Dissertation

Sustainable Web-Based Organisation of Project-Related Information and Knowledge

Ausgeführt zum Zwecke der Erlangung des akademischen Grades
eines Doktors der technischen Wissenschaften

unter der Leitung von

o.Univ.Prof. Dr.techn. A Min Tjoa
Institut für Softwaretechnik und Interaktive Systeme
a.o.Univ.Prof. Dr.rer.soc.oec. Dieter Merkl
Institut für Softwaretechnik und Interaktive Systeme

eingereicht an der
Technischen Universität Wien
Fakultät für Technische Naturwissenschaften und Informatik

von

Dipl.-Ing. Alexander Schatten
Matrikelnummer: 8925164
Gallitzinstr. 7-13/7/7
A-1160 Wien

Wien, im Juli 2003

Kurzfassung

Projektkooperation, besonders in verteilten Projektteams, sind zunehmend häufiger anzutreffen — in universitären Forschungsprojekten ebenso wie im kommerziellen Umfeld. Planung, Management und Zusammenarbeit der Projektteams sind immer noch eine große Herausforderung, und erfordern gute Planung im Vorfeld. Diese Dissertation beschreibt die wesentlichen Bedingungen um die technische und organisatorische Infrastruktur für solche Projekte zu planen und umzusetzen.

Im ersten Schritt werden grundlegende Systementscheidungen diskutiert, und der Schluß gezogen, daß offene Systeme (mit offenem Quelltext, sowie unter Verwendung offenener Standards und Datenformate) proprietären, geschlossenen Systemen vorzuziehen sind. Dies trifft nicht nur aus technischen Gründen zu: Langlebigkeit digitaler Information (die projektbezogenen Daten selbst, sowie die Ergebnisse der Projektarbeit) ist ein weiterer wesentlicher Gesichtspunkt, der schon in der Planungsphase zu berücksichtigen ist. Auch hier sind plattformneutrales Verhalten der eingesetzten Systeme, sowie gut dokumentierte offene Formate eine gute Voraussetzung für Wiederverwendbarkeit und Langlebigkeit von Information und Wissen.

Auf Basis dieser Erkenntnisse werden die Grundlagen des Informationsmanagements eingeführt. Verschieden strukturierte Daten (hoch-strukturierte Daten, semistrukturierte und unstrukturierte Daten), sowie verschiedene Grade der Abstraktion in den Daten erfordern unterschiedliche Konzepte um sichere Speicherung sowie flexiblen Zugriff für verschiedene Anwendungen zu gewährleisten. Daten-Management Strategien, offene Standards wie XML, SGML sowie Probleme des Datenaustausches und der Daten/Systemintegration werden diskutiert.

Als Konsequenz einer gut geplanten Informations-Infrastruktur (unter besonderer Berücksichtigung Projekt-relevanter Daten) kann flexible Verwendung der gespeicherten Informationen, bspw. zu Publikations-Zwecken für verschiedene Systeme (Web, Print, mobile Systeme, usw.) erfolgen. Auch für die in weiterer Folge vorgestellten Ideen des Wissens-Managements, ist die solide Konzeption der Informations-Infrastruktur von großer Bedeutung. Die Planung sollte sich jedoch nicht nur auf rein technische Aspekte beschränken, sondern auch organisatorische sowie gruppendynamische und psychologische Momente nicht außer Acht lassen.

Schließlich wird im zweiten Teil der Arbeit die Bedeutung der Erfassung und Organisation des *Nichtwissens* hervorgehoben, um auf diesem Weg zu einem reaktiven (Frage/Antwort-basierten) Wissensmanagement Systems zu kommen. Dieses System vermeidet die Schwächen proaktiver Wissenmanagement Systeme, die oftmals unter begrenzter Akzeptanz der Nutzer sowie unter dem Problem der Unklarheit von Auswahl und Tiefe des erfassten Wissens leiden. In weiterer Folge wird auch dargelegt, daß dieses Konzept zusätzlich als System-integrierender Ansatz verstanden werden kann, der es auch weniger gut geschulten Anwendern erlauben sollte komplexe IT Systeme von einem zentralen Punkt her zu bedienen. In diesem Sinn kann das System auch als Informations- oder Wissenspuffer verstanden werden.

Alle vorgeschlagenen Ansätze haben weiters gemeinsam, daß sie nicht nur für die primäre Anwendung in der operativen- oder projektbezogenen Kooperation nutzbringend eingesetzt werden können, sondern weiters meta-Analysen wie *post-mortem* Untersuchungen von Projekten erlauben .

Im letzten Teil der Arbeit werden die am *Institut für Softwaretechnik und interaktive Systeme* durchgeführten Projekte als Beispiel für eine Anwendung der genannten Konzepte vorgestellt. Die Arbeit an diesen Projekten wiederrum ergab Einsichten, die im theoretischen Teil der Dissertation berücksichtigt wurden.

Abstract

Project cooperation, particularly with dislocated scenarios, are a common procedure in recent company- as well as in university research projects. Project monitoring and collaboration under such conditions is still a challenge. Solid planning already *before* the project-start is required. The most important technical and organizational conditions to arrange and execute such projects are described in this thesis.

As a first step, fundamental system decisions are discussed. As a conclusion, open systems (open source, open protocol/format) are considered to be preferable above proprietary/closed systems not only by technical reasons: Longevity of digital information (project data and results of projects) is an important issue that should be taken into consideration from the very beginning of the planning phase. Even here, platform-neutral systems and well documented data formats are a good basis for reusability and longevity of information and knowledge.

Following these findings, the fundamentals of information management are introduced. Differently structured data (highly-structured, semi-structure and unstructured) just as different degrees of abstraction demand individual concepts in terms of organization and management to guarantee secure storage and flexible access for miscellanous applications. Strategies for data management and open standards like XML and SGML as well as the problem of data exchange and system integration are discussed.

As a positive effect of a well planned information infrastructure (with special regard to project related data) a flexible use of the information is easily possible (for example for multi-channel publication including web, print and mobile clients, or exchange issues). For the presented concepts of knowledge management, a solid information management structure is of great importance in the same way. Nevertheless planning should not only focus on technical aspects, but include organizational and psychological factors.

In the second part of the thesis, the importance of *nescience* management will be expressed. This leads to a reactive (question based) knowledge management (KM) system. This system avoids problems of proactive KM, which often lacks user-acceptance and poses problems in deciding what kind of information to acquire. Furthermore the proposed system can be seen as a system integrative effort, allowing even rather unskilled users to handle complex IT infrastructure by providing a single point of access. Hence this KM system is a kind of knowledge/information proxy.

All proposed concepts have in common, that not only a direct usage in the operative or project related business is possible. Also meta-research like a *post-mortem* analysis is feasible.

In the last part of the thesis the projects performed on the *Institute of Software Technology and Interactive Systems (Vienna University of Technology)* are presented. These projects were on the one hand examples for applications of the presented ideas and on the other hand, inspired the theoretical part of the thesis by the lessons learned during project work.

Acknowledgements

This work has been a result of essentially three projects done on the Institute for Software Technology and Interactive Systems (Vienna University of Technology) under the supervision of Prof. A Min Tjoa. Without his continous support this thesis would not have been possible. Also many colleagues on the institute were a great help and very supportive in solving problems in various projects.

I want to thank my parents as well as all my friends for their support during all years of studying.

Additionally parts of the work cited here was done with the great help of other colleagues and students, particularly to be mentioned (in chronological order): Josef Schiefer, Klaus Zelewitz, Artur Bodenstein, Marian Schedenig, Dieter Merkl, Stefan Biffl, Harald Pindel, Alexander Danmeyer, Gerhard Hipfinger, Franz Inselkammer, Horst Kargl.

