

(5) Administration usually happens decentrally for tasks like maintaining the contents of courses or defining access privileges for special groups and/or roles of users. Other tasks are usually centralized, e.g., technical tasks like systems maintenance, backup and maintaining connections to other systems. Learning platforms can provide programming interfaces to extend and customize their functionality. This often is deployed by specialized consulting companies or consultants of the product vendor.

Example: Stud.IP at the University of Halle-Wittenberg. Stud.IP is an open source learning management system that provides extensive functions for students, scholars and administrative staff to support activities around management of as well as communication and collaboration around on-site university courses. Examples for functions are discussions, schedules, document folders, chat, wiki-Webs and news to support courses, interfaces to other systems like library catalogues, personal information management functions like address book, calendar, individual

schedules as well as managing courses and resources with the help of an integrated room management, event timetables and staff directories.

The system was introduced in winter term 2003 at the School of Business and Economics and in February 2004 throughout the entire University. Integration with other initiatives aiming at creating ICT support for administrative tasks in the areas of university-wide planning and information management (e.g., personnel planning, management of student records and grades) was seen as a success factor. Since only a standard Web browser is needed to access the system, efforts of technical introduction were limited to installing the system on a centralized server.

Half a year after introduction, 848 courses of 320 institutes are hosted (October 2004). During the semester, about a quarter of all registered students uses the system regularly once per working day. It is used for room reservations and as a foundation for the creation of the online and published version of the directory and timetable of lectures. Currently, teachers apply it to announce news and timetables, manage enrollment in courses, distribute electronic documents and for communication between teachers and students in discussion forms and over the Stud.IP mail system. Students organize their studies with the help of the system and use it for communication with peers as well as for uploading student reports. The scoring system that rewards users with points for continually applying system functions led to considerable attraction, because many users headed for top places on the internal position table.

Figure 4-20 shows the personalized *MyCourses* page of Stud.IP which provides a good overview of the current state of all courses that the teacher or student is assigned to.

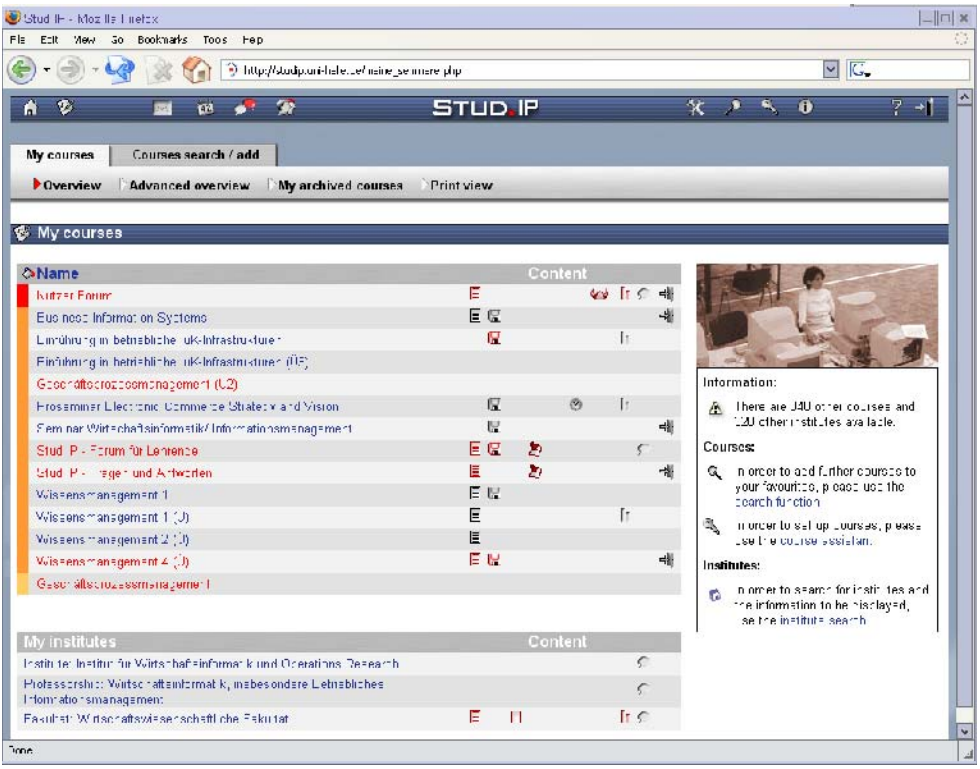

Figure 4-20. Stud.IP at the Martin-Luther University of Halle-Wittenberg

Questions and Exercises

1. What are differences and similarities between search for Internet resources and for resources in an organization's Intranet?
2. Find examples for cases in which on the one hand search by keywords and on the other hand by navigation and browsing visualizations of resource collections is most appropriate!
3. Draw an example for a knowledge map that shows which types of knowledge and resources are needed during the phases *design of questionnaire*, *data collection*, *data analysis* and *creation of final report* of a marketing research company. Give concrete examples how this map should be generated and applied!
4. A medium-sized company in the automotive industry wants to introduce a document management system. Typical types of documents are application for leave, construction plan, internal mailing to all employees, invoice, e.g., for office materials, and long-term contract with customers. Analyze which document types should be managed within the DMS! Therefore, explicitly name decision criteria and use them for classification of documents! Decide which documents should be held online, nearline and offline!
5. What is the difference between non-coded information (NCI) and coded information (CI) and what are advantages and disadvantages of each type for archiving documents?
6. Compare Groupware, document management and e-learning systems with regard to (a) goals, (b) focus on supported tasks, (c) role of users and (d) characteristics of system contents! Draw a conclusion on main similarities and differences of these system classes! Classify managed contents according to the dimensions of knowledge presented in section 1.1, 3ff!
7. What are the similarities of and differences between content and documents and document management and content management systems?
8. Describe the historical development of WCMS, discuss the supposed benefits of additions in each phase and give an outlook on further developments!
9. Describe a concrete example of a communication process structured according to the Shannon/Weaver model!
10. Draw a conversation network that structures the following conversation at the desk of a library: A student comes to the University library and says: "Hello, I would like to extend the due date of these books." The clerk at the library desk might answer "Sorry, but this is not possible. These books are ordered by another person." or "No problem, I have postponed the due date for two weeks." If the books have already ben

ordered by someone, the student could try to get at least two more days. If that is also no option, then the library clerk can at least offer an extension of 24 hours. If there was no extension possible at all, the student returns the books. In either case, he says goodbye before he leaves.

11. Give a concrete example how ICT can support a conversation that was structured with a conversation network! Which classes of systems are most suited to support structured conversations?
12. Give a definition of the term CSCW and name system classes that are important in this area!
13. List and explain functions of an e-learning platform required to support University courses following (a) a blended learning and (b) a distance learning approach!
14. Characterize how e-learning modules should be designed from the perspectives on learning of behaviorism, cognitivism and constructivism!
15. Explain principles of designing learning units and multimedia content! Provide a checklist for software developers which should guide them in actually designing contents!
16. Give an overview of the most important e-learning standards!
17. Describe how learning objects are structured, can be supported and discuss their potentials with respect to reuse!
18. Define e-learning platforms and describe their main functions!
19. Make at least three proposals for extensions that substantially enhance the system Stud.IP at the University of Halle-Wittenberg (studip.uni-halle.de)!

Further Reading

There are a number of articles in the area of knowledge management systems that target specific aspects of knowledge services. The following books give an overview of some of the more basic foundations of knowledge services.

Asprey, Len; Middleton (2003): *Integrative Document and Content Management: Strategies for Exploiting Enterprise Knowledge.* Idea Group Publishing, Hershey 2003.

The first chapter of this book outlays the architecture of an integrated document and content management (IDCM) and gives a brief historical overview over the historical development of document and content management systems. The features of the IDCM are examined briefly in the following chapters.

Baeza-Yates, R.; Ribeiro-Neto, B. (1999): *Modern Information Retrieval*. Addison Wesley, Harlow

This book comprises in-depth explanations and comparisons of concepts and techniques for all important phases of information retrieval, i.e. preparation of resources, indexing, query operations, searching and presentation of results. Additionally, they illustrate how searching in special content types (multimedia IR) and in special domains (digital libraries) can be realized.

Boiko, Bob (2002): *Content Management Bible.* New York, Hungry Minds 2002.

This book presents a comprehensive overview over content management and should be seen as a practical guide for doing content management with a focus on Web content management. On nearly 1000 pages content, content management, content management projects, the design of CMS and building as well as implementing a CMS are discussed.

Borghoff, U.M.; Schlichter, J.H. (2000): *Computer-Supported Cooperative Work: Introduction to Distributed Applications*. Springer, Berlin

This book includes all fundamentals about collaboration services. It extensively explains basic principles and concepts of Computer-Supported Cooperative Work (CSCW) like history and goals or concepts of asynchronous and synchronous communication as well as applications to support CSCW like communication systems, shared information spaces, workflows, workgroup systems and multiagent systems.

5 Access Services

Overview

The knowledge services discussed in the previous chapter should not be isolated from other applications a knowledge worker uses. On the one hand, users want access to them from within their usual work environment and they want access any time and any place. Therefore, mobile access and desktop-integrated access play an important role for utilization and user acceptance of EKI. On the other hand, there are some demands imposed by the need to run these infrastructures cost-effectively.

This requires solutions that run on any desktop, independently of the operating system and the installed applications. In addition to that, control over installed versions and software use is required to manage updates and provide efficient user support. These two demands lead to a variety of alternatives that are either quite well established like browser-based application access or just emerging such as rich thin client technologies. Ultimately, users want a single point of access to all services offered within the EKI, so that they are easy to use and can conveniently be accessed using single-sign-on. From a cost and administration perspective, central user management and profiling capabilities are beneficial (section 3.3, 198ff).

Access technologies are classified into server-based technologies, e.g., portals and server-side personalization technologies which are presented in section 5.1. Client-side access technologies can further be divided into thin clients, rich thin clients, rich clients and desktop integrated access and are presented in section 5.2. Finally, section 5.3 deals with the basic requirement of mobility which is typical for knowledge work (section 1.2, 24ff). After studying mobile computing in general, the specifics of mobile devices and mobile application are discussed with respect to their differences to more traditional access to EKI from desktops.

Learning objectives

On completion of this chapter, you should be able to

- define server-side, client-side and mobile access technologies,
- analyze potentials of each of those classes for accessing EKI,
- employ the concept of personalization to tailor knowledge services to the needs of participants,
- define portals and analyze their main functions and application areas,
- assess the overall contribution of access services to usage and user satisfaction when implementing EKI.

5.1 Server-side Access Technologies

Users of an EKI deal with a variety of applications, systems, services and contents. In practice, this usually leads to challenges since they continuously need to switch between multiple systems that regularly require logon with dedicated user names and passwords, have different interfaces and different ways to select and present data, information and knowledge. In this context, single point of access means integration of these various sources into one single interface with a consistent look and feel. Furthermore, there is the danger of "information overload" as access to too much information and knowledge might decrease productivity.

Overview

These requirements usually are met by means of Web-based portals which we will discuss in section 5.1.1. One of the main features of portals are selection and filtering of contents according to the user's individual needs and requirements. This is denoted as personalization and will be discussed separately in section 5.1.2.

5.1.1 Portals

Portals as starting points

The term *portal* denotes a monumentally designed entry to a building. It is used as a metaphor for Web interfaces that integrate various heterogeneous information sources and applications on a single Web page. Simple, Web-based portals are thus one way to realize integration on the presentation layer (3). The metaphor references an important characteristic of Web portals: They are designed to provide a central entry point for users to application systems, just as doors provide to buildings. However, they do not represent a work environment, rather they are designed to provide an overview of and integrated access to various information sources and systems (represented in so-called *portlets*) and to guide users to those systems they need to work with, just as the hallway behind a portal leads to rooms and chambers where people live and work.

Internet portals

The term portal is used differently in the context of the Internet and the Intranet/Extranet. Internet portals act as a major starting site for users in the Web or a site that users tend to visit as an anchor site. They provide a structured collection of links to Web sites and are often coupled with additional services, e.g., email accounts, chatrooms and calendaring. There are general-purpose portals comprising a large variety of topics and specialized niche portals focusing single topics and supporting communities in sharing information around this topic, e.g., travel, cycling, rock-climbing or knowledge management. Examples for general-purpose Internet portals are www.t-online.de and www.lycos.com. Single topic portals are e.g., www.contentmanager.de, www.kmworld.com and www.xml.com.

Enterprise portals

Portals available in the Intranet or Extranet denote the actual Web site that unifies access to different applications and also a software application

running on a server that provides functionality to achieve unified access. We will use the term portal in the latter sense. Portals of this type are also called enterprise portals.

Portals and EKI

However, sometimes Intranet portals are understood in a broad sense that subsumes several related components including interfaces to integrate the source systems, middleware and the underlying infrastructure. This includes many services of the EKI discussed in previous chapters, especially integration services (chapter 3, 147ff) as well as application server (section 2.4, 136ff). However, portals are just *one* way to access selected services and systems of an EKI. Employees regularly will use an EKI in many different ways and settings, e.g., as extensions to their desktop, with the help of mobile devices or in shared workspaces.

Portal tasks

Put briefly, portals are designed to bundle

- *personalized*, i.e. according to user's needs,
- *task, project or role-specific*, i.e. targeted at the actual working context of the user,
- *location-independent*, i.e. accessible with all devices able to present Web content including mobile equipment,
- and *secure*, i.e. through encryption of sensitive data as well as user authentication,

access to

- *software functions*, i.e. functions offered by different systems supporting business processes for employees, partners and customers, e.g., employee self-service functionality,
- *data, information, and knowledge*, i.e. structured as well as unstructured data with context and meta-data from multiple information sources.

Functions

Portals ideally provide the following functions: personalization of contents and systems, single sign-on to several systems and integration of systems on the presentation layer.

Personalization

Personalization denotes the ability of the portal to adapt to the specific needs and requirements of its users. This at least comprises the possibility for users to choose which elements are presented where on the Web interface and to customize the general appearance (e.g., colors, icons, decorations) of the portal. Advanced personalization includes user profiling as well as automatic adjustment and filtering of contents and functions (section 5.1.2).

Single sign-on

Users usually need to authenticate separately on every single system they use. Portals relieve users of this task by automatically transferring the data needed to systems that are accessed through the portal. In fact, this feature may be one of the major driving forces and reasons for the widespread use of portals. The necessary user-related data such as user name and password is managed and provided by account or identity management services (section 3.3, 198ff).

Drag & relate

Portals display user interfaces of various applications in a single consistently designed user interface. Currently, there are ongoing developments for a tighter integration of application systems with portals. Most integration efforts aim at data integration, specifically to ease data transfer between different applications accessed through the portal. An example is the drag & relate technology from SAP. Users can transfer single data items between portlets by selecting them (e.g., sales order number 537), dragging them to another portlet (e.g., order portlet) and thus invoke a function of the target application that uses the data item as parameter (e.g., details of sales order 537 are displayed). SAP calls this functionality that integrates heterogeneous systems drag & relate in analogy to drag & drop on the desktop.

Classification of portals. Various classes of portals can be distinguished depending on target group, primary content and area of application.

Target group

The group of users that predominantly accesses the portal can be (a) *employees* that use a B2E (business-to-employee) portal providing e.g., knowledge services, employee self-services or services that act as a channel for internal communication and marketing, (b) *business partners* such as suppliers or partners in a joint venture, that access a B2B (business-to-business) portal, e.g., to execute business transactions or to collaborate in joint projects with the organization hosting the portal, and (c) *customers*, targeted by a B2C (business-to-consumer) portal, e.g., an online shop for product sales or Web pages for advertisement and customer support.

Content and area of application

Another alternative to classify portals is according to their primary contents and areas of application into:

- *Enterprise application portals* that represent an interface to heterogeneous types of applications, e.g., enterprise resource planning systems, geographical information systems or customer relationship management systems,
- *enterprise information or knowledge portals* that supply information from various data and information sources like data warehouses, file servers, document management systems, report engines or external news feeds, as well as EKI services to handle knowledge, and
- *enterprise collaboration portals* that provide virtual spaces for task-oriented collaboration of teams spanning organizational units.

Design and implementation of portals. There is a rich variety of different ways to design and implement Web pages that represent the user interface of a portal. Internet portals often try to attract users with an appealing layout. Portals used in a business context usually are designed more function-oriented, though issues such as a layout that fits and communicates corporate identity are often considered important.

Portlets

Figure 5-1 depicts an example of a typical portal Web site applied in organizations (based on Priebe 2004). Portlets group functions, contents or

navigational elements similar to windows in graphical user interfaces of operating systems. Portlets display their title and functional elements (e.g., minimize portlet, print) within a headline. Below this headline, contents are displayed in the portlets' content area. Each portlet may be an interface to a different system. They encapsulate specifics of generating output for involved systems, are therefore directly connected to the systems although they run in the environment of the portal server. Many software vendors have already implemented portlet interfaces for their applications so that these applications can be easily integrated into a portal.

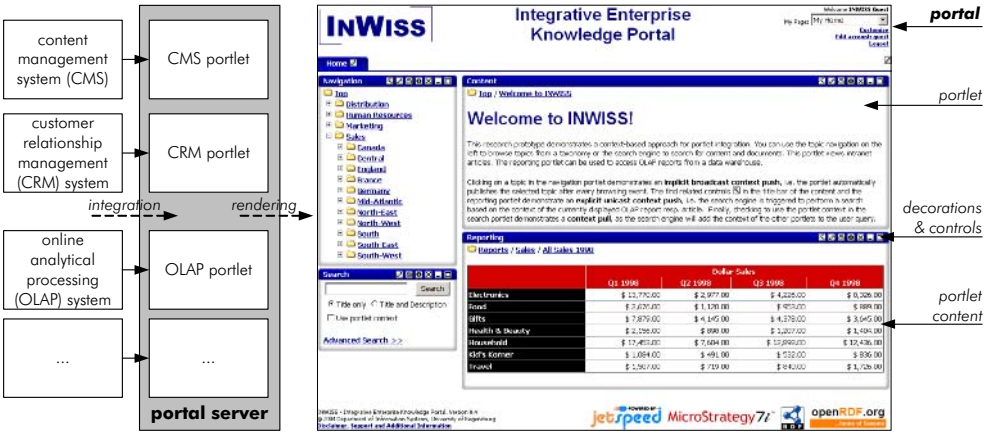

Figure 5-1. Portal server, portal and portal components

Portal server

Usually, it is the task of the portlet to process and display contents of the connected system within the content area of the portlet. A portal server integrates the individual contents of each portlet, adds the portlets' headlines together with their functional elements and produces the Web site around the portlets. It handles general interaction with users and communication between portlets (e.g., functions such as drag & relate). Product examples for portal servers are IBM WebSphere Portal and SAP Enterprise Portal.