Introduction

> "The past three centuries of science have been predominantly reductionist, attempting to break complex systems into simple parts, and those parts, in turn, into simpler parts. The reductionist program has been spectacularly successful and will continue to be so. But it has often left a vacuum: How do we use the information gleaned about the parts to build up a theory of the whole? The deep difficulty here lies in the fact that the complex whole might exhibit properties that are not readily explained by understanding the parts. The complex whole, in a completly nonmystical sense, can often exhibit collective properties, 'emergent' features that are lawful in their own right." *Stuart Kauffman [50]*

This PhD thesis is based on work performed on the Institute for Software Technology and Interactive Systems (Vienna University of Technology) between 1999 and 2003. Multiple projects were done during this period and gave significant input for this work.

In the first part of this thesis ("Problem Domain") the specific problems of scientific cooperation and project monitoring in dislocated scenarios, as well as system decision and information/knowledge management issues are introduced. Additionally the most important notions in this thesis are defined.

In the second part ("Solutions and Concepts") the topics: system decision, longevity of digital information, information and knowledge management, project cooperation in dislocated scenarios and unified information access are analyzed in detail.

The third part of the thesis covers a desciption of the projects that were performed on the *Institute of Software Technology and Interactive Systems (Vienna University of Technology)*. On the one hand, these projects can be seen as "proof of concept", of the ideas explained in part 2. On the other hand, the *lessons learned* during project work influenced the concepts in part 2.

A detailed list of (peer-reviewed) scientific publications published during this period, according to this topic are added to the Appendix.

Some Citations from German books are used in this thesis; as those original citation would not be understandable by most non-german speaking readers, I translate them into english, with the author asking for cross-checking if possible. To be precise, the original german citations can be found in the Appendix.

Contents

List of Figures

List of Tables

Part I.

Problem Domain

1. Introduction and Notions to Problems Concerning Scientific Cooperation

"Led by a new paradigm, scientists adopt new instruments and look in new places.", Thomas S. Kuhn

1.1. Introduction

This part of the thesis gives a first introduction of the meaning of scientific (project) cooperation and associated terms that are described and used in this piece of work. Experiences made in different projects are introduced as well as "general definitions" that will be the starting point for the description of the problem domain and for the development of possible solutions and strategies.

I will try to introduce the main topics (the "problem domain") of this PhD thesis, and to oppose the traditional "setup" with current development in information, knowledge and project management in cooperative scientific research.

Remark: The paragraphs marked with *Notion* are not necessarily to be understood as a *mathematical* kind of definition, but more as a definition of how terms are *used* in this thesis, and how they should be understood. This seems to be necessary, as there might be different notations for certain technical terms like *project management*, *groupware*, etc.

1.2. Scientific Cooperation

1.2.1. Basic Ideas

In the last years, all areas of (scientific) cooperation have undergone a rapid change induced by the impact of modern communication facilities like fax, email, world wide web and mobile phone/data connection. As a side effect of this development, the necessity to work on one specific place (because the resources like files, books, co-workers and other things are at this location) vanished in many areas of (scientific) work. This is especially true for software development. Additionally, many successful open source projects like Linux and the Apache software pool [122, 86, 10] demonstrated that work beyond "static" employer-employee relationship can be very productive[1].

[1] I will discuss these new development processes in great detail later.

The consequence of this self-amplifying process is, that those cooperative projects require (software) support in different areas. One main aspect is of course *communication*, especially as the teams are often dislocated. This is not only a result of the described innovations, but also a consequence of recent development of science, that tends to become a *global scale science.* Some areas of "modern" science like the *human genome project* [47], just to name a most popular one, become a heavily dislocated effort. Whereas these efforts are not *by accident* dislocated, they are *necessarily* dislocated, meanwhile often on a global scale. This is the essential point, considering the necessity to support such projects with appropriate IT systems.

> "Examining the record of past research from the vantage of contemporary historiography, the historian of science may be tempted to exclaim that when paradigms change, *the world itself changes with them.* Led by a new paradigm, scientists adopt new instruments and look in new places. Even more important, during revolutions scientists see new and different things when looking with familiar instruments in places they have looked before. *It is rather as if the professional community had been suddenly transported to another planet where familiar objects are seen in a different light and are joined by unfamiliar ones as well.* Of course, nothing of quite that sort does occur: there is no geographical transplantation; outside the laboratory every day affairs usually continue as before. Nevertheless, paradigm changes do cause scientists to see the world of their research-engagement differently. In so far as their only recourse to that world is through what they see and do, we may want to say that after a revolution scientists are responding to a different world." *Thomas S. Kuhn [55]*

This quotation reminds us to the fact, that we do not simply create tools for solving particular problems. Rather we have to be aware of the fact, that those tools can change the problem domain completely as soon as they become a (de-facto) standard, and modify the original problem-situation (see fig. 1.1 for illustration). Moreover this problem/solution combination generated new problems, originally unknown [39], and may create side-effects far beyond the original domain addressed.

The tool to support communication by creating email protocols as well as the world wide web system can be seen as a convincing example. Originally intended to solve some particular and maybe even small scale problems — as soon as the "virus" left its original host, it spread all over the world and re-defined many domains the creators surely did not think of. It defined new ways of working, new ways of living and initiated a broad range of new companies as well as new ways of making business (as can be seen at the examples of EBay and Amazon).

We should keep this idea in mind, when continuing with more details and moving to the solution part (page 55) So in this section, the main aspects of this new kind of cooperation paradigm with its new terms will be introduced.

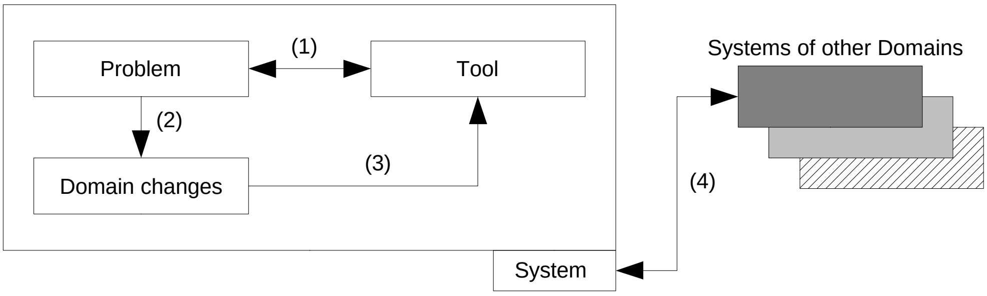


Figure 1.1.: This figure illustrates the (1) problem and it's solving process by using tools. (2) "Unfortunately" the "solving" of problems moves the system to a new level of complexity hence creates a new problem/tool domain [39] (3) This of course creates a feedback to the tools necessary. (4) Coincidently it occurs that the system starts interacting with systems of other domains, that never was originally intended.

1.2.2. Resource Management

One important issue is handling of resources as a key topic in this work. At this place I will give only a brief definition of the meaning of "resource" in this context.

Notion 1 *A resource is a consistent and relevant piece of information, with no specific limit in size or semantic, that usually belongs to a specific task of a project and can be assigned to one ore more persons. There are no principal restrictions to the resource-format, style or content. This piece of information can be accessed as data-stream on some storage device.*

Examples could be: word-processor documents, databases, results of database queries, XML documents, *portable document format* documents, emails, web pages, notes on "post-it" sheets, voice memos and so on. It is clear, that management of resources is a key problem in cooperative efforts, and difficulties increase, when the collaborators are dislocated in space or time. Some key efforts to be dealt with in a project management environment are:

- Exchange of resources: All coworkers should be able to access all resources required for their work.
- Security/Access Control: Different levels of security must be supported; Access control should be integrated with the user/project management and should not pose much additional effort to the user. *Security* and *comfort* are often defacto antagonists, so a harmonic, yet secure solution must be provided. A

well balanced solution has to be set up, because experience shows, that security functions/systems which are very often annoying to the user lead to the fact, that no security at all is used, or other systems are chosen (if possible)[2].

- Version Control is necessary, particularly in a multi-user environment. This could possibly be combined with a check-in/check-out system[3].

- Backup/Restore must be an integral part of each information management infrastructure.