Portlet standards

Currently, there are two standards worth mentioning in this context: The *Java Portlet Specification* is based on the Java 2 Platform (Enterprise Edition, J2EE) and defines the inner structure of portlets and interaction between portlet and portal server by means of Java interfaces that need to be implemented by the portlet. Naturally, the standard does not specify the contents that are rendered in the portlet. The *Web Services for Remote Portlets (WSRP)* standard defines Web Service interfaces for portlets that may run remotely outside the portal server. In this case, portlets can be implemented on different platforms, e.g., on J2EE or on Microsoft .NET.

Complexity of implementation

The implementation of a portal can easily rise to a considerable level of complexity, if several user groups and systems are involved. An example of a B2E portal of a large car manufacturer gives an impression of the complexity (Amberg et al. 2003): A typical user accesses 50 portlets per day that are connected to various systems. Some of these portlets are displayed per default, others can be individually selected and configured by users. The company has 7 business units, each composed of 15 departments that each have 10 user groups on average. This results in 1,050 user groups. The portal provides access to several thousand portlets, some providing rich functionality. This complexity is independent of the technological implementation, e.g., the type of the portal server that is used.

5.1.2 Personalization

In a broad sense, personalization encompasses adapting formerly standardized services to specific needs of individuals or groups. This definition includes adapting the behavior and services of a computer system, customizing products to customer needs as well as individualized marketing and customer services (one-to-one marketing). In the context of EKI, personalization denotes adapting contents and functions to users' needs. Personalization can be based on individual characteristics, such as skill level, interests or preferred media as well as on characteristics of users' actual work contexts, such as processes they are involved in, their location, time and technical requirements, e.g., work with a mobile device, and their current tasks. To accomplish this, the system needs to have an internal representation of relevant data about the user, a so-called user model.

Profiling

For efficient personalization, the model needs to be continuously updated by profiling data about users and their actions. Figure 5-2 shows the process of profiling and the subsequent application of the collected and analyzed profiles to personalize EKI. Grey arrows visualize data flows to or from the participant. Black arrows visualize the sequence in which steps are taken.

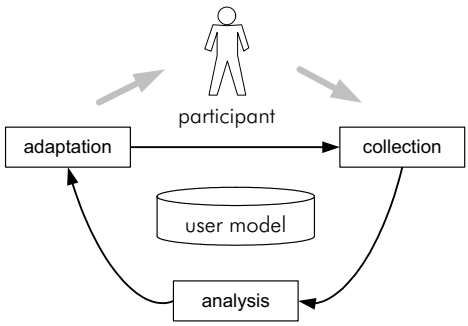

Figure 5-2. The process of profiling (based on Frielitz et al. 2002)

Application of user models

Creation of a user model can be understood as acquiring, formalizing and managing knowledge about users. The success of automated personalization depends on how accurately user models allow to determine actual user interests and information needs. Principally, a user model can be used to

- *diagnose and evaluate* knowledge and skills of a user, e.g., with an e-learning system that applies learner models to adapt the type or media of contents presented (section 4.4, 286ff) based on explicit user evaluations or implicit user profiling,
- *forecast* user actions and reactions by analyzing paths of development, though this regularly is a complex task since there may be multiple interpretations of one and the same user behavior, e.g., short response times may either indicate that the user is under-challenged or that his concentration decreases,
- *filter* contents and other outputs of the system, e.g., search results, system dialogues or information presented in a portal site,
- *improve* incorrect or insufficient user input by correcting or complementing it based on supposed goals of the user,
- *notify* users and *recommend* them topics, news, issues that may be of interest for them,
- *assign tasks* or *route requests* to users based on, e.g., their expert level, skills in certain domains or their current work load which may be supported by a workflow management system (section 3.4.2, 210ff).

Characterization of user models

Table 5-1 gives an overview of attributes that allow for a general characterization of user models (Mertens/Höhl 1999). The *subject* of the user model may be either the individual user or a role, group or organization the user acts for as an intermediary. For example, in a purchasing department, characteristics and requirements of the employee's organization may be more important than his individual preferences. *Individualization* may happen either on the level of individual parameters per user or by relating users to defined classes, called stereotypes. For example, collaborative filtering (section 4.3.2, 276ff) may recommend items to users based on categorization into groups that are representative for their interests. Another example is selection of contents according to the expert level of the user, e.g., novice, advanced beginner, competent, skillful master or expert (section 1.1.1, 4ff). User models can be based on different *types of information*. Capturing hard facts, e.g., gender, number of errors, invocations of user help, user response times, is substantially easier than collecting and storing "soft" information about the user, e.g., goals, aims, plans.

Dynamic models can be changed and thus evolve during a user session while *static models* remain unchanged. Systems with *transparent* models allow users to inspect what information is stored about them, sometimes even to change it. This may be undesirable as it could bias user behavior and lead to erroneous classifications. A user model can be either *valid* for

multiple sessions or only for one session. *Acquisition of information* about users may happen automatically by machine learning, e.g., by neural networks, or by manual addition of rules.

Table 5-1. Characteristics of user models

attribute	**values**		
subject	receiver (e.g., role)	vs.	user
level of individualization	individual	vs.	stereotype
type of information	soft	v.s	hard
dynamics	static	vs.	dynamic
transparency	transparent	vs.	intransparent
time of validity	multiple sessions	vs.	one session
acquisition of information	manual	vs.	automated
collection of information	explicit	vs.	implicit

Collection

Finally, the *collection* of information can be: *explicit* with the help of a number of questions that users have to or should answer, e.g., gender, age, topics of interest, area code, *implicit* by observing user behavior and monitoring interactions with the system, e.g., user tracking or click stream analysis, or based on a *combination* of data collected from other systems, e.g., enterprise resource planning systems or human resource management systems.

Data privacy

Collection of information about users may potentially harm personal rights and usually is regulated by a number of laws. In Germany, data collection, storage, processing and use is regulated in the federal data protection act (in German: Bundesdatenschutzgesetz, BDSG), data protection acts of the federal states as well as the information and communication services act (in German: Informations- und Kommunikationsdienstegesetz, IuKDG). Article 2 of the teleservices data protection act (in German: Teledienstedatenschutzgesetz) establishes the basic principle to store, process and use no data about persons or as little data as possible. Users principally need to explicitly agree to further use of their data. To guarantee compliance with laws, data should be processed anonymously where possible, e.g., only related to roles or groups of users. Legal regulations limit application of advanced user models. Users may be more willing to accept that systems collect and use information about them, if they see benefits, provided they know what information is stored and what happens with it.

Analysis

Analysis of the collected information includes

- *data mining*, e.g., the selection, cleansing, transformation and analysis of relational data, e.g., skill or interest profiles, in analogy to data warehouses and customer relationship management systems,

- *text mining*, e.g., the analysis of submitted documents or of contributions in newsgroups, and
- *web mining* (or click-stream analysis) which is a technique to analyze data about user movements on a Web site, an Intranet platform or Web-based systems of an EKI. It is widely applied and supported by a number of commercial products in B2C electronic commerce.

Web mining

In a personalization context, Web mining is used to discover navigation paths ("foot prints", "trails") through Web sites, evaluate which links are preferably clicked, or how long users stay on a Web site. It may help to discover correlations between contents and layout of a site, on how they influence user actions, e.g., purchase orders after visiting certain sites, preference of links according to their position on the Web site, and time spent on a Web page.

Adaptation

There are two basic types of adaptation: customization and personalization. Customization is under direct user control. The user explicitly selects between different options, e.g., the portlets displayed on their individual portal entry page. Personalization is driven by the system which delivers individualized pages to users based on user models.

Adaptation can be accomplished with pre-defined "if-then" rules that regard data, roles, membership in groups and communities of the user as well as with event- or time-driven triggers. The problem is that designers need to anticipate users needs which may change over time and according to the actual usage context. Other ways are automated content-based filtering, e.g., by comparing user profiles with contents of the knowledge base, and automated collaborative filtering (section 4.3.2, 276ff), e.g., "communities of preference", active recommendations by other users as well as automated or hidden recommendations.

Examples for adaptation in EKI systems are:

- search results that are filtered according to users' information needs,
- contents of learning units and evaluations that are selected and adapted to users' skill levels and user preferences,
- recommendations of items to develop users' skills in their areas of interest,
- wizards that guide users through system functions according to their level of experience with the system, and
- dialogues that provide information appropriate to the technical skills of users, e.g., instructions, error messages.

Critique to personalization

Personalization is based on the assumption that information needs of users can be identified and explicitly represented by means of a user model. This may not always be possible satisfactorily, since in many cases either needs can not be explicated or user models become too complex. Other issues are capturing and processing of sensitive user data. In some cases, users may not appreciate filtering, e.g., because they do not trust fil-

tering methods and fear to overlook items. On the contrary, they may not recognize important items at all if contents are not filtered. Some sort of basic filtering is almost always applied, e.g., spam filtering in email systems or basic rights management in knowledge infrastructures, however, this is not individualized. Altogether, personalization is a powerful, yet complex way to support individual user needs. Its success largely depends on how much effort is taken to create appropriate models of users which needs to be balanced with the benefits of personalization.

5.2 Client-side Access Technologies

Access to enterprise applications in general and EKI in particular should be as easy as possible. Thus, the ideal solution would be a single software platform that is able to support the whole palette of mobile and stationary devices regardless of what operating system runs on them. Given the different capabilities of the devices, this can currently not be realized. Many applications still require the installation of a platform-dependent software client. The only help for system administrators is the use of software deployment solutions that support remote installation of software. However, these tools are mainly platform-dependent and programmers are still left with the demand to supply clients for all main platforms. In the following sections, three different approaches are presented to overcome these limitations. They attempt to make it easier for software developers to create different graphical user interfaces (GUI) from the same source code and help administrators to automate updates and centrally control program versions used throughout the company. This is required so that services offered in EKI can efficiently be accessed by a multitude of access applications and appliances.

5.2.1 Thin Clients

HTML clients

A first approach uses Web browsers without any additional software as runtime environment on the client. The advantage of this approach is that applications run on every platform that offers a Web browser and that no specific client software has to be installed. However, this means that programmers can only use HTML for describing GUI elements and application logic has to reside on the server. That leads to a large number of so-called round-trips.

Inefficiencies

A *round-trip* is the cycle from (1) sending data from a client to a server, (2) evaluating and processing data on the server and (3) sending back the refreshed user interface to the client. Consequences of many round-trips are slow applications and high network traffic. Applications are compara-

bly slow in this inefficient approach, because server processing capacity per user is typically smaller today than client processing capacity. Network traffic is increased, because a large percentage of data transferred from server to client is identical for two consecutive round-trips. Since HTML pages for Web applications are dynamically generated on the server, they cannot be cached on clients. In addition to that, HTML was not designed to describe application user interfaces and therefore is limited in its capabilities.

DHTML

Script languages like JavaScript, JScript, VBScript or ECMAScript can be used to put some application logic on the client side. These script languages are mainly suited to evaluate user input for syntactical correctness, so that only syntactically correct and complete forms can be submitted to the server for processing and some round-trips are saved. In addition to that, some formatting information can be defined using cascading style sheets (CSS) so that less HTML code has to be transferred as the static CSS data can be cached on the client. Web pages that use both, script languages and CSS, are commonly denoted as dynamic HTML (DHTML) pages. DHTML overcomes most of HTML's limitations regarding GUI rendering. For example, dynamic menus are possible as well as dynamic tree views.

CSS

CSS is a language that supplements HTML to define layout-oriented properties of HTML elements. The properties defined in the CSS language include text fonts, colors and sizes as well as margins, alignments and adjustments just as borders, background colors and images. They can be specified within the `style` attribute of an HTML element, within the `head` of an HTML file or within a separate CSS file that can be referenced by several HTML files. Separate properties can be specified for different output media. Examples for important media types are `screen` (for PC clients), `handheld` (for clients with small displays) and `print` (for data that has to be printed). Listing 5-1 shows an example of styles for the body of an HTML page (`body`), for first-level headings (`h1`), paragraphs (`p`) and list items (`li`) specified in the CSS language.

```
body { background-color:#FFFFCC; margin-left:100px; }
h1   { font-size:48pt; color:#FF0000;
       font-style:italic;
       border-bottom:solid thin black; }
p,li { font-size:12pt; line-height:14pt;
       font-family:Helvetica,Arial,sans-serif;
       letter-spacing:0.2mm; word-spacing:0.8mm;
       color:blue; }
```

Listing 5-1. Example for a cascading style sheet[1]

[1] URL: http://en.selfhtml.org/css/, last access: 2005-02-04

DHTML drawbacks, DOM

However, applicability of DHTML also has its limits as different Web browsers use different ways to address HTML elements. An HTML document can be represented as hierarchical tree of elements. The scripting language can access every single element to read and write its properties and values. Browsers provide a programming interface called *document object model* (DOM) to enable that. DOM implementations are slightly different in major browsers. Therefore, developers of enterprise applications often have to create multiple versions of the client (e.g., for Microsoft Internet Explorer and Netscape / Mozilla browsers).

XSLT generated XHTML

Another approach that also uses HTML on the client-side leverages XML and XSLT to generate HTML pages (section 3.1, 149ff). This technique is especially interesting if the client is capable of transforming the XML file into HTML (or XHTML to be precise which is the redefinition of HTML using XML as meta language). Client-side transformation leads to significantly reduced server and network load as only data that has to be displayed is transferred in every round-trip. However, capabilities of different browsers are again diverse, so that client-side transformation is no real option. An advantage of the XML approach with server-side transformation in contrast to HTML solutions is that different versions for different browsers or client devices can be created relatively easy by using slightly different XSLT style sheets.

XUL, XAML

Recent approaches use XML-based languages to specify the GUI of an application including layout and elements. Two corresponding languages are XUL (XML User Interface Language), which is developed and supported by the Mozilla foundation and XAML (XML Application Markup Language) which is developed by Microsoft and will be used with forthcoming versions of the .NET platform in future Windows versions (codename Longhorn). The advantage of this approach is that complex client GUIs can be built quite easily, and the same XML files can be used for Web-based applications as well as for rich clients. The amount of data that has to be transferred is much smaller than with traditional HTML clients, as the entire GUI description only has to be transferred once. Simple checks can be done on the client-side with scripting languages. But again, there are at least two competing industry standards so that it will not be possible to write an application that runs on both, Mozilla and Internet Explorer. On the one hand, Mozilla has a bit of a head start as XUL is working today, whereas XAML will be introduced together with the next Windows generation in 2006. On the other hand, XAML is more comprehensive as it includes not only elements to describe GUIs, but also elements of SVG and SMIL (Synchronized Multimedia Integration Language) to describe vector graphics and animations.

JSF

Java Server Faces (JSF) is a technology that corresponds to XUL and XAML, but on the server-side. JSF extend Java Server Pages (JSP) and define a set of GUI elements that are converted to HTML, before the Web page is transferred to the client. Thus, they help programmers to design GUIs, but do not ease the problems with high server and network load.

5.2.2 Rich Thin Clients

Plug-ins

Rich thin clients use Web browsers extended by plug-ins as runtime environments. The advantage of such an approach is that plug-ins provide a richer set of GUI elements and offer additional capabilities to implement client-side application logic. Most plug-ins are available for all major browsers, so that deployment across platforms is possible. Network load is minimized through client-side caching, so that typical operations like resorting data or updating calculated fields do not require fetching data from servers again. Functionality like drag&drop, context menus and shortcuts are also possible. Unfortunately, plug-ins have to be installed on all clients and often require considerable amounts of time to load.

Java applets

An example for such a technology are Java applets. These are special Java programs that run within Web browsers. The browsers are extended by a special plug-in, the Java runtime environment (JRE). The JRE provides a security context called *sandbox* that shields the applet from interference with other applications and guarantees that only certain operations on the client machine are allowed, e.g., read and write only to a `temp` folder instead of the entire system. Parts of the application can be loaded on demand from the server to keep initial start-up times and network traffic low. Besides these differences, Java applets are ordinary Java applications (section 5.2.3). Java classes can be written so that they act as applet or stand-alone application depending on one parameter. Advantages are minimal overhead for developing a Web client in addition to the stand-alone client and deployment across platforms, since JREs are available for nearly any operating system and browser. However, each applet runs in its own JVM (Java Virtual Machine) and each JVM requires a substantial amount of memory (>10 MB) and even on fast clients needs several seconds to load. Some companies also forbid usage of applets due to security considerations, though Java applets are commonly considered secure, especially if signed applets are used.

ActiveX

Another way of enriching Web-based applications is called ActiveX, a technology developed by Microsoft. ActiveX controls run natively in Internet Explorer on Windows and Macintosh platforms, but controls that are developed for Windows which is the vast majority, do not run on Macintosh and vice versa. Capabilities of ActiveX controls are similar to Java applets and partially integrated with Windows, e.g., drag&drop from Windows applications to ActiveX controls. Differences are that ActiveX controls do not have a sandbox-like security mechanism. It can be defined

that ActiveX controls have to be digitally signed and users get a warning before an ActiveX control is loaded. However, users can only choose whether to run the control or not and no security context can be specified.

Flash and Flex

Macromedia's Flash technology was originally designed as player for multimedia shows and animations available stand-alone or as plug-in to a Web browser. In recent versions, Macromedia has extended the capabilities, so that client application development has become much easier. Flash now offers a set of GUI elements, data binding to server-side data sources over XML and Flash Remoting to connect to an application server (J2EE, .NET or SOAP-based servers).

Flex is a server-side technology for application development that uses the Flash player as client. Flex applications use MXML, a Macromedia-specific XML dialect to describe GUIs, and ActionScript for client-side scripting and invoking of server-side methods. The MXML and ActionScript files are compiled into an SWF file (ShockWave Flash) and then transferred to the client that runs it using the Flash player. Further data is exchanged in the action message format, via Java RMI or Web services. advantages of Flash and Flex are a compact, fast-loading client-side plug-in that is available for many platforms and advanced multimedia capabilities that enable the creation of sophisticated user interfaces requiring small bandwidth.

Remote GUI

A recent development that is not necessarily browser-based and shows that existing alternatives for client technologies are still not satisfying can be summarized under the term *remote GUI*. The idea is to use a platform-independent from the operating system, e.g., Java with its GUI APIs (like Swing or SWT), and implement a transparent network connection that transfers application events to the server and sends server responses back to the client. Advantages of this approach are low network traffic as the client interface is completely handled on the client-side, whereas the client is relatively independent of the application version of the server, because the entire domain logic runs on the server. As most existing remote GUIs are Java-based, disadvantages are similar to applets, namely the substantial time required to load the JVM. This is a recent branch of GUI developments, so there is no standard yet, but quite a few startup companies offer first products. Examples are Altio Live, Asperon AppProjector, Canoo UltraLightClient, Droplets and Remote SWT.

5.2.3 Rich Clients

In some situations, the best choice might be to build a stand-alone client that has no limitations regarding access of system resources. As there are abundant technologies available for implementing applications in general and GUIs in particular, we can only briefly discuss some examples for more or less platform-independent frameworks offered by different vendors. A wide-spread programming language for platform-independent

development of client applications is Java, an interpreted language that needs a runtime environment to execute an application.

Java, Java WebStart

Java offers a lot of rich features and is a proven platform, but also has some drawbacks as mentioned before. These are slow loading of the runtime environment (JRE) and consumption of substantial amounts of memory for small applications (>10 MB). A copy of the JRE is loaded for every single Java application which can soon lead to memory problems. In recent JREs (version 1.5, also known as 5.0) this issue is addressed by letting applications jointly use JRE class files in memory. Another drawback is that central deployment of Java applications, i.e. installation of applications over network connections on demand, is not possible. Only recently, Java WebStart enables administrators to make the latest client version available on the server. Users that have not yet installed this client have to download the classes. Once the classes are on the client, the application can be used as often as wanted without any further downloads necessary as long as the classes on the server are not updated. Thus, the application is started from within the Web browser, but then runs without the Web browser as stand-alone application.

AWT, Swing

Sun is the main developer and founder of Java and supplies two class libraries for GUIs: the Abstract Windowing Toolkit (AWT) and Swing. AWT is a limited set of light-weight GUI components and considered relatively fast with respect to response time. Swing on the other hand provides a rich set of useful and optically appealing GUI components, but is considered rather slow as it does not use native API functions of the underlying OS. Instead, it renders GUI components itself using basic drawing functions. Therefore, GUI elements do not look identical to those used by native OS-specific applications which some users perceive as a disadvantage.

SWT, JFace

IBM offers the Standard Widget Toolkit (SWT), a set of GUI components that can be described as Java wrappers around the native OS GUI components. Implementations exist for Windows, Linux GTK and a number of Unix derivatives. JFace is an extension to SWT that provides additional high-level GUI components like Wizards. Since it is based on native OS functions, SWT is considered to be comparably fast and the look and feel of SWT applications is closer to native OS-specific applications.

Eclipse rich client platform

IBM additionally provides an open source platform that has been derived from Eclipse, a Java integrated development environment (IDE) with abundant features and a large number of users due to its open source license model. With the latest version 3.0, Eclipse has been split into an application independent platform called Rich Client Platform (RCP, also called workbench) and IDE-specific functionality on top of it. The workbench uses SWT together with JFace and can be used by other application developers to build their own applications reusing functionality of Eclipse like the plug-in system or views. Eclipse has received tremendous atten-

tion, especially since IBM released Lotus Workplace 2.0, a comprehensive office suite built on top of the Eclipse workbench.

.NET

Microsoft's .NET platform is commonly seen as a direct competitor to Java, though it was not designed as a cross-platform development environment. Like Java, it provides a runtime environment (section 2.4.2, 139ff). In contrast to Java, it is dependent on the programming language used. This attracts many programmers that are not familiar with Java, but command C++ or Visual Basic. The .NET GUI components use native Windows GUI components and are therefore responding quickly.

Mono

Finally, an open source project called Mono provides a runtime environment for .NET applications to Linux, Mac OS X and other Unix systems. In version 1.0, Mono only offers class libraries for a certain subset of the large .NET API (e.g., classes specific to Web services are missing). Therefore, it is currently not possible to use the full functionality of .NET on non-Windows operating systems.

5.2.4 Desktop Integration

In this section, we will discuss ways to integrate services of the EKI into employees' desktop environments. We first have to explore which types of applications are used by typical EKI users and how they relate to knowledge work in order to understand the requirements for the integration of enterprise applications with desktop applications.

Personal information management. The electronic desktop is the main interface office workers use for communication with their computer or, more precisely, the computer's operating system. It is a metaphor that was created to help people to carry knowledge of how to do things in the physical world over to the electronic computer world. A first commercial product with a desktop, windows, icons, menus, and a mouse as pointing device was developed by Apple in the early 80s. Though PCs have seen dramatic changes since then as the table illustrates, the metaphor for user interaction is still largely the same (Table 5-2, Maier/Sametinger 2004).

Table 5-2. Early PCs and PCs today compared

	early PCs (1985)	**PCs today (2004)**
metaphor	desktop	
object	document (file)	
transformation	application (program)	
interaction	WIMP (windows, icons, menus, pointing device)	
software	limited number, primarily office applications	huge number, including multimedia and Web applications

Table 5-2. Early PCs and PCs today compared

	early PCs (1985)	**PCs today (2004)**
resources	limited, e.g. 128 kB RAM, 400 kB hard disk, floppy disk drive (170 kB)	extremely powerful, e.g., 512 MB RAM, 160GB disk, CD/DVD-RW (several GB)
input / output	keyboard, mouse, small, monochrome monitor	keyboard, mouse, large color monitor, color printer, microphone, photo and video camera
documents	a few dozen or hundreds	several thousands
connectivity	isolated	permanent, fast network connection (100Mbit/s)
PC - user ratio	one desktop PC per user	one user often has a desktop PC, a laptop, a PDA and a smartphone

Problems with the desktop metaphor

The number of documents and resources have substantially increased and applications do not run isolated any more. PCs are always online and connected to servers and other PCs. That leads to the fact that the desktop metaphor is not as useful any more due to several problems:

- *isolation*: information is bound to specialized applications so that users have to remember which application they need before they can start the search for the information, e.g., "Have I read the piece of information in an email (email client), was it in an attachment that is in the file system now (file manager) or was it a link to a Web site and I have bookmarked it (Web browser)?". Information is rarely reused across applications.
- *loss of context*: meta-data that often already exists in electronic form at some point of time, e.g. when an email with attachments is received, is lost when files are stored in the file system, e.g., who sent the file, the topic specified in the email subject and additional information in the email body.
- *versioning*: often, a document exists in multiple versions in the file system that either represent evolving states of the document or are representations of the same information in different file formats, e.g. doc, pdf, htm, rtf.

Desktop alternatives

That leads to the fact that often the file manager, the email client, the Web browser or other specialized programs are more often used for access to information as the desktop. Thus, a new metaphor has to be found. Several proposals have been made. Examples are (see Maier/Sametinger 2003 for an in depth comparison):

- *Lifestreams*: a stream of documents is used with temporal sequence instead of folders,
- *Time-Machine Computing*: an archive of the information history permits "time travels" through documents on the desktop,

- *Presto*: uses a uniform document model with no hierarchical arrangement,
- *TheBrain*: uses a semantic net to order and access resources on personal computers, Intranets and the Internet,
- *Creo SixDegrees*: a search engine for the desktop that uses search instead of browsing,
- *Data Mountain*: a collection of Web resources in a 3D space instead of bookmarks,
- *TopicShop*: collected Web sites are assessed and organized. Content of high quality is identified,
- *Infotop*: an OLAP-like, multi-dimensional view is used to access files, emails and Web sites. Dimensions are time, topic, location, person, process and document type. One- or two-dimensional views can be used for any combination of dimensions.

Integrated file manager and Web browser

Today, the desktop is often tightly coupled with a file manager and Web browser (e.g., Windows XP integrates the desktop, Windows Explorer and Internet Explorer, the KDE Konqueror integrates file manager and browser). Examples for other file managers are Nautilus and Norton Commander clones like TotalCommander. Most widely-used browsers are Apple Safari, MS Internet Explorer, Mozilla (and Firefox) and Opera.

Office software

Other office software used quite often in daily work are (1) text processors, e.g., Corel WordPerfect, MS Word and OpenOffice Writer, (2) spreadsheet software, e.g., Corel QuattroPro, MS Excel and OpenOffice Calc and (3) presentation software, e.g., Corel Presentations, MS Powerpoint and OpenOffice Impress. These three types of applications are usually bundled in office suites, e.g., Corel WordPerfect Office, MS Office or OpenOffice and also provide integration between the applications to some degree. Desktop DBMS, e.g., FileMaker Pro, Lotus Approach or MS Access, are part of extended office suites.

Groupware client

Another type of application central to office work today are Groupware clients, e.g., Lotus Workplace Messaging, MS Outlook and Novell Evolution (section 4.3.2, 276ff). Groupware contents can be stored on users' PCs or centrally on a Groupware server. Advantages of storing contents centrally are the possibility to access them from any PC in the organization or even through the Web from any PC with Internet connection using the Groupware Web interface.

Publishing software

These often-used software programs are supplemented by occasionally used specialized software. The creation of text documents with sophisticated layout, e.g., marketing material, or hundreds of pages, e.g., text books, requires specialized software for publishing. *Desktop Publishing (DTP) software*, e.g., Adobe FrameMaker and PageMaker, Microsoft Publisher or Quark XPress, are better suited for these purposes. For Web publishing, *HTML editors* are mainly used when export from office applications is not sufficient and there is no content management system in the

organization. There are WYSIWYG (what you see is what you get) editors, e.g., Macromedia Dreamweaver or MS Frontpage, for inexperienced or layout-oriented users and code editors, e.g., Macromedia HomeSite and Sausage Tools' HotDog Professional, that target professional or programming-oriented users.

Other text-oriented software

For small text files, e.g., configuration or log-files, it is useful to have a compact *text editor* instead of having to use a large memory-consuming text processor, e.g., jEdit, UltraEdit or VIM. *Spell checkers* are useful in order to aid users in creating texts and can be combined with *grammar checkers* and *thesauri*. Another often needed service is language translation. Current applications help to get a rough notion of the meaning of text, suggest translations and *dictionaries* provide some additional help. There are also a number of programs called *readers* or *viewer* for quick access to text documents without requiring the entire application which is only needed to edit and create the documents, e.g., Adobe Reader, GhostView or MS Word Viewer.

Multimedia applications

Handling of multimedia information becomes more important to office workers. We briefly summarize the most important types of multimedia applications. For images, one has to distinguish between vector and bitmap images. In the context of office work, *vector graphics* can be used to visualize technical or modeling information, e.g., workflows, organigrams or network architectures, or to illustrate text, e.g., culprit. For the former usage, applications like ConceptDraw, MS Visio or SmartDraw are well suited whereas the latter requires full-scale illustration software like Adobe Illustrator, CorelDraw and Macromedia FreeHand. Sometimes, vector-oriented animations are useful, e.g., supported by Macromedia Flash. *Bitmap graphics* like photos and icons can be edited with products like Adobe Photoshop, GIMP or Jasc Paint Shop Pro. Applications for easy management of large collections of image files, e.g., ACDSee, Adobe Photoshop Album or Thumbs Plus and quick viewing of images, e.g., IrfanView, JPegger or XnView, are often useful as well. *Video clips* can be watched with players like the Real Player, Windows Media Player or Xine.

Sound files can be played with e.g., Apple iTunes, WinAmp or XMMS. Those programs often also offer management functionality for audio files. *Sound recording and editing* software includes CoolEdit, SoundForge and WaveLab. A different type of audio information is encoded in *MIDI* (music instrument digital interface) files. Those can be seen as coded information whereas wave file formats represent audio data as non-coded information (section 4.2.1, 247ff).

Software tools

Finally, there are some useful tools that most office workers use sometimes including *compression tools* that compress files lossless into archives, e.g., GZip, PowerArchiver, WinACE, WinRAR or WinZip. *CD and DVD burning* programs are also more and more important, e.g., Ahead Nero, k3b and Roxio WinOnCD.

Table 5-3 gives an overview of important file formats for the applications discussed above.

Table 5-3. Overview of file formats

subgroup	format
text formats	TXT, RTF, DOC, SDW, SXW, CSV, HTML, XML
page description formats	PS, EPS, PDF, FM, PM6, QXD
other office formats	XLS, SDC, SXC, PPT, SDD, SXI, MDB, FP, APR
bitmap file formats	TIFF, BMP, GIF, JPEG, PNG
vector file formats	EPS, WMF, EMF, CDR, SVG
video file formats	AVI, WMV, RM, MOV, MPEG, DivX, XVid
audio file formats	WAV, MP3, RAM, WMA, AAC, OGG, AU, MIDI
compressed file formats	ZIP, RAR, TAR.GZ, ACE

Amounts of data

The amount of data produced in organizations worldwide is astonishing. Estimations for the year 1999 are summarized in Table 5-4. Averages for newspapers, journals, magazines and newsletters are calculated per issue, assuming 300 issues per year for newspapers, 6 issues per year for journals and magazines and 52 issues per year for newsletters. However, there is even more informal information exchanged. A study for 2002 shows that the production rate is still increasing and estimates e.g., 274 TB of data generated by instant messaging and 400,000 TB of data exchanged as emails (Lyman et al. 2003).

Table 5-4. Produced data volumes per document type in 1999 (based on Asprey/Middleton 2003, 13)

medium	titles in thousands	average size	total size
books	968.7	8.3 MB	8.0 TB
newspapers	22.6	3.7 MB	25.0 TB
journals	40.0	8.3 MB	2.0 TB
magazines	80.0	20.8 MB	10.0 TB
newsletters	40.0	0.1 MB	0.2 TB
office documents	7,500,000.0	0.03 MB	195.0 TB
music CDs	90.0	66.7 MB	6.0 TB
DVD videos	5.0	4400.0 MB	22.0 TB
total	**7,501,246.3**		**268.2 TB**

Synchronization

Modern office systems are no longer stand-alone applications. Instead, they are integrated systems that often handle data from central data stores on Intranet or Internet servers in addition to local data. Therefore, intelligent synchronization or replication mechanisms are required that control data redundancy on servers, PCs, notebooks and other mobile devices (section 5.3.3). Synchronization can also be used to share documents in a team or workgroup, e.g., with the help of Groove Virtual Office or Novell iFolder.

Plug-in technologies. A proven way for integrating additional services into desktop applications is using a plug-in system. The implementation requires interfaces defined by desktop applications, e.g., Web brokers, and software modules that implement those interfaces. Thus, the application can use these modules. For example in the Adobe Photoshop plug-in system, several hundred plug-ins are available for Photoshop that add new visual filters and effects. Some competing photo editing applications have also incorporated the system, so that the same plug-ins can be used.

File manager plug-ins

File managers are often used to access files for viewing or editing. A trend can be recognized towards launching applications implicitly by opening a file with a file manager or the desktop, also called document-centric paradigm, instead of explicitly choosing an application and opening the document from there, also called application-centric paradigm. Users mainly need discovery and publication services in connection with the file manager, i.e. integrated access to files on their local computer, remote file servers and especially remote DMS and CMS including navigation and search.

Example: Windows Explorer integration

The Windows Explorer provides a large set of integration points. There are three possible integration points for plug-ins. New commands can appear in the (1) *context menus* of folders or certain file types, (2) as new menus in the *menu bar* or (3) as icon in the *toolbar*. Finally, plug-ins can (4) display an icon for the module as part of the directory tree as *network drive*.

Context menu

Examples for context menu integration are compression with almost all Windows compression programs (WinZip, WinRAR, WinACE), Adobe Acrobat document conversion and UltraEdit editing features. A static way to achieve similar results that also works for Linux KDE is the assignment of functions to file types via the options dialog.

Toolbar and menu

The menus and toolbar icons are handled equally in Windows Explorer. Modules that use this integration point are called file manager extensions in Microsoft terminology. Examples for file manager extensions are Groove folder synchronization and Opentext Livelink Explorer.

Network drive

Integration as network drive is the most complex way of integrating to Windows explorer since users expect to get the same functionality as with local folders, e.g., context menus, drag and drop operation, searching. It is often used for DMS, e.g., Opentext Livelink or Windream, to access the

contents of the DMS just like files on a shared network drive. Some manufacturers also use this mechanism to show the controls for hardware devices, e.g., Logitech Quickcam.

WebDAV

A much simpler alternative to network drive plug-ins is the implementation of WebDAV on the server application (section 2.3.2, 119ff). Windows Explorer can connect to applications supporting the WebDAV standard by mounting a network drive. Many other file managers have the same capability. Thus, full read and write access is possible, but no application-specific context menus as it is with plug-ins.

Web browser integration, ActiveX

Web browsers need to display various file formats, especially multimedia and XML files that are required to facilitate GUIs for Web applications. Manufacturers cannot provide every desirable functionality on their own in a timely manner. Therefore, Web browsers usually offer plug-in systems that enable third-party manufacturers to build modules that fill the gaps. Examples for Internet Explorer (IE) plug-ins are Macromedia Flash, Java JRE, and viewer for scalable vector graphics, XForms and other XML standards. Another essential plug-in for Web browsers is the Adobe Reader plug-in to enable PDF viewing directly inside the browser. Microsoft calls Internet Explorer plug-ins ActiveX controls (section 5.2.2). Another problem of the Internet Explorer plug-ins is that they are using CLSIDs (class identifier) instead of mime-types to determine the plug-in to use. CLSIDs are the Windows standard to uniquely identify an application or library (DLL). That means that creators of Web sites have to prescribe which plug-ins users have to use which hinders open standards implemented by various vendors.

Netscape plug-in API

Despite that, the big market share of the Internet Explorer and lack of a single plug-in system for other browsers which are cross-platform and therefore cannot use ActiveX lead to the fact that many manufacturers of plug-ins provided only IE versions of their plug-ins or Netscape versions are not working as well. Therefore, other Web browser manufacturers have agreed to use a single plug-in system, the Netscape plug-in API (NPAPI). In addition, they created an ActiveX control that is able to use NPAPI plug-ins within the Internet Explorer. That is a strong argument for plug-in vendors, so that browser plug-ins that work in any browser can be expected.

Nested elements

One challenge for browser plug-ins for XML dialects remains. XML allows users to mix elements of various XML dialects within one document. It is for example possible to create an XHTML page that includes SVG-elements to graphically enhance headlines and XForms elements to let users enter messages to be posted to a newsgroup. XHTML is directly interpreted by the browser and different plug-ins display the SVG and XForms contents. However, if SVG and XForms elements are nested, e.g., when using SVG for form labels or when a form should be embedded in a SVG image, the page is not displayed correctly because one plug-in does not load another one.