- Longevity of Digital Information [90] should be considered already in resource management. This will be an essential issue in this work.

1.2.3. Communication

Various definitions of communication are existing in different areas of research. A technically basic definition of communication is

> "The *information source* selects a desired *message* out of a set of possible messages [...]. The selected message may consist of written or spoken words, or of pictures, music, etc.
>
> The *transmitter* changes this *message* into the *signal* which is actually sent over the *communication channel* from the *transmitter* to the *receiver*.
>
> [...]
>
> The *receiver* is sort of inverse transmitter, changing the transmitted signal back into a message on the destination." *Warren Weaver (emphasis from the original text) [106]*

and additionally Weaver notes to the effects of communication:

> "But with any reasonably broad definition of conduct, it is clear, that communication either affects conduct or is without any discernible and probable effect at all." *Warren Weaver [106]*

[2] Pretty Good Privacy could be seen as an example. It has well thought ideas, but hardly anyone uses it, because I would assume, that, e.g., the key management facilities are hardly understood by any "normal" user.

[3] This means, that documents can be checked out from the repository by a qualified user. This user might pose modifications on the document. After the changes are done, the document will be checked in again. During a document is checked out, outer users can read the document, but not modify it.

Where Shannon and Weaver are mainly interested in technical aspects of information processing (which is definitely not part of this work, and was widely misunderstood and interpreted as philosophical or metaphysical ideas, which was never intended) those basic notions of communication are required for the understanding of the following text. Nevertheless for the purpose of this work, the definition of communication has to be extended in the following way (where the rest of the terminology above stays valid) and at the same time focused to the desired problem domain:

Notion 2 *Communication is bi-directional[4] information exchange between two or multiple persons or groups. In this work I will distinguish between synchronous communication facilities like personal meetings, telephone, Internet chat; semi-synchronous systems like instant messenger, and a-synchronous communication like email, fax, letter.*

Though communication issues are sometimes thought to be a separate problem not directly connected to project management and information management infrastructure I believe that communication systems have to be closely associated to those systems by many reasons:

Communication systems produce information that is essential for working on projects. Consider discussions via email and discussion forums. Integration of communication information to the project software leverages the traceability of decisions and of the current project status. As users do not want to use different systems for similar problems — an integrated set of tools is consequently required for daily work. [123]

Today information is scattered to multiple applications, documents, databases — in digital as well as "analog" resources. Next generation tools will have to provide a unified and integrated view to dislocated resources and organize them according to the problem domain. In this problem domain, it is desired to have a resource view according to the project/task/user structure. It is possible to give users different views to the same resources following the problem view of the specific user.

1.2.4. Publication and Data Exchange Issues

In contrast to the bi-directional character of communication (see section 1.2.3 on the facing page) I introduce the notion of publication as follows:

Notion 3 *Publication is a one way data exchange, where the receiver is capable of transforming this data into information. One or multiple persons (or systems) publish*

[4] Uni-directional information access is not understood as being communication here. It will be referred as publication issues in this work. Of course in some contexts it is difficult to distinguish between communication and publication (e.g. consider discussion forums attached to online newspaper articles). However, from a practical point of view this definition seems to be useful.

a certain amount of information. Other persons (or systems) receive this publication through a publication channel and consume this information. As soon as there is an interaction between these two groups, a communication is established. So a publication process can form the initial step in a communication process.

Some examples:

- Project reports have to be written during a project to demonstrate the fulfillment of certain objectives.
- As resources are managed in CSCW (computer supported collaborative work) tools, part of these information might be needed for writing project documentation like software manuals.
- Articles or books have to be published (especially in the university context). Often data for these publications will be stored or managed inside those systems.
- As mentioned above: A required necessary publication for every project is a final report that illustrates the results of the work done.

More generally spoken: Data exchange issues are crucial for many projects. Data might be needed in other applications like project management software or statistic packages. The contractor can demand the results of the project in a format accessible outside the used CSCW software, particularly when financial data are addressed too (e.g. in managing the working time of project members). Moreover in some areas it is required to store this kind of information for years (eventually even for decades) to be able to access the information later on.

Talking about Project Supporting tools and *Computer Supported Collaborative Work* (CSCW) usually one aspect is forgotten: As CSCW systems can be seen as a huge resource of information. This particular collected information or parts out of it needs to be published.

Following those objectives, data exchange and publication issues in particular, are not marginal features of CSCW systems, but should be seen as core requirements in selecting or developing such infrastructure software.

1.2.5. Data Security, Archiving and Longevity of Digital Information

Planning the installation of a CSCW or any other information processing system, various aspects of "security" have to be kept in mind. Starting with security in terms of *access control*, reaching to security in the meaning of *data security*, backup and finally *longevity of digital information*.

Notion 4 *Authentication is a process, that verifies that a person (or another system) that tries to retrieve information from the system has the appropriate access rights. Certain levels of authentication are possible, e.g.: From public access to read-only, write-only, full access, etc.*

A usual way of authentication is the use of a login/password combination. This should guarantee, that different users are distinguishable in interaction with the system. Ofter mechanisms are also common like smart cards for user identification, and additional passwords for security purpose (e.g. credit cards)

Notion 5 *Encryption is a process, that modifies information in a way, that reconstruction is only possible under certain circumstances. Those include that only a specific party with specific knowledge is able to reconstruct the original information.*

Also in encryption, passwords are used to secure encryption systems and generated keys (in public key systems) that have to be kept secret and are used in combination with the password to encrypt or decrypt information.

Other issues in data security are the aspects of archiving and longevity of digital information.

Notion 6 *Backup is a process, that extracts all relevant information out of a running information processing system and stores this in a secure way. The backup has to be performed in a way, that a restoration of the complete information in the system is possible from the archive in the context of the same infrastructure that was available at backup time.*

Secure has multiple meanings in this definition: The archived information has to be stored on a medium, that is stable enough to keep the information physically secure for the desired storage time. The second aspect is, that the storage medium has to be kept in a secure place, ideally not near the data processing system.

Another aspect of data security is the problem often discussed under with the term *longevity of digital information*. Unfortunately the importance of this issue is widely underestimated. The problem has three main aspects:

- Data Formats: Are used data formats open or closed, well documented or not? (Details will be discussed in chapter 6 and 8)

- Storage hardware and media.

- Ontology and application context

These topics will be discussed in detail in chapter 7. To conclude with the definition of archiving: Whereas backup is a security measurement to ensure that a running system can be re-set into a specific status (e.g., in the case of a system crash, hardware or software failure), archiving means storage of data that allows reuse outside the original system context:

Notion 7 *Archiving is a process, that extracts all relevant information out of a running information processing system and stores this in a secure way. The archiving has to be performed in a way, that a restoration of the complete information in the system is possible from the archive even when the original infrastructure is no longer available.*

1.3. Additional Requirement: Project Management in Dislocated Scenarios

Besides the cooperation problems described before, special considerations have to be done, when project teams or certain project members are dislocated. As from now, such projects are named *dislocated projects* or *dislocated scenarios.*

Notion 8 *Project Management in "dislocated scenarios" also written as "dislocated projects" means, that one or a small number of project managers has the role to manage a group of co-workers dislocated by geographical, cultural or other reasons. Usually there should be one main-project manager, or if this is not possible, the project could be divided into highly autonomous parts, with one manager responsible for each part.*

In addition we will have to deal with aspects like the following:

- Project Monitoring: when team members are dislocated, it can be a hard issue for other members or the project manager to stay informed about the status of the work.

- When colleagues are located in different countries, one could have to deal with language issues, as reports have to be written in different languages, some members might not be able to use English software and so on.

- Communication is again harder when team members could not meet regularly. Not only the communication itself might be a problem, also the traceability of communication processes and decisions.

- As colleagues are often not well known with each other (sometimes they do not know themselves at all), this poses additional problems to collaboration, as electronic communication lacks the eye-to-eye communication and misunderstandings might occur more often. This is particularly true, when different cultures are involved.