Office software integration

In the context of office products (text processor, spreadsheet and presentation software), users expects seamless access to documents stored locally, on file servers and in DMS. Additionally, users need integrated support for document publishing including versioning, format conversion and storage. MS Office is the most wide-spread office package.

SmartTags

Since version 10 (Office XP), it offers a technology called SmartTag that was further extended in version 11 (Office 2003). SmartTags build on Word AutoCorrect functionality and offer a plug-in interface that has to be implemented to create own SmartTags. Every word typed in the text processor is parsed by the AutoCorrect function and gets verified for correct spelling. The SmartTag interface uses the event that a new word was verified to hook one or more modules conforming to the interface. It passes the word to modules which can decide whether it is a keyword or not. If it is found to be a keyword, it appears in the GUI as underlined with a dotted line. If the user moves the mouse pointer over this word, a menu pops up that offers functions in conjunction with the word. The SmartTags shipping with MS Office 2003 (English version only) try to identify persons, locations and addresses based on text mining functions, financial symbols, e.g., cusip numbers, based on lists in a special XML format and person names based on entries in Outlook address books. Functions associated with persons are for example *send mail*, *open contact* and *schedule a meeting*. There are also third-party products that integrate into MS Office using SmartTags. Ontoprise's OntoOffice links words in the text document to concepts in an ontology. Thus, associated documents can be searched or associated concepts from the ontology can be displayed.

Add-Ins

Office supports another plug-in system called Add-Ins. They consist of a document template that resides in the `Office/Addins` folder and is responsible for loading the module and creating the menus or toolbars and secondly either VBA code (Visual Basic for Applications) or a COM DLL (section 3.4, 203ff) that contains functions that are called by the menu entries or toolbar icons. Examples for applications that use this functionality are Adobe Acrobat to convert office documents into PDF, Groove Virtual Office to publish documents in Groove Workspaces or Opentext Livelink to enable users to load documents from and save to Livelink.

Printer driver

A printer driver for an operating system has to conform to a pre-defined interface and is therefore similar to plug-ins. Users can print documents using a special printer driver to send them to an application, e.g., a DMS, instead of a physical printer. The printer driver gets a copy of the document that it can send to the application over the network, which stores it at a given location. Only writing to an application is possible, reading from the application is not supported. An example for a DMS offering a printer interface is Lacerte Document Management System. Conversion of documents to PDF is possible using Adobe PDF printer.

Groupware client integration

Groupware clients often are used as central information access point for accessing emails, calendar items and contacts. One group of plug-ins for

groupware clients deals with supporting additional content types and information sources such as news in RSS (RDF Site Summary) format (e.g., NewsGator), additional messaging formats such as SMS (e.g., TeleMessage) or communication protocols for additional Groupware servers (e.g., SKYRiX ZideLook for access to Open Groupware from Outlook). A second group of plug-ins deals with a more comfortable or secure handling of the basic Groupware functionality by adding security functionality like encryption or digital signing for emails (e.g., PGP or FlexiSMIME) or automation of tasks like automatic compression of attached files (e.g., WinZip email Attachment Add-On for Outlook). Other plug-ins integrate Groupware clients with other enterprise systems like DMS (e.g., Livelink email integration client and Groove Virtual Office).

Application-spanning plug-ins

A recent development is adoption of a plug-in system across application domains as it currently happens with the plug-in system of the open source platform Eclipse which is reused in many other open source applications.

5.3 Mobile Access

5.3.1 Mobile Computing

The term *mobile computing* denotes all electronically supported activities of non-stationary users. We will concentrate on employees that use electronic devices to participate in business activities or processes.

Characteristics

There are some characteristics of mobile computing that differentiate it from stationary use of PCs. First of all, access can happen any time and any place whereas EKI traditionally are only accessed within business hours and from workplaces. An example is email access from many mobile devices. Furthermore, users can be located and information needs may vary strongly with the location. Presentation of contents on mobile devices is different due to limited capabilities of mobile devices, e.g., screen size, resolution, and due to connection speed, as in most cases quick text information is more efficient than graphically rich information that needs several minutes to be downloaded. Finally, context information about environments other than traditional workplaces have to be considered.

Costs and benefits

Utility of mobile computing for organizations can be broken down into the following factors. Employees can take part in the business processes and get information even if they are not at their workplaces or in company buildings. Without mobile access, workflows, e.g., waiting on approval by a mobile decision maker, often take weeks to be completed if the responsible person is out of office. Decision quality can be improved by providing required data at the right time and place. Data can be provided systemati-

cally depending on the current location of the user and vice versa, location data of the user can be used to improve business processes, e.g,. ad-hoc optimization of routes for field sales force. However, mobile computing can induce high costs. Mobile devices have to be bought and maintained, mobile network services are more expensive than stationary ones and transfer times are long due to small bandwidths.

Target groups

Employees who benefit most from the new possibilities of mobile business are those that have jobs requiring activities at various places within and outside company buildings, those that travel a lot, often need current information to make decisions and those that occupy important positions within business processes and therefore have to be permanently reachable. Examples are field sales force, consultants at customer sites, repair and maintenance teams, emergency doctors, as well as employees of procurement departments, coordination-intensive control stations, logistic units and other information-intensive positions.

Location methods, IP-based

The use of location data is one of the most important characteristics of mobile business. Services are called *location-based services* if they use location data to enhance service quality. A number of different techniques can be used to locate the user. The only way to locate users with stationary devices is based on IP addresses and only reveals country, city and organization of users. The latter only works for devices with permanent Internet connections and locating is not exact (about 10 kilometers).

Net-based locating

In mobile networks, the cellular structure of the network is used to locate the user more exactly. The *cell-of-origin method* uses the cell a user is logged in to determine location. Depending on the region (urban or rural), accuracy of locating the user is between 0.1 km and 35 km. The *time-of-arrival method* is far more precise. It uses at least three base stations that are able to receive signals from the end-user device. The device sends a signal with a timestamp and each station measures the time the signal needs to get to the station. Thus, distances between user and stations are calculated with up to 50 m precision using triangulation. In contrast to these two net-based methods, a group of different methods requires more "intelligence" for the mobile devices and is called terminal-based.

Terminal-based locating, GPS

The best known terminal-based method is GPS (Global Positioning System) which is controlled by the US military and uses a network of 24 satellites (plus four spare) to locate users. Again, at least three satellites must be available for triangulation. Despite the enormous distance (20,000 km), the method is very accurate. GPS allows locating up to a precision of 10 meters for public use and up to 3 meters for military use. The European Union currently tests its own satellite navigation system called Galileo with 30 satellites that ought to be publicly available in 2006. Disadvantage of satellite-based locating is that mobile devices need special, often expensive extensions to receive the signals. Additionally, it requires a lot of energy to send a signal that reaches the satellites which results in reduced operating time.

Location-based services

Five basic types of location-based services can be distinguished (Reichwaldt 2002, 402f):

- *Position acquisition* solely denotes locating the user ("Where am I?"). The result is a position in geographical coordinates (longitude and latitude).
- *Reverse geocoding* is then applied to transform coordinates into an address (city, street and number).
- *Geocoding* transforms addresses back into geographical coordinates and is the first step if users initiate any address-related searches.
- *Spatial search* can locate other objects or users and calculate distances (air-line) based on geographic coordinates whereas *routing* also considers physical attributes of the region like streets, rivers and mountains to calculate distances and recommend routes to get to the target.
- *Mapping* is a service that generates a map of the user's local environment.

Higher level services

On top of these basic services a number of higher-level services can be built. Examples are navigation systems, baggage locating, location-dependent notification services, e.g. advertising, traffic information, yellow pages and automobile roadway repair service. Table 5-5 gives an overview of these services and shows how precise locating must be in order to achieve broad user acceptance.

Table 5-5. Location based services and required locating accuracy

service	accuracy entry level	accuracy broad acceptance	utility	locating occurrence
breakdown/emergency help	500 m	125 m	get help fast	user initiated
yellow pages	cell	250 m	find points of interest nearby	user initiated
traffic information	cell	cell	avoid traffic jams	user initiated
luggage locating	cell	cell	locate, route	user initiated
navigation system	125 m	30 m	navigate	every 5 sec.
notification	cell	125 m	get current local info	every 5 min.
billing	cell	250 m	differentiate prices	user initiated

A few case examples will further clarify how mobile computing contributes to organizational performance in various industries.

Insurance company

A sales agent of an insurance company visits a customer who needs to insure his new car. The agent enters the car data directly into the insurance application on her notebook and looks up the rate online via an Internet connection established through her mobile phone. After the data is com-

plete, she directly submits it to the company server that starts a workflow. Employees working in the nearest subsidiary of the insurance company immediately get the order, print the policy and send all necessary documents to the customer.

Service or repair team

After the repair process at the customer site is finished, the repair team enters the description of the case, measures taken and time needed. They submit the data via a GPRS connection to the coordination center and get new orders for the next assignment. Their current location, customers' locations, time and current traffic situation are taken into account.

Logistic service company

Every packet has a unique identifier (transponder, RFID) and all transport vehicles have mobile devices that make it possible to locate them. Thus, new services like packet routing, electronic letter of acknowledgement and just-in-time planning of parcel pick-up can be realized.

Newspaper

A news reporter attends a press conference, takes photos and types an article directly on site. She inspects the pictures at her laptop and chooses the best fitting ones. She transfers pictures and text to the editorial office that compiles all articles into the online issue of the newspaper, so that timeliness of news is increased significantly.

Emergency team

In case of an accident, an emergency physician is alarmed by the dispatch center via his smart phone and the address together with first information about number of persons and injuries are transferred as text messages to the smart phone, so that there can be no misunderstandings. The GPS-based navigation system helps him to quickly get to the site of the accident. After he has inspected the patients and has made a first diagnosis, he can send a request for more emergency teams, enter data into the smart phone describing the condition of the patient(s) and request or suggest a hospital. The hospital therefore has more time to prepare the intensive-care unit or call doctors that are on stand-by at home.

5.3.2 Mobile Devices

The industry that develops and manufactures mobile devices is characterized by convergence of telephone and computer industry. Mobile telephones get more and more sophisticated and features are integrated that have originally been developed for PCs. Mobile devices are designed for different situations and usage scenarios which leads to varying product features.

Characteristics. Main features can be categorized into communication features, technical device characteristics, programming capabilities and available applications (Table 5-6).

Table 5-6. Characteristics of mobile devices

category	feature	feature values
communication	voice telephony	supported / not supported
	messaging	SMS / EMS / MMS / email
	mobile Internet	WAP+WML / TCP/IP+ HTML
technical characteristics	memory	RAM: 1-2048 MB flash memory cards: supported / not supported hard disk: none / 20-200 GB
	text input	12 digit keypad / QWERTY keyboard / touch screen with pen and handwrite recognition / voice
	screen size, resolution and colors	from 1 inch,100x60 pixels, monochrome to 17 inches, 1920x1200 pixels, 24 bit colors
	connectivity	PAN: USB, IEEE 1394, IrDA, Bluetooth LAN: Ethernet, WLAN cellular: GSM, GPRS, EDGE, HSCSD, UMTS
	operating time	from 60 minutes to 10 hours in use and from 3 to 15 days standby
programming capabilities	supported languages and libraries	Java MIDP / Java Personal Profile, .NET compact / Java Standard Edition, .NET, C++, …
available applications	PIM	address book, calendar, task list, notes
	office	text processor, spreadsheet, presentation software
	multimedia	image viewer / MP3 player / video player / image, sound and video recording and editing

Communication

Voice telephony was the first communication technique and provides synchronous communication, so that people could still share knowledge or make decisions while they were out of office. Additionally, a need for asynchronous messaging arose which was satisfied in a lean way with *SMS* (short message system) and richer with *emails* where message length is not limited. Information needs can be further satisfied with *mobile Internet access* as information from the Intranet or the Internet can be retrieved any time. It is important to note that access to the Intranet does not automatically mean access to all application services available there, as many of them have resource demands that only few device classes can fulfill.

Technical characteristics

Technical device characteristics often determine whether using an application on a device is feasible or not. Enough *available memory* is usually a mandatory requirement whereas a limited keyboard with small keys makes the use of applications that require much text input only more difficult. *Screen size* and *resolution* determine how much and how information

can be displayed. For some applications, this means that layout of the front end has to be adapted when porting it to mobile devices, other applications are not usable at all if they require a minimum resolution and size that is not provided. An e-book reader for example is usable on small screens if the text is formatted adequately, e.g., by using appropriate fonts, line breaks and image formats. If an e-book is e.g., in DIN A4 format, users of small devices inconveniently have to scroll left to right and top-down to read the text. *Connectivity* options determine together with the availability of according base stations if and with which bandwidth users can work online. Without online access, they can use only the limited set of information on the device which has to be synchronized with a base station before and after offline work. *Operating time* is described in terms of standby and usage time. Short operating times limit usability of devices.

Programming

Programming capabilities of a device are a significant factor for availability of applications. The more different devices a software developer can cover with one programming platform, the higher is the probability that this platform will be supported. Additionally, the more similar mobile platforms are to desktop platforms, the easier it is to port desktop applications to them. Therefore, Java MIDP (mobile information device profile, see also section 5.3.3) is currently the most supported platform as it runs on nearly all devices from small phones to larger PDAs (Personal Digital Assistants) and notebooks. Its object model is similar to the Java desktop version (Java Standard Edition).

Applications

Available applications also depend on device or operating system manufacturer, as the basic set of applications such as viewers, or PIM applications are usually packaged with the devices. Software providing office functionality, e.g., text processing, is mainly available for medium-sized mobile devices and bigger ones, e.g., PDAs and notebooks, though PDAs are often only suitable to briefly look-up information and correct or add a few words in existing documents. Multimedia applications are one of the key differentiators for mobile devices these days, although they currently are mainly oriented towards the consumer market and rarely needed for business activities.

Mobile phone

Device classes. Mobile phones are primarily designed for voice telephony and have SMS messaging capabilities. Besides that, some may also have WAP capabilities and a WML browser that work over a GSM or sometimes GPRS connection. These devices are typically small and have a monochrome screen with small resolution (up to 128x128 pixels) and small memory (e.g., 1 MB). Input possibilities are mainly a keyboard with 12 numeric and a few function keys. These keys are designed to type in telephone numbers. If any, these devices offer infrared connections.

Feature phone

Feature phones are extended mobile phones with simple multimedia functionality and basic PIM support. They often support MMS and have a small color display (e.g., 120x172 pixels with 12 to 16 bit colors). Some of

them also have an integrated digital camera with small resolution (CIF, 320x240 pixels). WAP is a common feature and all products support at least one high-speed data service. Another feature common in this category is the possibility to download ring tones and background images that are stored in memory (typically 8 to 16 MB). Simple programs in MIDP format can be downloaded and used. Connection alternatives are at least USB and IrDA, some also support a couple of Bluetooth profiles.

Smartphone

Smartphones are feature phones with full PIM and basic office functionality (text processor and spreadsheets), a larger display (176x220 or even 208x320 pixels with 16 to 18 bit colors) and large memory (at least 32 MB) that usually can be extended via flash memory cards. Some have a digital camera with at least VGA resolution (640x480 up to 1152x864 pixels). Advanced multimedia functionality like MP3 playback and video clip playback are also common. Smartphones regularly support several Bluetooth profiles. Internet connections are not only possible via WAP, but also via TCP/IP so that standard HTML pages can be viewed. All products in this category support GPRS and EDGE or HSCSD and there are products with UMTS support. Recent smartphones support at least the Java MIDP 2.0 specification and some of them support also advanced Java profiles like the PDA profile or even the Personal Profile or have .NET compact support. Smart phones are often bigger and heavier compared to feature phones (e.g. 120g, 109 x 56 x 19 mm). Input possibilities are often extended with a small keyboard or a touch screen with handwrite recognition.

Personal digital assistant

The main distinction between smartphones and Personal Digital Assistants (PDA) is that PDAs have larger screens (320x240 or 320x320, recently even 320x480 and 640x480) which leads to larger device size and weight (e.g. 150 g, 110 x 76 x 16 mm) and usually provide no support for voice telephony. Instead, they all provide powerful programming possibilities with Java Personal Profile or .NET compact and even more with a native compiler (mostly C++). Full PIM and nearly full office functionality is standard, as well as email clients and HTML browsers. PDAs have large memories (64 to 256 MB) and support mass storage extension with flash memory cards. Standard input is via a touch screen and handwrite recognition or a virtual keyboard that is displayed on the touch screen. Some also provide a QWERTY keyboard. Connection to PCs is realized via USB. Many devices provide wireless connections via Bluetooth and IrDA whereas only high-end products additionally support WLAN. Some of them can be extended with GPS modules, so that they can serve as navigation system.

Phone PDA

There are also some PDAs that integrate voice telephony features and close the gap to smartphones. They differ physically from usual PDA designs in having an external antenna.

Blackberry

A feature of smartphones and PDAs that recently came up is called Blackberry. It is an email push technology that sends emails directly to the

device without requiring user action. Devices supporting Blackberry offer QWERTY keyboards to help users to create emails on the phone PDA or smart phone.

Clamshell PDA

PDAs that get closer to notebooks are called clamshell PDAs, because they are foldable. They have a larger keyboard and screen size (e.g., 3.7 inches) than other PDAs which leads to increased size and weight (e.g., 220g, 128x83x24 mm).

Subnotebook

Subnotebooks are lightweight mobile PCs with only a few limitations regarding display size, memory and computing power. They are substantially bigger than clamshell PDAs (e.g., 1400g, 270x188x36mm) and have 9-12 inch displays with SVGA (800x600 pixels), XGA (1024x768 pixels) or higher resolutions. They do not support voice telephony. The biggest limitations compared to larger notebooks are lack of periphery, e.g., internal optical drive, slower CPUs and keyboards that are not suitable for typing long texts. Connectivity to other systems is possible via IrDA, USB and Ethernet. Some subnotebooks also provide an IEEE 1394 interface, Bluetooth or WLAN and have at least one PCMCIA (Personal Computer Memory Card International Association, a standard for expansion cards) slot, so that missing interfaces can be added as card solution. Subnotebooks, as well as all devices in the following classes are x86-compatible and can therefore run the same applications as stationary PCs.

Tablet PC

Tablet PCs are notebooks that can be controlled via pen-based input and handwrite recognition. They are similar to PDAs in the way that they have a touch screen, but in contrast to them are able to run standard PC software. They are nearly as small as subnotebooks (e.g., 1500 g, 324 x 220 x 22 mm) with display sizes ranging from 10 to 14 inches. CPUs, hard disks and connectivity capabilities are similar to those of subnotebooks.

Convertible PC

Convertible PCs (short: convertibles) are only slightly bigger than tablet PCs and provide both, a keyboard and a touch screen, so that one of these input devices can be used depending on situation and application. There are currently more convertible PCs than tablet PCs on the market and tablet PCs might be completely replaced by convertible PCs.

Notebook

Notebooks are bigger and heavier than subnotebooks (e.g., 2700 g, 340 x 275 x 35 mm) with displays usually ranging from 14.1 to 15.4 inches, up to 17 or 18 inch displays in the 16:9 aspect ratio wide screen format. XGA resolution (1024x768 pixels) is standard whereas larger ones have resolutions of 1400x1050 or even 1920x1200. With the additional space of larger notebooks not only larger displays, but also faster CPUs, larger hard disks and larger keyboards are integrated, so that these notebooks are suitable for typing longer text passages and CPU-demanding applications. Notebooks usually have an integrated optical (DVD) drive with burning capabilities. Connectivity options include IrDA, Bluetooth, WLAN, USB, Ethernet and IEEE 1394. GSM or other cellular phone standards are not supported by default, but can be added as PCMCIA cards. The same is true for GPS receivers.

Desktop replacement notebook

Desktop replacement notebooks are compact alternatives for desktop PCs and are not intended for mobile work in a narrow sense. These notebooks usually have more computing power and memory, but are much heavier and have short battery runtimes.

Table 5-7 gives an overview of the device classes and presents typical product examples for each class. Due to the short product life-cycles in the computer and mobile phone industry, many of them may soon be replaced with a successor, but nevertheless product pictures will be present in the Internet for a longer period, so one can get an impression of the devices.

Table 5-7. Typical example products by device class

device class	**example products**
mobile phone	Siemens CX65, Nokia 6610, Motorola C650, Sony Ericsson T200 and Samsung SGH-Q100
feature phone	Siemens C55, Nokia 6310, Motorola C350, Sony Ericsson T630 and Samsung SGH-S100.
smart phone	Siemens SX1, Nokia 9210i Communicator, Sony Ericsson P910i and Samsung SGH-D710
PDA	PalmOne Tungsten T3, HP iPAQ h2210, Sony Clié PEG-TH55 and Dell Axim X5
PDA with VGA resolution	Toshiba e800 and Asus MyPal A730
PDA with full keyboard	PalmOne Tungsten C and Sony Clié PEG-TG50
phone PDA	PalmOne Treo 600, HP iPAQ h6340 and T-Mobile MDA III
phone / PDA with Blackberry	Siemens SK65, RIM BlackBerry 7730
clamshell PDA	Sony Clié PEG-UX50 and Sharp Zaurus C760
subnotebook	without optical drive: JVC MP-XP731, Panasonic CF-T2 and Sony Vaio VGN-X505VP with optical drive: JVC MP-XV841, Panasonic CF-W2, Samsung Q25 TXC 1300 and Sony TR1MP
tablet PC	Fujitsu-Siemens Stylistic ST5010 and HP Compaq Tablet-PC TC1100
convertible notebook	Acer TravelMate C302XMi, Fujitsu Siemens Lifebook T3010 and Toshiba Portégé M200
notebook	Dell Inspiron 510m, IBM Thinkpad R51 and Toshiba Tecra A2
desktop replacement notebook	Acer Aspire Aspire 1714SMi, Dell Inspiron 9100 and Toshiba Satellite P20

There are a lot of other mobile devices available on the market and many of them integrate functions from other device classes. We are not

discussing different types of MP3- and video-players, mobile gaming devices, photo and video cameras or mobile storage devices. These cannot connect wireless to companies' Intranets and help users to access emails or other knowledge sources.

5.3.3 Mobile Applications

Based on the description of mobile devices, a number of characteristics limit development of mobile applications (Reichwaldt 2002, 44):

- small bandwidth,
- less processing power and small memory,
- limited output (screen size, resolution, no printer) and input capabilities (small keys, often no full keyboard), and
- high degree of variability in device capabilities and programming environments.

Mobile operating system

The further up programming moves with respect to software layers from device-dependent machine code towards high-level programming interfaces, the higher is the number of users with different devices a programmer can reach with one programming platform. Therefore, high-level platforms are often favoured, although they may be slower and do not use all device capabilities. A first step towards standardization is the use of a common operating system. There are currently three important operating systems for mobile devices that are not x86 compatible, i.e. are not able to run Windows, Linux and other common PC operating systems.

Palm OS, Symbian OS, Windows Mobile

Palm OS currently is the dominant system in the PDA market whereas Symbian OS is leading the field of smart phone operating systems in Europe. Microsoft is gaining market share with Windows Mobile (formerly Windows CE) in versions for PocketPCs (PDAs) and for Smartphones. All three operating systems are similar in their capabilities and can be programmed in the programming languages C or C++ using OS-specific APIs. Main differences are screen resolutions and number of colors supported.

- Palm: 160x160 grayscale, 320x320 16 bit colors, 320x480 16 bit colors
- Symbian: commonly 176x220, some with up to 640x480 16 bit colors
- Windows: 320x240 16 bit colors, 640x480 16 bit colors

Developers usually use device simulators such as the Palm Simulator to test and debug their applications before they are transferred to and tested on real devices. Unfortunately, simulators behave not 100% identical to real devices.

Mobile frameworks, J2ME

A further abstraction is reached by using higher-level programming frameworks that are (ideally) independent of the OS. The most widespread framework on PDAs and smart phones is Java. The version used is called *Java 2 Micro Edition* (J2ME). The J2ME framework is broken

down into a large number of specifications (Java Specification Requests, JSR) that are partly inter-dependent. The first distinction has to be made between the *connected device configuration* (CDC) and the *connected, limited device configuration* (CLDC). A configuration is a Java virtual machine (JVM) and a minimal set of class libraries and APIs providing a runtime environment for a selected group of devices. A configuration specifies a least common denominator subset of the Java language that fits the resource constraints imposed by the family of devices for which it was developed.

MIDP

The most frequently used configuration is the CLDC which targets devices with 128-512 kB of available memory. On top of the configuration, a so-called profile supplies additional classes necessary to develop mobile applications. For CLDC, the *mobile information device profile* (MIDP) was created and is the most commonly supported profile for smart phones and even PDAs, although PDAs are not the targeted device group. For devices with at least 2 MB of available memory, the CDC would be suitable together with the *foundation profile*, but up to now there is hardly any implementation to be found. There is also a *PDA profile* with additional classes for the CLDC, but again there is no implementation up to now. In analogy to Java applets that run within a Web browser, the applications that run on top of MIDP are called midlets. MIDP provides features such as enhanced user interface, multimedia and game functionality, better connectivity, over-the-air (OTA) provisioning, and end-to-end security.

.NET compact

Microsoft puts its .NET compact framework up against J2ME which provides good integration into the PocketPC PIM applications, but is only available for Windows Mobile platforms. Besides that, it is similar to J2ME.

WAP applications

Another possibility to create applications with platform-independent access is to use WAP micro browsers to access server-side applications. WAP uses *WML2* as markup language that is based upon XHTML. The use of *wireless profiled-TCP* (WP-TCP) and WP-HTTP instead of WAP-specific network protocols leads to the fact that WAP2 browsers can directly connect to HTTP servers instead of using a WAP gateway. Nevertheless, using a gateway is reasonable, e.g., for transcoding content to fit small display sizes and memory of mobile devices. Otherwise, data transfer takes longer and users have to scroll both, horizontally and vertically to see the contents, or they have to use a special browser like Opera to scale down the contents which consumes considerable processing power. Several APIs such as the *User Agent Profile* or the *Wireless Telephony Application* (WTA) let WAP applications access user information stored on the SIM card or using device functionality such as initiating a phone call.

Data exchange and synchronization. An important specialty of mobile applications is that they are usually limited versions of desktop applica-

tions or rely on a server to share data with other devices. Thus, they need data transfer and synchronization capabilities.

OBEX, vCard, vCalendar

A well-known standard for exchanging data on a high ISO/OSI level (e.g. calendar items or tasks) regardless of the concrete data structure is OBEX (object exchange protocol). It can be compared to FTP, but is optimized for mobile use and designed as a push technology. It works on any reliable transport protocol like TCP, L2CAP (Bluetooth) or TinyTP (IrDA). It became popular through usage in Palm devices to exchange entries of PIM applications between different devices (known as beaming). Principally, objects of any type can be exchanged which makes it necessary to transfer meta-data in addition to binary object data. This can be done via a name header analogous to the MS-DOS file extensions (e.g., `gif` for an image, `vcf` for a contact or `vcs` for an entry in a calendar) or via an explicit type header that uses MIME types (e.g., `image/gif`, `text/x-vCard`, `text/x-vCalendar`). Thus, standards for the contents of object data are necessary, in addition to OBEX. Examples for such formats are vCard and vCalendar that define structures of contact and calendar data respectively. For image and text data, common Internet standards like GIF, PNG, JPEG and ASCII text are used.

Synchronization

Data of applications can be synchronized as well as applications themselves, e.g., to install them on the mobile device. Synchronization is the bidirectional exchange and transformation of data between two separate data stores in order to align their contents. Key tasks that have to be addressed by synchronization software are:

- data sub-setting and partitioning (only changed parts have to be transmitted),
- data compression (to keep network load low),
- data transformation (from one application-specific format into the other),
- transactional integrity (atomic, consistent, isolated, durable),
- detection and resolution of conflicting changes (e.g., if the same data record was modified on both sides),
- support for different transport protocols and multiple devices,
- authentication, authorization and encryption (for security reasons).

There is a trade-off between keeping the number of synchronizations low and currency of data that can hardly be addressed by synchronization software, but requires user decisions.

Proprietary synchronization software

Synchronization can be necessary between servers and end-user devices, e.g., Exchange server and Outlook client, or between different end-user devices, e.g., PDA to PC file synchronization. Communication with servers seems to be currently moving away from using proprietary network protocols towards using Web services over HTTP. A similar development can be observed for device synchronization. It is still com-

mon for mobile devices that each has its own synchronization application installed, depending on the OS, e.g., Microsoft ActiveSync, Palm HotSync or Symbian Connect. Architecture and functionality of these applications are similar. Each provides a synchronization manager that is installed on the PC, handles the synchronization process and abstracts from the underlying network connection (USB, IrDA, Bluetooth, GSM, …). Additionally, an interface is specified and a set of implementations for PIM applications on the device are delivered. Third-party manufacturers can implement the interface to enable their application for synchronization. On the remote device, data can then either be synchronized with the same application (maybe running on a different platform) or with an application of the same class, but from a different vendor.

SyncML

SyncML is standardized XML-based markup language for synchronizing data that seems to get more and more adopted in the industry, e.g., it is supported in Symbian OS 8. It distinguishes several types of synchronization like two-way synchronizations (based on flags or complete compare) and different one-way synchronizations where client data overrides server data or vice versa and either all or only new data is transferred. Conflict resolution is not addressed by SyncML, although a number of suggestions are made. Example suggestions for conflict resolution are duplication of the record, mix in of the changes if different fields are affected, overwrite data on one device or no synchronization of this record. Listing 5-2 gives an example of a SyncML file for synchronization of contact information between a Palm PDA and a PC. Upper level structure distinguishes a header part (`SyncHdr`) from a body part (`SyncBody`). The body part contains information about data source and target, data type, an identifier for the contact and contact data itself in vCard format.

The `Add` operation shown in the example defines that the contact data should be added to other contacts on the PC. Other basic operations supported are `Copy`, `Replace` and `Delete`. Additionally, there is support for transactions by using `Atomic` or `Sequence` elements to specify that operations must be processed all or none and in the given sequence.

```
<SyncML>
  <SyncHdr>
    <VerDTD>1.1</VerDTD>
    ...
  </SyncHdr>
  <SyncBody>
    <Status>...</Status>
    <Sync>
      <Target><LocURI>./contacts/</LocURI></Target>
      <Source><LocURI>address.pdb</LocURI></Source>
      <Add>
        <Meta><Type>text/x-vcard</Type></Meta>
        <Item>
          <Source><LocURI>2341</LocURI>
          <Data>
            BEGIN:VCARD VERSION:2.1
            FN:Herbert Schmidt
            N:Schmidt;Herbert
            TEL;WORK;VOICE:+49-345-4434124
            EMAIL;INTERNET:hs@gmx.de
            END:VCARD
          </Data>
        </Item>
      </Add>
    </Sync>
  </SyncBody>
</SyncML>
```

Listing 5-2. Example for a SyncML document

Questions and Exercises

1. Why do users need portals as single-point of access if they already have a Web browser as universal access application?
2. What is the relation between LDAP directory services, identity management solutions and portals?
3. What is the utility of personalization for the user of enterprise applications?
4. What are limitations of front-ends for enterprise applications that use only HTML and client-side scripting? Why does it make sense to use such thin clients instead of using rich clients?
5. What is the difference between applications implemented with Java Server Faces, Java applets or as Java application started with WebStart?
6. What is the difference between desktop integration and function integration?
7. What is the utility of using client-side office applications to access server-side enterprise applications?
8. What impact does accuracy of location-based services have on mobile applications?
9. Why are there so many different classes of mobile devices? Which type of mobile device should be used in organizational settings to support employees doing knowledge work?
10. What impact do features of mobile devices have on mobile applications?
11. What are the main challenges for developers of mobile applications and how do application frameworks like .NET compact or J2ME contribute to deal with these challenges?
12. What is the advantage of mobile devices and applications that support synchronization using an open standard like SyncML compared to synchronization via vendor-specific exchange formats?

Further Reading

The following books cover mobile technologies as well as technologies for the integration of the access layer, particularly portals.

Davydov, M. M. (2001): *Corporate Portals and E-Business Integration*. McGraw-Hill Education, 2001

Although focusing a lot on the e-business side of enterprise infrastructures instead of knowledge-oriented applications, this book is one of the few resources that not only has the buzzword portal in its title, but really covers related technologies, integration and their strategic goals.

Webster, S., McLeod, A. (2004): *Developing Rich Clients with Macromedia Flex*. Macromedia Press 2004

The authors have a great deal of experience with developing Internet applications with a variety of technologies reaching from DHTML over SVG to Flash. Although most of the book covers details about Flash and Flex programming, it also shows architectural issues and compares Flash with other technologies.

Mallick, M. (2003): *Mobile and Wireless Design Essentials*. Wiley Publishing, Indianapolis (USA) 2003

The book starts with an overview of mobile devices and network technologies and then discusses the development of mobile applications from several perspectives. It covers well-established technologies like WAP as well as recent developments like BREW, SALT and VoiceXML.

Schiller, J. (2003): *Mobile Communications*. 2nd edition, Pearson Education Limited, (UK) 2003

Schiller paints a detailed picture of the technical basis of mobile communications. It is an excellent reference for the different mobile network protocols including their security features and partly even physical foundations. It can be seen as an encyclopedic treatise of mobile networks.

6 Conclusion

In chapters 1-5, we presented the vision of an enterprise-wide knowledge infrastructure and reviewed existing technologies that can be used to realize parts of this vision today. In this chapter, we will take up the introductory thoughts on knowledge work, knowledge management, KM instruments, IS architecture and enterprise knowledge infrastructures and review results from both theoretic and empirical research projects.

Overview

We start with the discussion of alternative methods for bundling EKI services to support knowledge work (section 6.1). An activity-based approach is suggested to overcome limitations that traditional workflow- or process-oriented methods impose on employees. Afterwards, we take a look at the state-of-practice in KM by reviewing some of the findings of a comprehensive empirical study in large German companies (section 6.2). Many components of the EKI on different layers of the architecture can be bundled to support the implementation of KM instruments (section 6.3). Being well aware of the fact that KM instruments consist of organizational, people-oriented and ICT measures, we will stick to the focus on ICT support which we applied throughout this book. The dynamic environment and mobile way of doing knowledge work need flexible ICT support. Centralized systems are often not flexible enough to provide this support. We show in section 6.4 how peer-to-peer systems can complement centralized systems to provide better support for employees working in dynamic teams. We end with a brief outlook on the future of enterprise knowledge infrastructures from a technical and an architectural perspective (section 6.5).

Learning objectives

On completion of this chapter, you should be able to

- explain the concept of knowledge stance,
- give an overview of the state-of-practice of EKI,
- relate EKI services on different layers to knowledge management instruments,
- analyze advantages and disadvantages of client/server vs. peer-to-peer EKI solutions,
- elaborate on future trends in enterprise knowledge infrastructures.

6.1 Activity-based Support for Knowledge Work

Challenges for modeling knowledge work

Well structured business processes can be modeled with the help of business process modeling methods and then can be supported by workflow management systems or generally process-oriented standard software as found in many application systems targeting enterprise resource planning. Knowledge work, however, consists in large parts of weakly structured processes that can only be marginally supported by traditional workflows. EKI are aimed at supporting knowledge work. We discussed a large number of concepts, tools and systems that can be combined to support knowledge work. Concepts are needed that guide the conceptual design of context-oriented, personalized ICT support for weakly structured processes connected to activities for knowledge sharing and learning as found in knowledge work.

Process-oriented KM

Under the headline *process-oriented knowledge management*, a substantial amount of research is currently undertaken to extend workflow- and process-oriented concepts with knowledge-oriented concepts (see Maier 2004 for an overview). Examples are extensions to the architecture of integrated information systems (ARIS-KM), the business knowledge management (BKM) framework and its corresponding modeling method PROMET®I-NET, knowledge modeler description language (KMDL), and process-oriented methods and tools environment for KM (PROMOTE). Main extensions are the introduction of additional object types like knowledge object, i.e. topics of interest, documented knowledge, individual person, and skill as well as the introduction of model types like knowledge structure diagram, communication diagram and knowledge map. Ideas for concepts that have to be modeled in order to capture more detailed aspects of knowledge-intensive tasks have been implemented in tools for flexible workflow management. Even though the added concepts describe a portion of the context of knowledge work, they are not suited to model the often unstructured and creative learning and knowledge practices in knowledge work and particularly their link to business processes. Knowledge stances are one approach to accomplish this based on activity theory (see Hädrich/Maier 2004 for review and integration of activity theory and process modeling).

Knowledge stance

A knowledge stance is a class of recurring situations in knowledge work defined by occasion, context, and mode resulting in knowledge-oriented actions. It describes a situation in which a knowledge worker can, should or must switch from a business-oriented function to a knowledge-oriented action.