Remark: A careful planning as described in this thesis, is obviously also suited to enhance the quality of traditional projects.

1.4. A First View on Communication and Groupware Platforms

Notion 9 *The term groupware will be used for software packages that support collaboration by offering communication tools (mailing, discussion, online, offline), resources and user management with authentication. This software is usually implemented in a client/server architecture, though this is not a condition, also other architectures like peer to peer networks are possible.*

In the area of Groupware, there are a set of well-known commercial products available. The best known products are Lotus Domino [62] and Microsoft Exchange [67]. Both systems are client/server based: the server is called Domino[5] (Lotus Notes) and Exchange (Microsoft), both systems allow at a certain level clustering and upscaling. Especially the Domino system is well known to be able serving tens of thousand users. On the one hand this is possible by special cluster functions, on the other hand, the Domino server is available for different operating systems also for high-end servers like IBM AIX Unix, OS/400, z-series mainframes, Solaris and Linux. The Exchange server is only available for Microsoft server systems.

On the client side, both worlds offer special clients: Lotus Notes on the one side, Microsoft Outlook/Exchange on the other. Also a web-front end is available for both systems, though not the complete functionality is available when using the webbrowser as front end application. Unfortunately the special clients are only available for Microsoft operating systems, and to a certain extent for Mac OS. This is also the case for Lotus Notes, though there seem to be preparations for supporting Linux in future versions.

[5] In fact, the Domino product line consist of a wide range of different products. Starting with the "normal" mailing and groupware database system ending with document management and e-learning servers. In addition Lotus Notes/Domino has state of the art synchronization features between different instances of the same Domino database. Even if a database is part-time offline (like on notebooks), Lotus Notes does synchronization on the dataset field level. This is an extraordinary feature, however, it seems no to be so important for the issues discussed here. More important is the fact, that Notes/Domino supports workflow management at a certain level.

There are multiple reasons, why those systems are not considered in this work as a basis for a CSCW development:

- This work is focused on *open systems* and *universal access*. This is not the case for both commercial products.
- In case of collaboration in scientific context and *across system borders* it is not easy to assume special clients to be installed on all systems, even if they would be available for all systems.
- As users in scientific and also in many companies tend to be very mobile, a flexible access through different types of clients is desirable (also including mobile clients).
- Limited functionality in terms of project management and organization (at least in the base systems).

Additional considerations can be found in chapter 6 on page 57 (Systems) and chapter 7 on page 87 (Longevity of Digital Information).

The consequence of this situation is, that an open (source, protocol) and web-based system that supports different clients (like cell phones and PDA[6]) is considered.

There are some other groupware systems available like BSCW [20], which is a web-based system, but it has many functional limitations and is not open as well. I suggest a concept that fits more naturally to the daily work, hence organizes data in a project/task specific way.

1.5. Computer Supported Collaborative Work

A part of the problems and solutions mentioned here are also addressed in the field of *Computer Supported Collaborative Work* (CSCW) research. However, CSCW research often does not focus to project management and monitoring aspects as crucial parts of the problem. Some CSCW research topics do focus to particular processes [14] and try to support a specific work process.

Notion 10 *CSCW — Computer Supported Collaborative Work will be used somewhat similar to the term groupware. The main difference is the notation, that CSCW is mainly understood as a research topic, whereas groupware is understood as a product category.*

In this thesis the CSCW aspects are organized in a more natural way for day to day project based work, namely according to the structure projects are usually organized

[6] Portable Digital Assistant: like Palm or Sony Clie systems or Microsoft Windows CE devices.

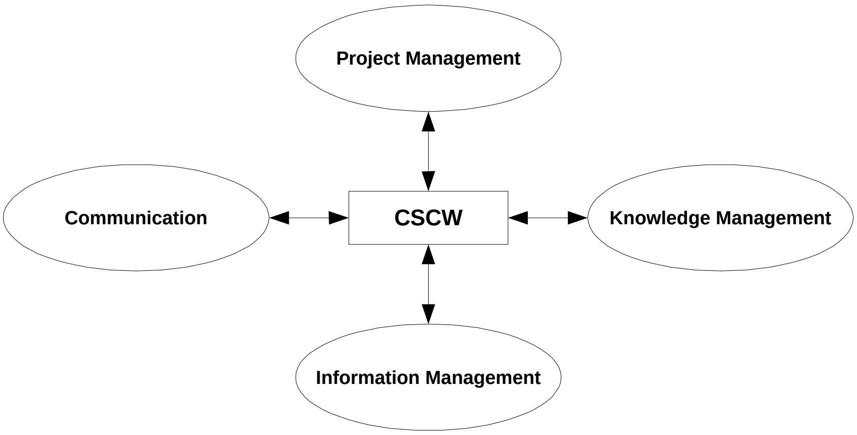


Figure 1.2.: Integrative CSCW Concept: Knowledge-, Information- and Project-Management as well as parts of Communication Issues are often simply the same problem seen from a different perspective.

in: Projects, Tasks, Resources, Todo lists, collaborators on specific tasks... But also other views to this topic are possible (see also Fig. 1.2). I will describe fields, where the most important are: Information Management, Project Management, Knowledge Management and Communication Issues. Each of these fields have specific research disciplines that try to focus to particular parts of the problem and often also tend to differentiate from the other problem fields. I believe, that an integrative view is more useful, as many aspects of these topics seem to be very similar, just seen from different sides.

So I do hope this work will show, that the CSCW concepts proposed here are not *yet another xyz-buzzword* which tries to outperform all other concepts, but show much more that contemporary CSCW research should try to integrate those different concepts and give the user the opportunity to select the *viewpoint* useful for his or her work.

I will also show, that besides many technical problems that arise on trying to provide an CSCW strategy there are also significant subtle problems to consider:

> "Yet social systems are very resistant to change and have an enormous ability to tolerate, rather than solve, problems. The path of least resistance — simply allowing problems to fester — all to often ends up pulling down even the greatest of societies. [...] The issue is always how a society with festering problems can force itself to act before there is a crisis that may take the system down with it." *Lester Thurow [118]*

Obviously what is true for a society is also true for smaller social systems like companies or institutes or project teams. Overseeing this very important factor mentioned in the citation above often is the main reason, why new IT infrastructures become *stranded investments*. Hence it is clear, that the implementation of a CSCW solution may not be limited in technical planning and engineering, it has to deal intensively with the issue mentioned above. Besides others, there are two main precautions to be taken: (1) the system has to offer the user a concrete advantage over other older systems or strategies and (2) the system implementation has to be fully supported by *all* management hierarchies[7], and sometimes even slight pressure from top down might be required so that the usage of the CSCW systems become the needed critical mass. Equally important is the tight integration of the system administrators that will be responsible for the technical management of the system, to ensure a smooth integration into the running IT infrastructure.

So details about the project related implementation of CSCW are described in the ideas of *Open Science Workplace* see chapter 13 on page 185.

1.6. Open Source and Open Protocol

Open Source and Open Protocol software/specifications are an important factor in cooperative work, especially when work is done "across system boundaries". This means e.g. when different universities or companies are collaborating. The reasons are:

- Software is easily available for all partners (without the need of license fees which is particularly important in university context)
- System communication and data is open and documented, which allows better integration into existing systems.
- The software is extensible and transparent, and so is the data generated by the system (if the system is based on open standards)

These are important factors and will be analyzed in more details in section 5 on page 51 and in the second part of this work.

[7] The emphasis on *all* is crucial as experience shows, that sometimes big system changes are performed without enough communication with middle or low hierarchical positions. The effect often is resistance of those people ignored during the planning phase with all negative consequences!

2. Experiences and Examples

> "Every good work of software starts by scratching a developer's personal itch." *Eric Steven Raymond [86]*

2.1. Introduction

This thesis is based on a set of projects performed at the *Institute of Software Technology and Interactive Systems (Vienna University of Technology)*, experiences in commercial projects and the evaluation of scientific literature as well as discussions on scientific congresses.

I will give a brief introduction to the projects mentioned here; A more detailed inspect will be given later in the second part of this work, when the specific problems are discussed (see part II).

2.2. German Literature and Language Related Projects

Between 1999 and 2002 two projects for German literature and language science at University of Salzburg were performed. The first project dealt with *Austrian literature round 1900.* The first version of a content management system was developed, that allowed the German literature and language scientists to manage the information for generation of an open distance learning website. Detailed informations about this project were published[1] at ED-Media conference 2000 in Montreal [101].

The second project exceeded the original limited functionality and included e.g. interactive elements. Moreover, as about 5 persons were involved in the first project, the second project increased to a size of about 20 scientists. Again those colleagues were working at different locations (Vienna, Graz, Salzburg, Slovenia, . . .). This increased the complexity in terms of managing the project and the resources associated with the project like reports, multimedia elements, texts. More details about the second project can be found at [98, 99]

Besides the technical aspects of those projects, lessona were learned in terms of cooperation between colleagues, that come from very different "cultures". In this

[1] All mentioned papers in this section can be found in the appendix, see V on page 199.

case from different domains. In case of the *Open Science Workplace* “real” culture differences had to be overcome (see section 2.3).

Additionally we found, that a clear documentation is useful to keep all project members (as well as the ministry of science, that financed the project) up-to-date with the current (development) status, this documentation was done manually by creating web-pages.

2.3. Open Science Workplace — Cooperation with Iran

We learned from the situation described in the section above [97] and started a new project to support (dislocated) project teams in terms of project monitoring, resource management, communication. This system is developed in two projects starting in 2001 and ending in 2003 in a collaborative effort between the *University of Kerman (Iran)* and the *Institute of Software Technology and Interactive Systems (Vienna University of Technology).*

The findings and experiences of those projects as well as the concrete results will be a relevant part of this work and will be described in detail later. Results were also published on scientific conferences. Those papers can be found in the Appendix.

2.4. Open Source Commitment

I believe, that there are many reasons, why it is essential to avoid building basic closed source infrastructure on top of proprietary software, especially when this software is a product of a monopoly. Open source and again more open protocols are a guarantee for transparent and flexible infrastructure and minimal dependency of a specific vendor. This is a relevant argument especially in governmental applications (e-government) and all applications where security and transparency are an issue.

Besides “political” arguments also the cost factor have to be considered. This is especially the case in educational institutions like schools and universities. As on the university level cooperation with partners in different nations are usual, open source software leverages the use in such projects as every project partner easily can access and use this software on different systems.

As a logical consequence all software produced during my work on university is based on open source software and is delivered itself under open source license. Moreover several activities in the open source community (e.g. Apache project, Enhydra project) took place. As an example, an article about the Apache Software Pool in the leading German professional information technology journal was published [94].

As a matter of fact, the situation of open source and open protocol development as well as the consequences to society and politics are an essential part in this thesis.

3. Project Management

3.1. Definitions

First of all, three terms have to be introduced and defined, as they will be used in different chapters throughout this thesis:

Notion 11 *A project is an undertaking with a clear specified goal, time- and cost-frame with more than one person (usually from different special fields) involved, located outside the regular "operative business".*

Notion 12 *A task is a work-package (a part) of a project.*

Notion 13 *Project monitoring, is the passive monitoring of critical parameters of a running project by one or more representative persons who are responsible for different areas of the project.*

3.2. Special Aspects of Dislocation Scenarios

The (scientific) teams mentioned above should be supported with new software tools and strategies that support their specific problems. Communication is an essential topic, but more than this: Project management is a severe job already in traditional projects. In dislocated projects, maybe even with different languages involved, it becomes even more difficult to manage upcoming problems. Much more: to keep the overview over the projects tasks as well as the persons and resources involved. Handling of different resources like documents, project documentation "official" documents, databases and the connection between different versions of those resources and relations to coworkers is a huge challenge.

Efficient tools that support project monitoring as defined above, can reduce the risks of running projects. Briefly, critical parameters of a project are:

- Partitioning of the projects work to tasks and subtasks and modifications of this partitioning during the project runtime
- Associations between tasks and persons working on tasks, and changes in those relationships

- Associations between tasks (dependencies), and detection of (time) critical dependencies
- Progress and quality of tasks.
- Communication structure: Overview about all persons involved (and their contact information), Possibility to address all coworkers, logical groups, ...
- Management of all relevant resources
- Publication facilities for "external" audience, this means: scientific publication, books, web-publishing and the like.

The success of a project can depend on clear strategies on how to handle those problems, and it becomes more critical with the increasing number of people involved. Concepts that support project cooperation should consider the items above. In this thesis such concepts are discussed and a solution is proposed.

3.3. Acceptance of Technology

In starting new projects the project manager has to decide what kind of technology support and project monitoring strategies he or she will choose. An integral part of this decision has to be the question whether the selected technology will be accepted by the co-workers [138, 42]. It is not trivial to design software and systems, that support teams as mentioned in section 3.2, and are accepted by the project members. Certain aspects should be considered:

- Designs that require special client software installation (as mentioned in section 1.4 on page 31 can cause problems).
- Software that users are already using should be supported (Web-browser; Office packages e.g. by using WebDAV protocols, . . .)
- Tools should be as highly integrated as possible. Users do not like to use different applications for similar tasks. What seems to *belong together*, should be *integrated* under one user interface, see also chapter 11. (This includes communication issues and project management tasks.)
- The approach should be standard compliant, easy to install and to extend. These might be aspects less relevant for "normal" users, but they are the more critical ones for managers and administrators.

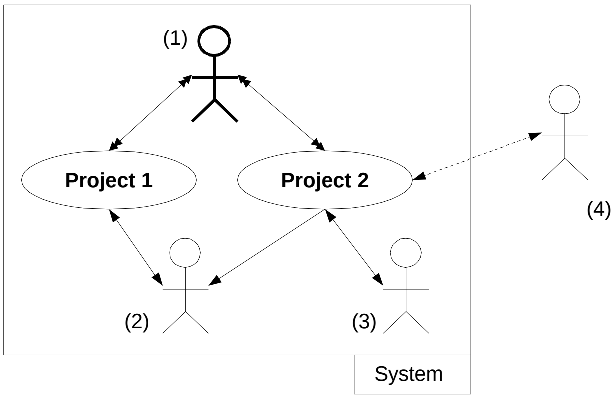


Figure 3.1.: (1) A system/project administrator for each project exists. (2) This user is member of project 1 (read and write access) and monitor of project 2 (only read access). User (3) is member of project 2. User (4) is a public user. Specific rules have to be defined which information should be accessible for this user role.

Eventually one has to accept, that success is nothing we can plan. I believe, that severe mistakes in design can be avoided, but as different examples show (WAP, UML, ...) that even good planing is not a guarantee for success. Finally I suggest different attempts, analyze concepts and make implementations, the user has to decide the ones which are valuable for daily work.

3.4. Roles in Projects

In project management and CSCW scenarios, it is necessary to define roles for the users. Of course different types of classifications are possible, e.g. depending on the structure of the tool. In some CSCW applications, process oriented structures are suggested [14]. Other systems like Lotus Notes define access control mechanisms and depending on the functionality of a user, certain access rights are granted.

Following my experiences I will suggest to start with defining the roles according to a project/task/work and the appropriate communication structure. After those roles are found, it should be easily possible to define access control rules, for each role. However, one has to decide under what circumstances projects are usually performed. For our *Open Science Workplace* implementation, we assumed a “friendly policy”, meaning that collaborators on a project should be interested in cooperation not in hiding information from each other. In fact under other conditions one might want to assure stricter rules.

Hence I decide to keep the number of roles small. If too many roles are introduced, the consequence is, that the project members tend to be confused about how to use a system with too many slightly different role definitions. From the practical experience I extract the following roles (illustrated in Fig. 3.1):

- The *Project Manager* is one person, who is responsible for the complete project.
- The *Project Member* is a person, that works on at least one task of the project.
- The *Project Monitor* is a person, that has read-only access to the project information and is perhaps the manager of the company, institute head, ...
- The *Public User* is "the rest", meaning all persons who have technically access to the project management information system, but do not incorporate one of the other roles.