Figure 6-1 shows the concept of knowledge stance connecting on the one hand the business-oriented side, represented in three levels of generalization/specialization as value chain, event-driven process chain and sequence of tasks, and on the other hand the knowledge-oriented side rep-

resented by three levels of routinization as activities, actions and (automated) operations.

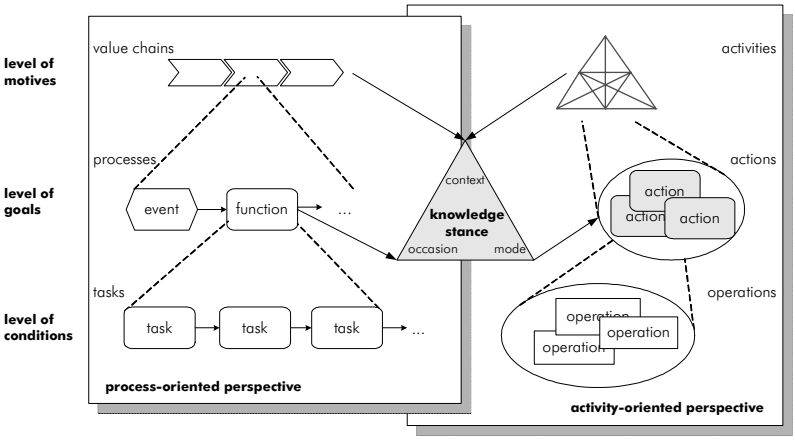

Figure 6-1. Concept of knowledge stance

In a process-oriented perspective, employees perform functions that are part of business processes. Simultaneously, they can be involved in an activity framing knowledge-oriented actions. These can be focused on the business process or pursue a motive not related to the business process, e.g., effort to build competencies. Thus, the knowledge-oriented action may have more or less direct impact on completion of a business process.

Occasion

A business process can offer several *occasions* to learn and to generate knowledge related to the core competencies of an organization. Occasions trigger knowledge stances and are associated with the tasks a business process is composed of. Occasions offer the opportunity or create the need to carry out knowledge-related actions. A knowledge stance is not limited to generating knowledge, but may also include translating and applying knowledge created outside the knowledge stance which in turn offers the possibility to generate knowledge.

Context

Context comprises all relevant dimensions suitable to describe the actual situation of an employee. It consists of data about the current process like other persons involved, desired outcomes, formal rules or other process steps as well as data about the involved knowledge-oriented activity like related community, their objectives and social rules. Person-related data comprises levels of expertise, skills, interests, relations to other persons and positions in the organizational structure. Context includes information about which knowledge products about what topics are needed, used or created. Finally, available functions and systems are defined together with privileges to access them.

Mode

Mode classifies what actions can be performed and can be described by four informing practices (Schulze 2004):

- ex-pressing is the practice of self-reflexive conversion of individual knowledge and subjective insights into information objects that are independent of the person,
- monitoring describes continuous, non-focused scanning of the environment and gathering of useful "just in case"-information,
- translating involves creation of information by ferrying it across different contexts until a coherent meaning emerges, and
- networking is the practice of building and maintaining relationships with people inside and outside the organization.

Action

Context, mode and occasion are means to specify a set of available, allowed or required knowledge-oriented actions. Examples for actions are: summarize, prioritize contents, evaluate source, indicate level of certitude, compare sources, link content, relate to prior information, add meta-information, notify, alert, ask questions, and offer interaction. In contrast to the clearly defined sequences of tasks in the process-oriented perspective, there is no predetermined flow of actions

ICT support

Depending on occasion, context and mode, it can be decided which software tools are suited to support the selected knowledge-oriented action. From an EKI perspective, those knowledge stances are of primary interest that can be supported by ICT. A straightforward approach would be to suggest adequate EKI services to users or offering context-specific check-lists that help to take all relevant aspects into consideration.

Context-dependent services

Context should be derived with as little user effort as possible. Currently opened documents on the desktop, emails in the mailbox or history of the Web browser could be used to determine some context information. This could be enriched by data about the current function in the business process the user performs and data about actions that other users took in similar situations. Furthermore, awareness services could monitor current activities of other employees relevant in the knowledge stance and thus be helpful in analyzing which cooperation partners are currently available or even engaged in similar business-oriented functions or knowledge-oriented actions respectively.

Integration

Context elements and their relation can be represented by a standardized or shared ontology. Thus, inference techniques can be applied and context can be communicated to and translated for other applications. A portal that integrates different knowledge services offered by EKI would be an intuitive starting point for the use of context information to personalize the information displayed and services offered (section 5.1, 312ff). For desktop-integrated access (section 5.2.4, 326ff), an interface agent (section 1.5.1, 49ff) could be used to present a context-specific selection of actions the user can take.

6.2 State-of-Practice of Knowledge Management

In the following, the state of practice of ICT-supported KM initiatives that has been investigated in an empirical study (Maier 2004) will be summarized in the form of theses that together describe activities concerning KMS in German organizations. Theses are based on the results obtained in a broad questionnaire conducted in 1999/2000 which were compared to the results of related empirical studies, on qualitative findings from in-depth interviews with knowledge managers as well as numerous experiences the authors collected in the last couple of years in consulting projects, conferences and discussions with researchers and practitioners. Theses are organized into four blocks: strategy, organization, KMS and economics.

KM and KMS are increasingly implemented

Strategy. Up to the last couple of years, more and more organizations showed interest in KM. So far, organizations are most successful in achieving rather basic KM goals in both, the codification and personalization side of KM, such as an improved access to existing knowledge or an improved communication and location of experts. More ambitious KM goals, such as turning implicit into explicit knowledge, or changing culture have been achieved to a much lower degree. Also, there has been a strong increase in the interest, the state of implementation and the usage of KM-related ICT systems. Most organizations have installed an advanced Intranet infrastructure during the last years which they try to extend towards an EKI. Mostly large, knowledge-intensive organizations have invested in KM. Professional services firms and a number of pioneers in a variety of industries have been leading the way. As recent related empirical studies have shown, more and more small and particularly medium-sized organizations have started to evaluate the potentials of KM.

Agreement about strategic relevance

Most organizations agree on the potentials of KM. The initiative quickly gains high visibility. Most KM initiatives report to the two highest levels of the organizational hierarchy. In many organizations, the executive board has pushed the approach. Organizations have high expectations towards KM which unfortunately can rarely be achieved within a short period of time. There is broad agreement over most empirical studies that KM is a relevant and important topic. The interviewees were also convinced about the positive effects of their KM initiative on business goals. But they admit to have difficulties in establishing clear, well-documented and measurable KM goals. The lack of a well-defined and (empirically) proven set of KM strategies is obvious as most organizations aim at a large number of different KM goals at the same time. This missing link and the measurement of the impact of KM or EKI on knowledge and business goals seem to be the most important challenges ahead.

KM initiatives as multidisciplinary effort

KM initiatives regularly comprise a strategically relevant combination of organizational and ICT instruments. Even though organizational instruments are the main drivers for a change in the handling of knowledge, it is often ICT implementations that play the role of an enabler, a catalyst for the changes to take place as they visibly change work environments of participating employees. Consequently, multiple disciplines are required in order to implement KM successfully. In a substantial part of the organizations, KM is not embedded in a single functional area, but assigned to an interdisciplinary group. Also, many KM initiatives are split into at least two separate groups within an organization with frequently a large gap between their perspectives. These are enterprise communication, human resources and organizational design on the one hand and IT on the other hand. Regularly, marketing, R&D and strategy are also major players in the KM initiative.

Organization-internal knowledge

Most KM initiatives have their focus on knowledge flows between organizational units or groups of employees within the organization's boundaries. Much less do KM initiatives aim at knowledge that crosses organizational borders. Neither do most organizations support the ICT-supported acquisition of external knowledge nor do they systematically profit from knowledge developed internally by selling knowledge products or services. Also, most KM initiatives only foster organization-internal work groups, teams, networks and communities whereas those collective structures that cross organizational borders are rarely systematically supported. Organizations have just begun to establish positions for key strategic alliance management that address these challenges at least for the most important liaisons to partner organizations.

KM as a set of independent activities

Organization. Today, large companies have a multitude of KM efforts working in parallel to tackle the problem. In many cases, several core groups start KM activities independently. Thus, in some cases even the various KM groups, teams and communities do not coordinate their efforts or even exchange knowledge which gives an indication of the complexity of the challenge. An EKI therefore should provide a platform for integrating individual ICT projects supporting KM that together could have much more positive impact on productivity of knowledge work.

KM initiatives are decentral

KM was implemented as a predominantly decentral approach leaving as much responsibility with decentral functions as possible. Responsibility for contents handled by EKI in most cases is shared between authors of knowledge elements and subject matter specialists. One of the most important goals of the implementation of EKI solutions is to increase participants' ability to actively handle ICT supported knowledge, e.g., to publish knowledge elements and information about their skills, project assignments and the like and to react to activities of other participants, e.g., to answer questions and contribute to discussions. An EKI helps to decentral-

ize corresponding KM tasks. A central unit, either a separate, permanent organizational unit or a project, frequently coordinates decentral activities.

Subject matter specialist as key role

Subject matter specialists are responsible for the majority of KM tasks surrounding EKI. They take on responsibility for one subject area or domain in the organizational knowledge base, help knowledge providers to document, link and organize their experiences, refine and organize their subject area and help knowledge seekers to locate expertise and knowledge elements. Subject matter specialists also play the role of "linking pins" for knowledge-related design tasks such as updating or mapping of ontologies as well as integration of knowledge sources into existing ontologies.

No responsibility for important KM tasks

There are also several organizations in which responsibility for KM tasks is not assigned at all. About a third of the organizations just assigned responsibility for basic tasks related to publication and distribution of knowledge, but did not pay equally high attention to what happens to it once it is documented and inserted into the organizations' knowledge bases. In a number of organizations, important tasks, such as actualization and refinement of existing knowledge, quality assurance, deletion and archival of knowledge, are not systematically assigned. This might trigger a vicious circle in which participants use EKI less frequently because they do not find what they are looking for. Thus, investments in (contents of) EKI are cut which deteriorates quality of the contents. This reduces trust in the knowledge and in turn negatively affects usage of the systems starting the circle over again.

Intranet and/or Groupware platform as foundation for EKI

Systems. By now, almost all large organization have installed an Intranet and/or a Groupware solution which can be considered a basic ICT infrastructure and predecessor of an EKI. These platforms together with a multitude of extensions and add-on tools provide good, basic KM functionality. During the past couple of years, corporate Intranet solutions have been implemented to connect employees, to support the easy sharing of electronic documents and to support access to company information. Also, organizations have installed Groupware tools in order to support teams and to master the increasing complexity of organizational structure and processes along with advanced information and communication needs.

KM functions are implemented, but not used

Large organizations have already implemented many KM-specific functions as part of an EKI or in more specific solutions, such as content management, document management or customer relationship management systems. Many functions are not used intensively, in some cases due to technical problems, but mostly because they require substantial organizational changes. Therefore, there still seem to be considerable potentials when applying ICT to KM initiatives.

KM-related ICT systems lack integration

In most organizations, a multitude of partial systems are developed without a common framework which could integrate them. Only recently, comprehensive EKI-like solutions gain market share. They offer extensive functionality integrated within one platform. Some organizations also build enterprise knowledge portals that at least integrate access to a number of organizational and organization-external ICT systems considered relevant for the KM initiative. Still, in most organizations functions and especially contents of KM-related ICT systems are largely dispersed in, e.g., messaging systems, document and content management systems as well as plain file systems. Each source of knowledge elements has its own basic taxonomy and home-grown meta-data concept, so that standardization of meta-data descriptions, extensions of taxonomies to ontologies and mapping of these seem to bear large potentials.

EKIs are highly complex systems

Comprehensive EKIs are highly complex ICT systems because of (1) the *technical complexity* of the advanced knowledge services and of the large volumes of data, documents and messages as well as contextualization and personalization data that have to be handled, (2) the *organizational complexity* of a solution that affects business and knowledge processes as well as roles and responsibilities throughout the organization and (3) the *human complexity* due to the substantial change in the handling of knowledge that is required from the organization's knowledge workers as EKI have to be integrated into their work environment.

Self-developed EKIs are prevalent

The majority of organizations relies on organization-specific developments and combinations of tools and systems rather than on standard solutions available on the market. The most important explanations for this finding might be two-fold. On the one hand, the market for EKI modules is a confusing and dynamic one. There is no leading vendor or group of vendors yet and interoperability with other EKI modules that the organizations have in place is still often difficult to realize because standardized Web services are not supported that much yet. On the other hand, organizations might fear that they loose strategic advantages if they exchange their home-grown organization-specific EKI for standard software that might not fit their needs as well.

Diversity of EKI contents has increased

Generally, more organizations handle a larger variety of knowledge contents when compared to previous studies. Many organizations use modern KM contents, like employee yellow pages, skills directories, idea and proposal systems, lessons learned and good/best practices. Recently, organizations seem to have extended the scope of their EKI to include more types of internal knowledge previously unavailable to a larger group of employees. The biggest potentials seem to lie on the one hand in the systematic management of organizational knowledge processes that connect specific states of knowledge, e.g., personal experiences, lessons learned, best practices and, finally improved business and knowledge processes. On the other hand, external knowledge bridges the gap between the organization and its environment. Many organizations do not distinguish

between these KM-related contents and more traditional contents of ICT systems, such as a broad view of all documents or the entire content of the corporate Intranet, data in data warehouses or transactional and communication data about customers and business partners. There is still considerable uncertainty in many organizations about what is or what should be considered a knowledge element.

EKI is a major, long-term investment

Economics. EKI consist of highly complex systems. Implementation of an EKI represents a major investment. A KM initiative and its support with ICT are long-term investments because they require a substantial shift in employees' roles, organizational processes and often even a change of the organizational culture. Success is dependent on network effects. The more employees participate, the more useful the EKI will be and the more these people will profit from the solution. That is why individual ICT projects aiming at specific KM initiatives should be integrated on the basis of a comprehensive, enterprise-wide knowledge infrastructure. However, most companies apply KM-related ICT systems and concepts that promise quick returns-on-investment and are reluctant to commit themselves to a substantially higher investment to integrate their knowledge-related systems and especially to changes in work processes.

Success is assessed by story-telling

Benefits of KM initiatives in general and an EKI implementation in particular so far are mostly determined by story-telling. In most organizations, this is the primary justification for the budgets allocated to the KM initiative along with references to similar activities performed by competition. The reason is that it is extremely difficult to measure knowledge directly. There are several promising approaches to the quantitative assessment of knowledge-related activities. They all require a fundamental shift in the organization's management systems and in many cases organizations are as reluctant to massively change their management paradigms as they are in fundamentally changing their ICT infrastructures.

Organizational design is crucial for a successful EKI

The higher KM expenses per participant are, the higher respondents estimate the impact of a KM initiative on business goals. KM initiatives with a formal organizational design, but a decentral assignment of responsibility, a high rate of KM activity and systematic support of communities seems to be more successful than KM initiatives which apply a different organizational design. The relatively obvious tendencies in the case of the organizational design compare with a more uncertain picture in the case of ICT systems. The implementation of an EKI alone seems to have no positive impact on business goals, but has to be combined with people-oriented and organizational instruments.

6.3 Supporting KM Instruments

In the first chapter KM instruments were distinguished according to the main medium of knowledge into person-oriented, organizational and product-oriented instruments (section 1.4, 39ff). In the following, we will briefly discuss some examples of how services on different layers of the EKI architecture could be combined in order to support KM instruments.

Competence management

Person-oriented instruments. Competence management is a central KM instrument, because people as knowledge carriers are central to KM. The value of ICT support for KM instruments lies largely not in building sophisticated stand-alone applications that foster all relevant functions, but in connecting and integrating existing ICT systems that are relevant to support the activities in an integrated way.

Employees' skills could be stored in any relational data base as it is pure data at first hand. In EKI, skills should be stored together with user data handled by identity management (section 3.3, 198ff). The number and type of skills that can be used has to be centrally defined and managed. Challenges are to create a consistent and understandable taxonomy or network of skills, to support maintaining comprehensive ratings of employees and to map skills to concepts in the central ontology or the number of ontologies used to classify documents. OWL can be used as a standardized language to document mappings between skill and topic ontologies.

Skills and interests of employees are brokered to all advanced knowledge services. They can also be published as dynamic HTML pages within the EKI. This can be done by providing XSL transformations that create HTML output from (relational) data. Results are personal homepages for all employees, or yellow pages for experts. These pages should include not only skills, but also project experiences with references to project management tools and authorship information with links to documents the individual employee has (co-)authored.

Learning paths can be shown based on the entries in the skills data base. Depending on skills and interests, employees can also get personalized references to courses offered in the learning management system or by course providers (blended learning), to books and other resources that are available in the company (digital) library or to whitepapers stored in a DMS. This functionality should be realized as part of a personalizable portal. Project staffing can also be supported by competence management systems. From the perspective of an employee, projects that require staff with certain skills could be shown, so that the employee can contact the project manager and offer his or her help.

EKI can also be used to locate experts in a certain domain. The search engine should not only find documents to a certain topic, but also people on basis of project or community memberships, authorship information

and skills. To further foster informal communication, the yellow page of the experts can also contain a picture and information about the work place, so that people are recognized if they are met on the corridor or in the cafeteria. However, protection of privacy has to be guaranteed. There is a trade-off between privacy of employees at the workplace on the one hand and useful offerings based on extensive user data on the other hand. Some organizations enable search queries that match skill profiles to people and prevent queries that ask for all skills of a person. EKIs offer the possibility to entirely decentralize maintenance of user data in general, so that employees can individually decide what personal information is stored and who has access to it.

Personal experience management

Up to now, there are hardly any ICT tools that support personal experience management. For short ideas, the notes function of Groupware clients can be used, but they are not searchable for colleagues. For experiences that are documented in a more detailed way, templates can be provided that guide explication of knowledge in a structured way. Publishing these structured, documented experiences usually requires to decide about where to store it. Often, there are a large number of potential locations, e.g., file servers, DMS or a direct transfer per email. Employees rarely go to the trouble of publishing personal experiences, if there is no systematic container that is considered useful by them.

Web logs or blogs are tools that recently gain a lot of attention for easy handling of personal experiences accessible to others. Blogs technically work similar to WCMS with a rigid structure and are organized using the dimensions time and topic. New entries can be made easily using a standard Web browser. Colleagues can search Web logs and comment entries. Web logs can further be cross-linked with other Web logs extending the usual linking capabilities that HTML offers. By commenting and referencing other Web logs, a kind of community emerges. Such a community is called *blogosphere*. Blog users can subscribe to RSS (RDF Site Summary) feeds that deliver abstracts of new entries in Web logs of other people via email. Summed up, it seems that Web logs can be used for personal experience management and close the gap between well-structured experience documents handled in DMS or file servers on the one hand and short ideas in personal information management or Groupware systems on the other hand.

Knowledge mapping

Organizational instruments. From a technical perspective, the differences between knowledge source and asset maps, knowledge application and development maps, and knowledge structure maps (sections 1.4.2, 42ff and 4.1.4, 242ff) are limited to different data sources as well as different visualization methods that are best suited for the respective type of map. Skill data bases are main sources for people-centered knowledge maps. Document and content management systems as well as learning management systems can serve as sources for knowledge maps that focus

documented knowledge. Finally, learning management systems, process warehouses and workflow management solutions can be tapped in order to build knowledge application and development maps. Central to all types of maps is the concept of ontology that can be used to relate contents of different source systems to each other by means of an inferencing engine (section 3.2.2, 175ff).

Challenges for visualization algorithms are to keep a good balance between the display of details and the need to keep maps concise and understandable as well as to support straightforward navigation in knowledge maps that avoids the "lost-in-cyberspace" problem often encountered by users in visualization systems that employ unusual metaphors. Often, it is desirable to visualize relations between people or documents with similar skills or content, e.g., by displaying them close together. The calculations that have to be performed to accomplish such layouts are complex which makes it difficult to use standard transformation technologies such as XSLT as means to generate such maps. XSLT would otherwise be a good choice as the data can often be extracted from source systems in an XML format and the resulting SVG maps (scalable vector graphics) can be displayed using a plug-in to a Web browser. However, XSLT is suited for simple selection and regrouping of elements rather than complex computation. Therefore, maps must either be generated entirely on the server using Servlets or similar server-side technologies and then sent to the client as static images, or on the client side using technologies like Java Applets or Macromedia Flash (section 5.2.2, 323ff).

Knowledge process reengineering

EKIs can systematically be used to collect information about how personal experiences are step-by-step turned into redesigned knowledge processes. This requires relating a variety of knowledge containers, such as personal experiences, lessons learned or good/best practices gained during the execution of processes to sources describing these processes. Thus, systematic collection and handling of information about processes is the basis for knowledge process reengineering. From the EKI perspective, this is a big challenge as current solutions for business process management (BPM) are often autonomous systems with proprietary formats for data and meta-data. There is currently no well-established standard format neither for the description of processes, nor for capturing meta-data that has to be documented along with processes. An example for a general standard is the Open Information Model standardized by the Object Management Group that provides a standard meta-data model described in UML and serialized in XML for meta-data describing

- *processes*, e.g., event-driven process chains,
- *data* bases and warehouses, e.g., data models, descriptions of data bases,

- *object-oriented elements* describing analysis, design and implementation, e.g., class diagrams, descriptions of components, use cases, activity diagrams,
- as well as *knowledge*, e.g., ontologies, descriptions of resources.

Publishing process descriptions in standard formats has only recently found its way into the feature list of BPM solutions, e.g., ARIS Web Publisher.

Communities and knowledge networks

The ICT support for both communities and knowledge networks comprises some kind of Web publishing and authoring systems. Many systems can be distinguished that differ mainly with respect to the way they structure contents and authoring policies they impose on users.

Community builder, also called community home spaces, are based on discussion groups (section 4.3.1, 268ff) and enrich them with additional functionality, e.g., shared information spaces, chats and yellow pages. Their structure uses topic, time and user as dimensions. Topic determines the top-level structure of discussions. Different threads are listed on the second level of the structure. A thread is a collection of time-ordered messages on one subject related to the topic submitted by different users. The remaining structure is determined by time and user. Messages can be added by any member of the community. Existing messages can not be altered. Only the system administrator can delete single messages, e.g., if they contain offending content. Discussion groups are used to exchange opinions about topics or to get advice from experts concerning topics that are not (yet) well documented.

Wikis focus solely on topics (section 4.3.2, 276ff). They allow joint editing of contents and can be used as a tool for knowledge explication in communities. Authors of contents take a back seat, contents come to the fore. Editing is open to anybody. In its original application, wiki users are even anonymous. Versioning enables recovery of overwritten or deleted content that was more valuable than its replacement. Wikis could be used in communities to jointly document more mature knowledge and reach consensus about topics if the resulting knowledge product as a joint effort is important, e.g., a thesaurus, an electronic handbook or manual. Using wikis in organizations will certainly raise questions about how to structure contents, how to handle user privileges and the like which could quickly turn this simple and effective solution into yet another tool for Web content management.

Lessons learned and good/best practices

Product-oriented instruments. As mentioned in the case of knowledge process reengineering, EKIs need to manage entire knowledge processes, sometimes also called knowledge life-cycles. Organizations must define a number of containers that comprise different types of knowledge. In addition to offering support for managing personal experiences, it is particularly important that EKIs are structured to support organizational instru-

ments that specifically target knowledge developed in groups of people. Depending on commitment and legitimization, lessons learned, good and best practices are examples for such containers. From an EKI perspectives, these instruments require templates that specify fields that have to be filled when a lesson learned or a good practice is developed. Integration services are required to relate several knowledge elements to each other and to participants that created and/or applied them. Additionally, contextualization allows for e.g., locating experts that are identified on the basis of the lessons learned or good practices they co-authored. Competence management systems can be dynamically updated by the information gained about people involved with specific knowledge elements. Collaboration functions can finally be used to contact, communicate and work with experts or other knowledge providers in order to ease the reuse of lessons learned and good practices.

Semantic content management

We have already argued that document management and content management increasingly are integrated in actual implementations of CMS and DMS software, sometimes called enterprise content management (section 4.2.2, 258ff). Semantic content management builds upon integrated enterprise content management solutions that support both, content and document life-cycles including creating, capturing, annotating, versioning and archiving of documents, publishing on the Web and in other media as well as workflow functionality.

Semantic content management requires that enterprise content management is based on a single organization-wide ontology or a number of shared and mapped ontologies that can be used to relate contents and documents (1) to each other, to occurrences and versions in different sources as well as to resource descriptions, (2) to persons, e.g., experts with skills in the described domain, or members of the project team which created a resource and (3) to processes, e.g., the process step that is described in a document or the process in which a document was created or can be applied. The value of the system rises with the number of meaningful links it establishes between knowledge elements from different knowledge sources. Semantic content management also profits from semantic discovery services, e.g., a search engine which uses ontologies in order to find relevant content in the growing amount of data handled by an EKI.

6.4 Centralized versus Distributed Architectures

Centralized EKI architectures

Many EKI solutions implemented in organizations and offered on the market are centralistic client/server solutions (section 6.2). In larger organizations, multiple servers can be accessed in the Intranet that offer specific services. EKI integration often happens with the help of an additional

server that draws together data from various data and knowledge sources. The ideal architecture as presented in 1 and applied as the main structure of the book therefore applies the metaphor of a central knowledge server that integrates all knowledge shared in an organization and offers a variety of services to participants.

Benefits of peer-to-peer EKI

An alternative to offer all EKI services on a centralized system is to implement them on distributed peers. Recently, there have been several attempts to design information sharing systems or even EKI to profit from the benefits of the peer-to-peer metaphor. This approach promises to resolve some of the shortcomings of centralized EKI, e.g.,

- *costs*: to reduce the substantial costs of design, implementation and maintenance of a centralized knowledge server,
- *external knowledge*: to overcome the limitations of an EKI that focuses on organization-internal knowledge whereas many knowledge processes cross organizational boundaries and
- *integration to desktop*: to seamlessly integrate the shared knowledge workspace with an individual employee's personal knowledge workspace.
- *flexibility*: semi-autonomous organizational units can easily create and share knowledge in temporary, dynamic networks of participants with the help of tools and ontologies that fit their domain,
- *direct communication*: knowledge is exchanged directly without central units that often act as unwanted filter to knowledge,
- *acceptance*: local storage together with efficient management of access privileges reduces barriers to provide knowledge that some central EKI solutions experience.

Implementation of a peer-to-peer-based EKI may either follow a pure peer-to-peer architecture or a hybrid architecture where the peers are assisted by additional centralized servers or by super-peers (see "Peer-to-peer architecture" on page 53). The more functionality for coordination is required in a peer-to-peer system, as is the case in an EKI, the more likely it is that some kind of assistance by a server is needed to coordinate parts of the system.

Additional services

Principally, services to be implemented remain the same as in the centralized case, except those that target specific requirements of the peer-to-peer architecture. Firstly, this is the need for discovery and communication with peers and eventually with centralized servers. Peers may dynamically join the network or go offline. Secondly, services to handle integration of the distributed knowledge base are needed, since it is ideally spread over all participating peers. Each peer has a knowledge repository that may be structured according to an individual ontology whereas central repositories managed by e.g., subject matter specialists might offer domain-specific

ontologies and tools to map ontologies. Thus, the following services need to be provided in addition:

- *Infrastructure services* help locating peers, exchanging data with other peers and assuring security of the personal knowledge base. Servers may provide a connection to additional, shared data and knowledge sources and services for lookup and message handling that improve efficiency of the peer-to-peer infrastructure.
- *Integration services* that handle meta-data of the knowledge objects in the personal knowledge base and establish a personal ontology. Centralized replication services ensure synchronization of peers that sometimes work offline. The knowledge base comprises private, protected and public areas. Private workspaces contain information that is only accessible for its owner. Public workspaces hold knowledge objects that are accessible by all authenticated users. Protected workspaces contain knowledge objects that are accessible to a single or a group of peers that the owner explicitly grants access.

Figure 6-2 visualizes three communities, which may be project teams, work groups, knowledge networks or communities-of-interest, practice or purpose. Members of communities have a shared context, e.g., they are interested in the same topics or work on common tasks. The context becomes apparent in a shared ontology, in communication relationships between the members and in jointly accomplished knowledge-oriented actions and processes. In contrast to centralized EKI, the structure of the peer-to-peer network may directly support this shared context, e.g., by means of flexible community workspaces, support of ad-hoc communication between members or support of joint knowledge processes.

Knowledge processes

Some examples for knowledge processes are depicted in Figure 6-2 and related to a distributed EKI:

- *Externalization:* Information is externalized with regular applications and results in documents that are stored on the local peer together with attached meta-data, reflecting an individual or shared ontology.
- *Distribution:* (Ad-hoc) distribution of information between peers with shared context is substantially less effort than transferring it to a centralized server with a generalized ontology.
- *Submission*: New knowledge elements are published and distributed towards a topic-oriented network. Subject matter experts assess quality of information and can act as hubs for distribution of information.
- *Search:* Searching involves local search and search on connected peers with the same user interface and methods. Efficient search is one of the major challenges in pure peer-to-peer networks. Super peers that contain a directory of peers and resources organized according to topics might be one way to overcome the challenges.

- *Acquisition:* Knowledge is retrieved from peers in the neighborhood. Since it is not determined that these are always online, a peer-to-peer network needs to offer ways to secure provision of essential documents, e.g., through caching or backup on other peers or by centralized servers.
- *Application:* Any resources that have been retrieved from other peers can be applied and internalized. The shared ontology and corresponding annotations and meta-data including feedback on retrieved resources help interpreting and thus internalizing information.
- *Feedback:* Quality of information can be improved by feedback which also reports useful experiences with repeated application of resources in different situations.

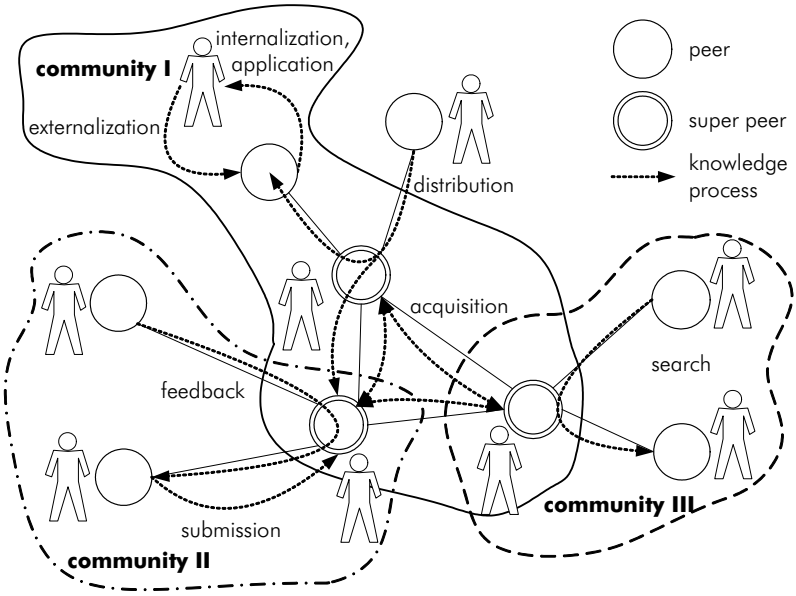

Figure 6-2. Knowledge processes and shared context in p2p networks

Comparison. An EKI may be implemented following predominantly a centralized or a decentralized approach. The decision about which approach may be applicable depends on the actual design and goals of an organization's KM initiative. Table 6-1 compares both alternatives.

Table 6-1. Centralized and distributed EKI compared

characteristics	centralized EKI	distributed EKI
type of knowledge	stable, objectified, proven, generalized knowledge	ad-hoc, subjective, unsecured, specific knowledge
knowledge repository	centralized repository with an organization-wide ontology	decentralized repositories with ontologies reflecting shared contexts of groups
contents	(local) best practices, (approved) knowledge products, secured knowledge as well as ideas, experiences and individual contents	individual contents, ideas, results of group sessions and experiences, lessons learned, good practices
control	centralized	decentralized
goals	secure and reuse organizational knowledge	store knowledge according to individual requirements
KM orientation	technology-oriented KM	human-oriented KM
organizational design	hierarchy, departments	project teams, communities, ad-hoc groups

Centralized EKI

A centralized approach fits best with a technology-oriented KM strategy where securing and reuse of knowledge are primary goals. A centralized repository is built in which either all or only a specific part of the members of the organization contribute and retrieve knowledge which can be centrally controlled.

Distributed EKI

In contrast, a decentralized approach seems to be most suited to support ad-hoc, individual knowledge and spontaneous interaction between individuals in groups. There is no centralized control and administration, neither of knowledge structures nor of technical maintenance and administration tasks. This approach best fits a human-oriented KM strategy where focus is on fostering social interaction and collaboration to jointly create innovative solutions.

6.5 The Future of Knowledge Infrastructures

Numerous technological trends influence the development of EKI. Many concepts, tools and standardization efforts concurrently take place on various layers of the architecture. This makes it impossible to reflect current developments for all services touched in this book. Thus, we will limit our considerations to the areas of network infrastructures, devices for access-

ing EKI, integration technologies and the market for comprehensive EKI solutions.

Network infrastructures

Besides the ever increasing size of network bandwidth, e.g., GBit Ethernet to the workplace, and usage of fiber instead of copper cables, there is also a large number of emerging standards, especially for wireless networks. Even before Bluetooth, WLAN and UMTS have been widely adopted, the next generation of standards are under development. Examples are nearfield communication (Zigbee) for short ranging and quickly established connections, ultra wideband (UWB) as cable-less replacement of USB and WiMax (Worldwide Interoperability for Microwave Access) for long-reaching high bandwidth connections.

Always online

The trend clearly is towards mobile workers being always online, connected to a world-wide inter-operable network of networks that permanently offers personalized services in the context of time, location and the states of co-workers and connected servers. Seamless hand over between networks using different protocols is a substantial challenge here.

Access devices

Mobile devices undergo an ambivalent development. On the one hand, there is a strong convergence between device classes. PDAs implement GSM and GPRS communication means and mobile phones implement PIM functionality. Both device classes get enhanced with rich multimedia capabilities formerly found in specialized devices like MP3 players and digital photo cameras. On the other hand, there is an increasing number of devices that can not be sorted into an existing category. Blackberry email push technologies result in new specialized devices. The number of different notebook classes is increasing in both directions to desktop replacement and mini-sub-notebooks. Both, convertible notebooks and PDAs with VGA screen resolution close the gap to ebook readers. Increasing sizes of disc (40GB in 1 inch), flash (8 GB) and main memory (256 MB in PDAs) as well as faster CPUs (>600 MHz in PDAs) enable new applications and allow mobile workers to have important data always with them.

EKI accessed by variety of devices

Enterprise information and communication infrastructures are still focussed on the desktop PC that is connected to an organization's Intranet. The challenge will be to provide appropriate access to personalized services offered by whatever system run by the organization or by whatever provider in the Internet, no matter with what type of device via whatever network connection a mobile knowledge worker accesses the organization's EKI.

Data sources

Generally, integration of data sources poses an increasing problem to the effective handling of documented knowledge in organizations. EKI solutions need to connect enterprise resource planning systems, document management systems, content management systems, e-learning platforms, data warehouses and business intelligence tools, product data management systems, CAD tools, project management tools, customer relationship management systems, visualization tools etc. The problem is that most, if not all of these tools and systems implicitly attempt to play the role of the

leading provider of documented knowledge within a certain context, e.g., a project, a customer contact, a product, a process or a learning situation.

Integration

Standardized languages and corresponding tools could broker semantics in the form of ontologies back and forth between these tools and systems. However, semantic integration is time-consuming and costly and thus limited to application areas where reuse of knowledge objects is highly likely to pay off. Therefore, we have some way to go until we will see smart, interconnected services in an EKI that truly benefit from the fast developments in the hardware sector and the large amounts of data collected in heterogeneous applications. Standardization of languages that help to semantically connect data sources and applications, e.g., based on Semantic Web technologies, point in the right direction, though they suffer from their tremendous complexity.

Furthermore, they need to be integrated with concurrent developments in the software sector. In the last few years there has been a clear trend towards highly modular infrastructures with decoupled systems that are integrated with dedicated middleware systems. The hype around Web services and service-oriented architectures accelerates this development. But when the application of Web services and Semantic Web technologies leaves prove-of-concept scenarios and moves towards real-world scenarios, it becomes apparent that complexity is not entirely mastered, but to some extent only relocated. Composition of Web services to form processes remains as much a substantial challenge as semantic description of Web services and the data exchanged between applications.