The concepts and implementation suggested in the next parts, will follow those first suggestions.

3.5. Structure of a Project

It is clear, that an approach, that tries to support and organize project work has to do some structure assumptions about how a project works and how it is organized. In this thesis the following aspects of project structures and project types are taken into consideration:

- Projects, that include users from different "domains" like artists and technicians require a kind of "mediation" between the groups involved.
- A decision has to be taken if open systems are preferred, as well as the question if support is available for the system.
- CSCW software can be structured by different means: task oriented, resource oriented, communication driven, process oriented, ... A consistent system will be suggested.
- Collaboration can also be distinguished by the type of project. E.g. software projects, art-projects; scientific collaboration and business projects and so on. Differences should be discussed and common properties extracted.
- Additionally work-flows have to be taken into consideration in some (usually formally organized) projects.
- Also the specific features discussed in the sections before and the influence to a CSCW concept have to be analyzed (distributed-, mulit-language projects)

4. Information Structure

4.1. Representation of Knowledge and Information

4.1.1. Data

As the terms data, information, and knowledge are used many times in this thesis, a definition of the meaning seems to be appropriate. The definitions used here are inspired by the use of the terms by H. Willke [129], whereas the notion "data" is the most difficult one, as many common definitions are circular in so far, as the term *information* is used in the definition of *data*. This is avoided in the notion provided here:

Notion 14 *Data is a documentation or a record of properties of arbitrary entities, such as human generated artifacts or results of measurements, not bound to any specific system. Accuracy, precision, semantics and so forth are not relevant criteria for the term data.*

So every property in an arbitrary system that is documented is data. It is completely unimportant whether this data is relevant for anything or anybody. This is the matter of the second step:

4.1.2. Information

Notion 15 *Data becomes Information as soon as it is found to be a relevant by any operational system.*

Hence data is any documented property of entities, whereas information is any *relevant* detected and documented property of entities. As there has to be a *criterion* for deciding whether a difference is relevant or not (there is no relevance *a priori*), every information is *system-dependent* and *system-relative*. Moreover (as it will be analyzed in the next section) this information has, according to the system-dependence, a specific structure. Information with *no structure* can not be understood as information following the definitions above. It is data; because as soon as some criterion is available that "pushes" data to information, it seems to be clear that some amount of structure is a consequence too[1].

[1] Nevertheless the term *unstructured data* is often used describing information structure, also in this thesis. In fact *unstructured* means *unstructured from the viewpoint of an information processing*

Nevertheless I want to remark here, that there are more critical positions about the definition and usage of the term “information”, e.g., by Joseph Weizenbaum [128] or Barwise et.al. [29]. Weizenbaum argues, that there is no information outside explicit human context. E.g. there is no information inside the telephone book of New York. This is simply data; this data is transformed to information if a human reader gets information out of this book and — having a hypothesis (!) — for example about the use of this data. So the point is, that data becomes information if a (human) receiver processes the data and uses a hypothesis about the meaning of the data.

I am not sure, whether this critics goes to far. In fact, the arguments are impelling, though other perspectives are possible too: A book or a computer makes no sense in a non-human related context. These machines or devices are built from humans for humans. This is true even if no information is stored in the system: e.g. if the hard disk is formatted and the book pages are empty. A non-human user would most probably have no idea what those artifacts are for.

So if we discuss now about the data stored and processed inside those artifacts or machines, it seems not to make too much sense, if the discussion is done from a non-human viewpoint. I think Weizenbaum's critic might be strictly spoke true, but I believe from a pragmatic point of view it might be more useful to think of information as soon as data is *intended* and stored for a *specific purpose* in a *specific cultural context* on a *fitting medium*. This is obviously true for the telephone book of New York which is available in New York, not in Vienna, which is intended to be used by Americans and which is printed on paper in a way, Americans are used to read and understand. Consequently I assume, that the usual usage of the word information for e.g. the data that is stored inside a telephone book seems to be appropriate.

However an interesting point can be extracted from this example: *if* this mentioned telephone book would be given to a human, who is grown up and educated in a completly different cultural context, what would be the meaning of it then? This example shows the problem on a broader scale then Weizenbaum discusses it. In fact we might argue, in the latter case, this book might not contain information for the imagined cultural context; in the worst case, not even the artifact *book* would be recognized as such.

On the other hand, Barwise et.al. argue, that definition of information (especially of transmission of information) usually is a circular one:

> “Most existing treatments simply assume that the transmission system (including the people or sources at each end of the transmission and the entire surrounding environment) all works correctly. Not only does this lead to an explanation that is extremely brittle — the moment anything goes wrong, the theory completely disintegrates — it is also circular since ‘all

system. In other words: the software whatsoever has problems to detect or process the structure of the “unstructured” information.

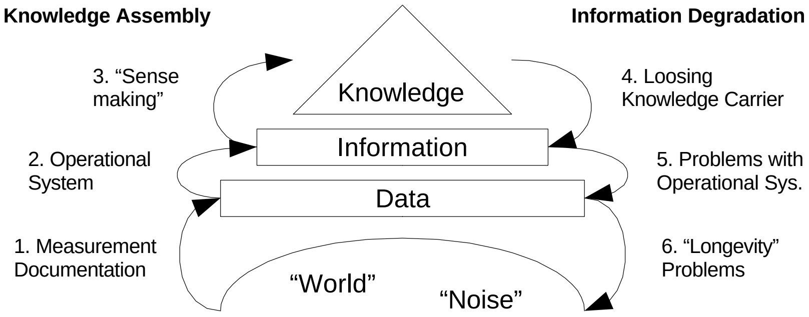


Figure 4.1.: Knowledge Assembly and Information Degradation as counterparts in the data-information-knowledge processing workflow.

> working correctly' boils down to 'does transmit information as intended.' "
> *Keith Devlin [29]*

The reason, why those two points of critic are outlined here is to demonstrate, that the term information is highly discussed in multiple fields of science. Discussion and definition ranges from very technical ones (like the information definition of Shannon and Weaver) over ontological discussions (Weizenbaum) to a mathematical problem (Barwise et.al.).

So to conclude: It might seem ignorant facing the citations above, but from the point of view of this thesis a very pragmatic definition of information is sufficient. *Pragmatic* means that users are the ones to decide the difference between data and information. This is not sufficient for an ontological discussion, but good enough for this particular work. Hence the term information is used here as in the definition above. Whereas the "operational system" includes different cultures as well as the question who is the intended audience for a particular information[2].

The definition of knowledge is only the third step:

4.1.3. Knowledge

Notion 16 *Knowledge is the result of an intentional "sense making" operation.*

This *intentional sense making operation* means linking or including the "new" information into any relevant context of experience and already available knowledge and in that generating a new level of complexity. Moreover, as it has an *intention*, it means,

[2] One could probably even see a maschine that is able to interpret data, as an operational system that creates information out of data.

that the "new knowledge" increases the value of the knowledge available, e.g. because the intention is some pragmatic use.

So to conclude: knowledge is bound to information, which is depending on data, which is depending on any kind of measurement or detection process. "Measurement" or "detection" are a technical synonym for an empiric process. Hence the consequence is, that *knowledge is always a based on an empiric process.* This process of knowledge assembly and information degradation is visualized in figure 4.1. It is obvious, that the left part of the figure, the knowledge assemble is a desired process, and the right part, the information degradation is a negative effect. The main topic of this thesis will be the analysis as well as the suggestion of concepts that should strengthen the left side while limiting the effects of the right side of this illustration.

Another interesting observation can be done, following the definitions in this section: the process of knowledge assembly is in most cases a *positive feedback loop.* The first step from data to information needs an operational system, that is capable to do this "processing". For this "processing step", the operational system again needs knowledge "assembled" earlier.. So it becomes clear, why a good information and knowledge management strategy that includes the building of decision capability and problem solving (usually done by employees) is so important as a factor for efficiency and flexibility of the company.

4.2. Structure and Organization of Information

4.2.1. Highly Structured Information

In processing information it is an essential question to what level the information is structured or at least *can be* structured by a system or a person who is aware of the semantic meaning of the information. From a "historical viewpoint" information processing started with highly structured information. It is difficult to give a unique and precise definition of the term structured data, as it might be used in different ways. So I suggest a pragmatic definition of what is understood by structured data in this thesis:

Notion 17 *Data is called (highly) structured if it is (easily) possible to give a (1) precise and (2) persistent definition (schema) of this data and if (3) the granularity of the data is high.*

It is essential to emphasize the part (2) of the definition. Even Graph-like structures can be seen as highly structured, as long as the schema is persistent and not very flexible or often changing. Highly structured data is mostly stored in databased following the relational model [24], but also other data management strategies are possible as XML or SGML based formats (as an example). Also part (3) of the definition might need some additional comment or clarifying example: It is possible to store every kind of digital information in a highly structured way. E.g. one could store the binary information of, say photos of 100 cars in one *binary large object* database field. The granularity of this kind of storage would be very low in this case. On the other hand, if a system or a person would have analyzed those 100 photos in advance, extracted informations like: car type, color, price and so on and would store this information in the described high granularity, the data could be understood as highly structured. Unnecessary to mention, that the value of the latter information structure might be far higher than the unstructured storage, as the information management system is able to provide a lot of tools to work, filter, query and restructure the data.

As a matter of fact there are cases, where it is difficult to decide whether data is highly-structured or semi-structured:

> "Indeed, the same pice of information may be viewed as unstructured at some early processing stage, but later become very structured after some analysis has been performed." *Serge Abiteboul [3]*

4.2.2. Semi-Structured Information

Semi-structured information has a certain amount of structure, but either has a lower granularity than highly structured data or is organized in a different manner. As already mentioned in the definition above, semi-structured information can be seen as graph-like information [22, 124]. To put it together into a definition:

Notion 18 *Semi-structured information is information that can be viewed as a graph-like, often document-oriented structure with high to medium granularity.*

From a historical point of view, the relevance of processing semi-structured information increased dramatically in the last decade my numerous factors: First of all the rise of the World Wide Web generated a hugh amount of semi-structured documents (written in hypertext markup language). This generated the need to organize this kind of information and to bring it to some meta-structure for making access easier (search engines, . . .).

A second important factor is the so called: paper-less office. In fact, people print more documents then ever, but the reason is, that most people do not like to read larger documents on the screen. Many companies started to archive and manage documents only in digital form. This generated hugh amounts of semi-structured content (e.g. documents with metadata).

These and other recent development increased the need to develop powerful systems to handle those semi-structured documents. Also in project-cooperation a lot of semi structured information is created (like project reports, emails, todo lists, notes, . . .).

So organizing and managing this kind of information is an important task for any project supporting CSCW tool.

Semi-Structured information consists of data and metadata[3]. Metadata has the functionality to structure the information and to add additional *information about the information*. Most important in this field of semi-structured information management is the markup language XML (extensible markup language) [133] and many more specific languages and frameworks expressed in XML (like the resource description framework RDF or the semantic web [13]). Using the data and metadata, which can be nested, it is possible to see build a tree-like structure (graph).

In some publications like in [22] semi-structured and unstructured data/information is used as a synonym. In my opinion, this is definitely not a good idea, as there is "real" unstructured information, and using those two terms equally will generate some confusion. Consequently in this thesis the terms *structured*, *semi-structured* and *unstructured* have different meanings and are distinguished clearly.

4.2.3. "Unstructured" Information

I use the term *unstructured information* different from semi-structured information.

Notion 19 *Unstructured Information has no machine-understandable structure or if so, is of very low granularity.*

To put it in a more pragmatic way: all information that is not highly structured or semi-structured by the use of any machine-readable metadata is considered to be unstructured data. It is important to note the term "machine-readable" or "machine-understandable": A Word, PDF document can be unstructured information for a particular system, if either no metadata is provided in the document or if the system is not able to analyze/read the document.

4.2.4. Alternative Approaches

Usually the structure of data is described as mentioned above. However, the terms *structured*, *semi-structured* and *unstructured* may be confusing in some context. Interestingly in knowledge-management research another terminology is sometimes used to describe the structure of data [110]. This suggestion is reprinted in table 4.1.

I believe, that this kind of definition is very useful too, and can be mapped to the more common definitions above. So the elements in the table can also be seen as an

[3] At least it *should*. A very fundamental problem is the so called ontology of metadata. Is the meta-information understandable for everyone? For every processing system? Is it described well, or did the information manager use abbreviations only understood be herself like <i> instead of <image>, and so on. In section 8.3 on page 115 attempts to solve this ontological issues are discussed.

Degree	**Model**	**Interface**	**Example**
Thoroughly Formal	Relational	Form Interface	Database Interface
Formal	Content-structured document	Tight XML Structure	XML-EDI
Partially Formal	Document template	Loose XML Structure	Investment recommendation template
Informal	Free text	No predefined structure	ASCII text file

Table 4.1.: Degrees of formal and informal knowledge (from [110])

introduction to my definitions, where *thoroughly formal* information is *highly structured*, *formal and partially formal* are differentiations of *semi-structured* and *informal* means *unstructured* information.

4.3. Availability

Availability of information is an essential topic in project cooperation. There are multiple steps in terms of availability:

1. Is the information available in digital form or only in some “analog” version like as fax, letter, contract, “post-it” note . . . ?

2. *If* the information is existing in digital form: is it possible to manage it with the (CSCW) software? E.g. is a document repository available? Are all necessary documents in it? Is it possible to include all relevant information like emails, large documents, databases. . . , even when they are produced and handled by a different software system (like an email client or database management system)?

3. *If* all required information is in the system, or may be accessed through the systems user interface: is it available on different locations? Without using special software installed (access through browser)? Accessible with mobile clients (cell phone, PDA. . .)? What are the restrictions?

4. Information (especially in our kind of projects) can be dislocated[4]. How about management of dislocated information?

[4] *Dislocated information* means, that (1) information is generated at different locations and (2) is probably also stored at different locations e.g. in local database systems.

5. If all problems above are "solved", is the access read-only, or is there also write access. If so, how is synchronization handled (if necessary), e.g. when systems like PDA's are involved, that are not always "on-line"?

These are the main topics that have to be addressed by a CSCW software system. In Part 2 a detailed discussion about those problems is done and solutions are proposed. *Unified Information Access* is a goal to be reached in the future, and means, that the end-user is not distracted by a multitude of information processing systems and applications, but that those systems are running invisible in the background, managed and administrated by a dedicated expert, and the end-user has one uniform access application/system like a web-browser or web-desktop, or web-mobile-client that offers precisely the functionality he or she needs for the work to be done. Such a system is outlined in Part 2 in chapter 11

4.4. Reusability

Reusability of information and knowledge is an essential issue in project management, cooperation and communication. I differentiate two kinds of reusability: *operational reusability* and *semantic reusability.*

Notion 20 *Operational reusability means, that the information stored in a specific system at a specific point of time can be restored without loss of information (1) at another place and (2) at another time.*

Notion 21 *Semantic reusability means, that information, once added to a system under a specific semantic meaning (context), might be useful (re-used) in a completely different domain.*

To make those definitions clear: An example for *operational reusability* would be a typical *backup/restore* process: is it possible to make a backup of the currently system state und restore it completely (including re-installation of all software parts)? Is it possible to do this restore even much later in the future (e.g.: 15 years later)? This would be a typical *archival* problem. Those terms have already been defined earlier (see section 1.2.4).