Market for EKI solutions

Major vendors seem to understand the need for comprehensive, yet modular systems and have supplemented their offerings to provide the whole range of modules from application server to identity management and Web service composition. Recently, the market for centralized EKI solutions has seen a number of strategic alliances, mergers and acquisitions. Many vendors of KM tools have vanished from the market or their technologies have been integrated in product offerings of major software companies such as IBM or Microsoft, or of leading vendors of EKI technology, such as Hyperwave or Open Text. However, there are still many small companies offering innovative tools that pose substantial challenges with respect to requirements such as scalability, integration with other application systems and platforms as well as security. Many innovative tools only target small groups of users or single users, e.g., in the case of some visualization tools.

Combination of centralized and peer-to-peer architectures

Most EKI solutions so far are comprehensive, server-based, organization-wide solutions that target large organizations with hundreds of knowledge workers. Peer-to-peer EKI promise to resolve some of the shortcomings of centralized KMS, however, major challenges still lie ahead until peer-to-peer systems can truly be called EKI and can be used to support the still growing share of users involved in knowledge work. Conse-

quently, future EKI solutions might attempt to combine advantages of peer knowledge workspaces that directly exchange knowledge on the one hand with servers that help to achieve the advantages of integrated and quality assured partial organizational knowledge bases on the other hand.

ERP as one pillar of ICT infrastructure

Many organizations have undergone substantial reorganization during the last ten years when they exchanged proprietary, unintegrated solutions for standard ERP systems. Both, horizontal and vertical integration of structured data stored in relational data bases has substantially improved transparency of business transactions in organizations, increased data quality and flexibility of the organization's business processes and reporting system, just to name two. Thus, ERP solutions are one important pillar of information and communication infrastructures in many business organizations. What is left, is the integration of semi-structured data dispersed in numerous servers and individual PCs, largely unintegrated and consequently hindering knowledge work.

EKI as the other pillar

This sort of data is typically addressed by KM solutions. The implementation of KM technology in organizations has entered a new stage. It is not anymore the quest for the best individual tool targeting a specific KM problem that organizations should engage in. Organizations should now systematically build a second pillar in their information and communication infrastructures. EKI focus integration of valuable knowledge elements needed in weakly structured knowledge processes as much as ERP solutions targeted integration of business data needed in well-structured business processes. Ultimate goal of implementing EKI in organizations is to substantially increase productivity of weakly-structured knowledge work as much as ERP and other process-oriented systems have increased productivity of well-structured data and service work.

Bibliography

Alavi, M., Leidner, D. E. (2001): Review: Knowledge Management and Knowledge Management Systems: Conceptual Foundations and Research Issues, in: Management Information Systems Quarterly - MISQ, Vol. 25, No. 1, 2001, 107-136

Alonso et al. (2004): Web Services - Concepts, Architectures and Applications, Springer, Heidelberg

Amberg, M., Holzner, J., Remus, U. (2003): Portal-Engineering - Anforderungen an die Entwicklung komplexer Unternehmensportale, in: Proceedings of the Wirtschaftsinformatik 2003, Medien, Märkte, Mobilität, Physica, Heidelberg, 795-817

Asprey, L., Middleton, M. (2003): Integrative Document and Content Management: Strategies for Exploiting Enterprise Knowledge, Idea Group, Hershey

Austin, J. (1962): How to Do Things With Words, Oxford University Press, Oxford

Barney, J. B. (1991): Firm Resources and Sustained Competitive Advantage, in: Journal of Management, Vol. 17, No. 1, 1991, 99-120

Baumgartner, P., Payr, S. (1994): Lernen mit Software. Series Digitales Lernen, Österreichischer Studienverlag, Innsbruck

Chase, R. L. (1997): Knowledge Management Benchmarks, in: Journal of Knowledge Management, Vol. 1, No. 1, 1997, 83-92

Christ, O. (2003): Content Management in der Praxis. Erfolgreicher Aufbau und Betrieb unternehmensweiter Portale, Springer, Berlin

Clark, R.C., Mayer, R.E. (2002): E-Learning and the Science of Instruction. Proven Guidelines for Consumers and Designers of Multimedia Learning, Jossey-Bass/Pfeiffer, San Francisco

Clemens, P., Northorp, L. (1996). Software Architecture: An Executive Overview, CMU Software Engineering Institute Technical Report, Carnegie Mellon University

Davenport, T.H., Jarvenpaa, S.L., Beers, M.C. (1996): Improving Knowledge Work Processes, in: Sloan Management Review, Vol. 37, No. 4, Summer 1996, 53-65

De Clercq, J., Rouault, J. (2004): An Introduction to Identity Management, Hewlett Packard Developer Whitepaper, http://devresource.hp.com/drc/newsletters/2004/newsletter0704.jsp, last accessed 2004-12-07

DeSanctis, G., Gallupe, R. B. (1987): A Foundation for the Study of Group Decision Support Systems, in: Management Science, Vol. 33, No. 5, 1987, 589-609

Dourish, P. and Belotti, V. (1992): Awareness and Coordination in Shared Workspaces, in: Proceedings of the Conference on Computer-Supported Cooperative Work - CSCW'92 (Okt. 31-Nov. 4, Toronto, Canada). ACM, 107-114

Eisenhardt, K. M., Martin, J. A. (2000): Dynamic Capabilities: What Are They?, in: Strategic Management Journal, Vol. 21, 2000, 1105-1121

Eppler, M.J. (2003): Making Knowledge Visible through Knowledge Maps: Concepts, Elements, Cases, in: Holsapple, C.W. (ed.): Handbook on Knowledge Management, Vol. 1: Knowledge Matters. Springer, Berlin, 189-205

Fowler, M. (ed.) (2003): Patterns of Enterprise Application Architecture, Addison-Wesley, Boston

Frielitz, C., Hippner, H., Wilde, K. D. (2002): eCRM als Erfolgsbasis für Kundenbindung im Internet. in: Bruhn, M., Stauss, B. (eds.): Electronic Services. Diensteistungsmanagement-Jahrbuch 2002, Wiesbaden, 537-562

Garvin, D. A. (1993): Building a Learning Organization, in: Harvard Business Review, Vol. 71, No. 7-8, 1993, 78-91

Gilliland-Swetland, A. J. (2002): Setting the Stage - Introduction to Metadata, http://www.getty.edu/research/conducting_research/standards/intrometadata, last accessed 2004-12-07

Göggler, M. (2003): Suchmaschinen im Internet: Funktionsweisen, Ranking-Methoden, Top-Positionen, Springer, Berlin

Grant, R. M. (1991): The Resource-Based Theory of Competitive Advantage: Implications for Strategy Formulation, in: California Management Review, Vol. 33, No. 3, 1991, 114-135

Grant, R. M. (1996a): Prospering in Dynamically-Competitive Environments: Organizational Capability as Knowledge Integration, in: Organization Science, Vol. 7, No. 4, July-August 1996, 375-387

Grant, R. M. (1996b): Toward a Knowledge-Based Theory of the Firm, in: Strategic Management Journal, Vol. 17, 1996, Winter Special Issue, 109-122

Gulbins, J., Seyfried, M., Strack-Zimmermann, H. (2002): Dokumentenmanagement, Springer, Berlin

Hall, R. (1992): The Strategic Analysis of Intangible Resources, in: Strategic Management Journal, Vol. 13, No. 2, 1992, 135-144

Heckhausen, H. (1988): Motivation und Handeln, Springer, Berlin

Heisig, P., Vorbeck, J., Niebuhr, J. (2001): Intellectual Capital, in: Mertins, K., Heisig, P., Vorbeck, J. (eds.): Knowledge Management. Best Practices in Europe, Berlin et al. 2001, 57-73

IANA (1999): Complete UDP/TCP Port Number List, http://linux.cudeso.be/cdp/UDP_TCP_PortList.pdf, last accessed 2004-12-07

IANA (2001): The Official IANA Registry of MIME-Types, http://www.iana.org/assignments/media-types/, last accessed 2004-12-07

Jennex, M., Olfman, L. (2003): Organizational Memory, in: Holsapple, C. W. (ed.): Handbook on Knowledge Management. Vol. 1, Berlin 2003, 207-234

Kaplan, R. S., Norton, D. P. (1996): Using the Balanced Scorecard as a Strategic Management System, in: Harvard Business Review, Vol. 74, No. 1-2, 1996, 75-85

Katz, H (ed.) (2004): XQuery from the Experts - A Guide to the W3C XML Query Language, Addison-Wesley, Boston

Kirn, S. (2002): Kooperierende intelligente Softwareagenten, Wirtschaftsinformatik, Vol. 44, No. 1, 53-63

Kohonen, T. (1995): Self-Organizing Maps, 2nd edition, Springer, Berlin

Korpela, J. (2004): A Tutorial on Character Code Issues, http://www.cs.tut.fi/~jkorpela/chars.html, last accessed 2004-12-04

Leavitt H.J. (1965): Applied Organisational Change in Industry: Structural, Technological, and Humanistic Approaches, in: March J. (ed.), Handbook of Organisations, Rand McNally & Co. Chicago, 1144-1170

Leonard-Barton, D. (1992): The Factory as a Learning Laboratory, in: Sloan Management Review, Vol. 34, No. 1, 1992, 23-38

Leuf, B., Cunningham, W. (2001): The Wiki Way: Quick Collaboration on the Web, Boston, Addison-Wesley

Levi, D., Meyer, P., Stewart, b. (1998): SNMPv3 Applications, http://www.ietf.cnri.reston.va.us/rfc/rfc2273.txt?number=2273, last accessed 2004-12-07

Liebeskind, J. P. (1996): Knowledge, Strategy, and the Theory of the Firm, in: Strategic Management Journal, Vol. 17, 1996, Winter Special Issue, 93-107

Lyman, P. et al. (2003): How Much Information 2003, http://www.sims.berkeley.edu/research/projects/how-much-info-2003, last accessed 2004-12-07

Maier, R. (2004): Knowledge Management Systems. Information and Communication Technologies for Knowledge Management, 2nd edition, Springer, Berlin

Maier, R., Sametinger, J. (2004): Peer-to-Peer Information Workspaces in Infotop, in: International Journal of Software Engineering and Knowledge Engineering, Vol. 14, No. 1, 79-102

Mayer, R.E. (2001): Multimedia Learning, Cambridge, Cambridge University Press

Mertens, P., Höhl, M. (1999): Wie lernt der Computer den Menschen kennen? Bestandsaufnahme und Experimente zur Benutzermodellierung in der Wirtschaftsinformatik, in: Wirtschaftsinformatik 41 (1999) 3, S. 201-209

Nonaka, I. (1994): A Dynamic Theory of Organizational Knowledge Creation, in: Organization Science, Vol. 5, No. 1, 1994, 14-37

Nonaka, I., Takeuchi, H. (1995): The Knowledge Creating Company, New York 1995

O'Dell, C., Grayson, C. J. (1998): If We Only Knew What We Know: Identification and Transfer of Internal Best Practices, in: California Management Review, Vol. 40, No. 3, 1998, 154-174

Oracle (2003): Oracle Identity Management Concepts and Architecture, http://www.oracle.com/technology/products/id_mgmt/pdf/OracleAS_IDmanagement_10g_TWP.pdf, last accessed 2004-12-07

Porter, M. (1985): Competitive Advantage: Creating and Sustaining Superior Performance, Free Press, New York

Porter, M. E. (1980): Competitve Strategy: Techniques for Analyzing Industries and Competitors, New York

Porter, M. E. (1990): The Competitive Advantage of Nations, London 1990

Prahalad, C. K., Hamel, G. (1990): The Core Competence of the Corporation, in: Harvard Business Review, Vol. 68, No. 5-6, 1990, 79-91

Priebe, T. (2004): INWISS - Integrative Enterprise Knowledge Portal. Demonstration at the 3rd International Semantic Web Conference (ISWC2004), Hiroshima, Japan, November 2004, http://www.inwiss.org, last accessed 2004-12-07

Reichwald, R. (ed.) (2002): Mobile Komunikation - Wertschöpfung, Technologien, neue Dienste, Gabler, Wiesbaden

Röder, S. (2003): Eine Architektur für individualisierte computergestützte Lernumgebungen, Dissertation, Lang, Frankfurt a.M.

Roth, J. (2002): Mobile Computing - Grundlagen, Technik, Konzepte, dpunkt, Heidelberg

Ruggles, R. L. (ed.) (1997): Knowledge Management Tools, Boston (MA)

Sangoma (2004a): Sangoma X.25 Tutorial, http://www.sangoma.com/x25.htm, last accessed 2004-12-07

Sangoma (2004b): Sangoma Frame Relay Tutorial, http://www.sangoma.com/fr.htm, last accessed 2004-12-07

SAP (2003): SAP Mobile Infrastructure: An Open Platform for Enterprise Mobility, http://www.sap.com/solutions/netweaver/brochures/index.aspx, last accessed 2004-12-07

Scheer, A.-W. (1994): EDV-orientierte Betriebswirtschaftslehre: Grundlagen für ein effizientes Informationsmanagement, Springer, Berlin

Scheer, A.-W. (2001): ARIS - Modellierungsmethoden, Metamodelle, Anwendungen, Springer, Berlin

Schrader, J. (1994): Lerntypen bei Erwachsenen: empirische Analysen zum Lernen und Lehren in der beruflichen Weiterbildung, Deutscher Studien-Verlag, Marburg

Searle, J.R. (1969): Speech Act: An Essay in the Philosophy of Language, Cambridge University Press, Cambridge

Shannon, C. E., Weaver, W. (1949): The Mathematical Theory of Communication, University of Illinois Press, Urbana (IL)

Siegel, J. (2000): CORBA 3 Fundamentals and Programming, 2nd edition, Wiley, New York

Skyrme, D., Amidon, D. (1997): The Knowledge Agenda, in: The Journal of Knowledge Management, Vol. 1, No. 1, 1997, 27-37

Spence, R. (2001): Information Vizualization, Addison-Wesley, Harlow (GB) et al.

Spender, J.-C. (1994): Organizational Knowledge, Collective Practice and Penrose Rents, in: International Business Review, Vol. 3, No. 4, 1994, 353-367

Spender, J.-C. (1996): Organizational Knowledge, Learning and Memory: Three Concepts in Search of a Theory, in: Journal of Organizational Change Management, Vol. 9, No. 1, 1996, 63-78

Spender, J.-C. (1996a): Making Knowledge the Basis of a Dynamic Theory of the Firm, in: Strategic Management Journal, Vol. 17, 1996, Winter Special Issue, 45-62

Stein, E., Zwass, V. (1995): Actualizing Organizational Memory with Information Systems, in: Information Systems Research, Vol. 6, No. 2, 1995, 85-117

Stewart, T. A. (1997): Intellectual Capital: The New Wealth of Organisations, New York

Suresh Raj, G. (2002): The Advanced COM/DNA Tutorial, http://my.execpc.com/~gopalan/com/com_tutorial.html, last accessed 2004-12-07

Sveiby, K.-E. (1997): The New Organizational Wealth. Managing and Measuring Knowledge-Based Assets, Berrett-Koehler, San Francisco

Sveiby, K.-E., Lloyd, T. (1987): Managing Knowhow, London; based on: Sveiby, K.-E., Risling, A.: Kunskapsföretaget (in Swedish; the Know-How Organization), Malmö 1986, also published in German as Sveiby, K.-E., Lloyd, T.: Das Management des Know-How. Führung von Beratungs-, Kreativ- und Wissensunternehmen, Frankfurt/New York

Sweller, J. (1999): Instructional Design in Technical Areas, ACER Press, Camberwell

Szyperski, C. (1997), Component Software: Beyond Object-Oriented Programming, Addison-Wesley, London

Turowski, K, Pousttchi, K. (2004): Mobile Commerce - Grundlagen und Techniken, Springer, Berlin

W3C (1999a): XML Path Language (XPath) Version 1.0, http://www.w3.org/TR/1999/REC-xpath-19991116, last accessed 2004-12-07

W3C (1999b): XSL Transformations (XSLT) Version 1.0, http://www.w3.org/TR/1999/REC-xslt-19991116, last accessed 2004-12-07

W3C (2001): Web Services Description Language (WSDL) 1.1, http://www.w3.org/TR/2001/NOTE-wsdl-20010315, last accessed 2004-12-07

W3C (2003): SOAP Version 1.2 Part 1: Messaging Framework, http://www.w3.org/TR/2003/REC-soap12-part1-20030624/, last accessed 2004-12-07

W3C (2004a): Extensible Markup Language (XML) 1.0 (Third Edition), http://www.w3.org/TR/2004/REC-xml-20040204/, last accessed 2004-12-07

W3C (2004b): XML Schema Part 1: Structures Second Edition, http://www.w3.org/TR/2004/REC-xmlschema-1-20041028/, last accessed 2004-12-07

W3C (2004c): RDF Primer, http://www.w3.org/TR/2004/REC-rdf-primer-20040210/, last accessed 2004-12-07

W3C (2004d): RDF Vocabulary Description Language 1.0: RDF Schema, http://www.w3.org/TR/2004/REC-rdf-schema-20040210/, last accessed 2004-12-07

W3C (2004e): OWL Web Ontology Language Reference, http://www.w3.org/TR/2004/REC-owl-ref-20040210/, last accessed 2004-12-07

WFMC (1995): The Workflow Reference Model, http://www.wfmc.org/standards/docs/tc003v11.pdf, last accessed 2004-12-07

WFMC (2002): An Introduction to Workflows, Section from Workflow Handbook 2002, http://www.wfmc.org/information/introduction_to_workflow02.pdf, last accessed 2004-12-04

Wiig, K. M. (1997): Integrating Intellectual Capital and Knowledge Management, in: Long Range Planning, Vol. 30, No. 3, 1997, 399-405

Willke, H. (1998): Systemisches Wissensmanagement, UTB, Stuttgart

Winograd, T., Flores, F. (1986): Understanding Computers and Cognition: A New Foundation for Design, Ablex Publ. Corp., Norwood

Wooldridge, M., Jennings, N.R. (1995): Intelligent Agents - Theory and Practice, in: The Knowledge Engineering Review 10(2)

Zack, M. H. (1999): Managing Codified Knowledge, in: Sloan Management Review, Vol. 40, No. 4, Summer 1999, 45-58

Zigurs, I., Buckland, B. K. (1998): A Theory of Task/Technology Fit and Group Support Systems Effectiveness, in: Management Information Systems Quarterly - MISQ, Vol. 22, No. 3, 1998, 313-334

Zschau,O., Traub, D., Zahradka, R. (2002): Websites professionell planen und betreiben, 2nd edition, Galileo Press, Bonn

Index