An example for *semantic reusability* could be the problem, that information that is stored in the system like source code documentation, reports and the like need to be recombined to form a complete project documentation or final project report. Or the typical problem in e-learning environments: if information is added into the specific e-learning software, is it possible to reuse this information (text, multimedia, . . .) in some other learning environment like another software product or book and so on. More generally spoken: many kinds of publication tasks are closely related to information reusability containing references, citations, multimedia content, textual information and much more. This publication finally might become a source of reuse in later publications again.

Both types of reusability scenarios open a wide area of problems and pitfalls which will be analyzed in details in the second part of the thesis.

4.5. Meta-Levels

It turns out, that appropriate data and information storage and management strategies are just the first step in contemporary IT infrastructures and project/business scenarios. If well designed, it is possible to reuse information stored on the next level of complexity using *data-mining*, *data-warehouse* and *knowledge management* strategies. This goes beyond “simple” reuse of information in different contexts. Those strategies are of greatest importance as the amount of information and data stored, e.g., in company databases increases dramatically and it becomes possible to draw conclusions out of these databases far beyond the original meaning.

Notion 22 *Data warehouse and data mining strategies try to recombine and reuse data stored in various locations to retrieve information and knowledge beyond the original scope of the databanks.*

Similarly knowledge management strategies profit dramatically from well a designed information management [71].

Notion 23 *Knowledge management tries to capture explicit as well as tacit knowledge relevant for the operative business of an institution. This captured knowledge should be accessible by anyone (inside the system) to solve problems in the daily work.*

5. Proprietary versus Open Systems

5.1. Open/Closed...

> "Now that we see both open source and open standards in the same light, it is appropriate to examine the differences between them to determine how the values of each can improve the other." *Ken Krechmer [53]*

Open Source and Open Protocol development are a core topic of this thesis. As these terms are not always used in a concise manner, a definition is necessary too.

Notion 24 *Open Source software development is software development that produces products where binary versions as well as complete sources of the product are available (usually on the Internet). Moreover open source software is published under a license, that allows any user to do arbitrary modifications to the sources and create new versions of the software[1].*

The public reception is used to the term *open source*, but *open protocol* is not so widely known. In fact protocols might have a far larger impact to IT landscape than open source software. Hence this is an important topic in this thesis and in the software development described here.

Notion 25 *Protocols are specifications how different (technical) systems communicate with each other. The specification of such a protocol might be closed/proprietary, so that the control over the protocol as well as the use will be limited by the owner of the specification. Open Protocols are such, that are specified and published in public, hence everyone is able to use these specifications in his own products. Usually specifications are published on the Internet.*

An example of a site dealing with important open protocols is the website of the W3C — *World Wide Web Consortium*[125] that hosts a large amount of web-related protocol specifications like XML, RDF ... The practical importance of protocols in general and open protocols in particular is enormous. Details will be discussed on different parts of this thesis.

[1] Examples are Gnu Public License (GPL)[60] and Apache License [59]. There are minor differences, especially in the question whether commercial use of this software is allowed and under which conditions.

5.2. Proprietary Systems

The importance of open protocols and formats has been mentioned earlier. On this basis, the definition of a proprietary system is:

Notion 26 *A proprietary system is a system, that is essentially based on closed protocols and closed or not well-documented data-formats.*

Considering proprietary systems for cooperative work the same questions have to posed as mentioned previously:

- Is the software accessible and usable for all project partners (language, price, support, . . .)
- Is the persistence of the project-data guaranteed (even for longer periods): is the document format open and well documented, are open standards used and so on?
- If necessary: are customization and modifications allowed, and if so, under which circumstances and under which conditions?
- Is the software company solid? Will the software be supported in the future? Is the development of the software consistent or are e.g. data formats changed regularly to create incompatibilities between versions. . .

Part II.

Solutions and Concepts

Mission

In the first part of this thesis a brief overview about the problems in scientific project cooperation — particularly in distributed projects — was given as well as a definition of the most important terms used in this text. Part 2 will cover a detailed analysis of those problems as well as suggestion for solutions:

The basic message of the thesis boils down to this: How is it possible to cooperate in (dislocated) projects, avoiding the degradation of information, (according to figure 4.1 on page 43) and at the same time increase the capability of the project team/institution to successful and persistent information acquisition and knowledge assembly.

Thus the second part starts with an analysis of consequences of possible system decisions and the problem of longevity of digital information. Following those findings, strategies for persistent information- as well as concepts for knowledge-management are suggested. The second part is concluded by a review of concrete methods in project cooperation in dislocated environments and to connect the numberless ICT systems, a unified information access system is proposed as an important factor in information integration and usability even for the non-technical user.

6. Systems

6.1. Introduction

In this chapter the foundation of IT (communication) infrastructure is discussed as well as the consequences for society and politics. Before building concrete functionality, alternatives have to be taken into consideration. One aspect is the technical evaluation of different approaches. This is an important step as it is very hard to change the technical basis for a running project, even if only parts of the systems need to be changed[1]. But I will show, that the policy of the system may be in the long run even more important then technological issues, that may change in a brief period of time anyway. I will discuss the different approaches considering open and closed systems, discuss the effects on development processes and even more important the consequences of monopoles in IT infrastructure and the necessity to support the open source community. I will offer arguments for the importance of building a basic communication infrastructure for the "digital society", that is based on open protocols and open sources, to avoid unnecessary dependencies on monopolistic companies, with all negative consequences for society. Those arguments are founded on the perception of many authors, that something like a *third industrial revolution* [118, 129] took place changing the economical and social conditions fundamentally, as the importance of knowledge and conventional resources shifted dramatically.

Finally a conclusion is drawn with some suggestions for decision taking in implementing new IT infrastructure. Those findings also were an important factors in the planning phase of our own systems, that will be described in the third part of this thesis.

[1] We made this experience in the first part of the OSWP project. At the beginning of the project there were nearly no open source application server available, and the direction we decided to take ended up being the wrong one. Though the application server is still available as OS project, we decided to move to the meanwhile stable J2EE standard. In fact we have to rewrite the complete project, though the basic language (Java) and the functionality stays the same.

6.2. Extreme Positions

6.2.1. Only Available (Open) Tools and Systems are Used

One "extreme" approach to the problem of building a CSCW system is the idea to refer only to already available and well known (open) tools. Such an approach was shown e.g. by Jon Udell [123], who was responsible for the ICT infrastructure of *Byte magazine*. As we know, unfortunately, Byte magazine is not existing any longer, and so his ideas were inspired by a time, where the Internet was rather new (I would say between 1996-1998). However, the described approach was pretty interesting and contained a mix of available systems like: Apache webserver, nntp (news) protocol and server, ftp and file-system for resources. Only the glue between the systems has to be written in some script language (Perl at this time, today one might consider a more recent language like Python or PHP).

What are the characteristics of such a system? To start with the advantages:

- The system uses available open applications, hence the functionality that need to be implemented from scratch.
- The system is based on standard applications and protocols. This might make interoperability easier.
- There is no need to deal too much with bugs and other problems of the used systems, since one might hope, that the community of the used system will correct the problems, and the system "simply" has to be patched regularly[2].
- Users might be trained to the use of some of the systems already, so not so many new features have to be introduced (e.g. they might be used to the web-browser and the email client already, . . .)

On the other hand there are considerable disadvantages:

- The complete system is a mix of different subsystems which are connected by some application logic (the "glue" mentioned above). This makes installation, administration, backup and so forth rather difficult[3] or unpredictable because of the possibly unclear dependencies.
- In the end, one might find not having the functionality (of the important parts of the system) under control. E.g., some extension to the communication system

[2] Of course, this can be seen also as a potential risk of the system, as will be outlined later.

[3] Just imagine the user management. If login is required, different user management systems of the used applications have to be connected somehow.

needs to be added, and one would need to modify the email application[4]

- Integration of those applications might not be as easy as expected, since they might be developed following different philosophies, using different databases, different programming languages and so on.
- Each extension of one of the applications might create additional problems when trying to patch or update this particular application.
- And finally, and this may be one of the main problems: the complete system is not homogeneous; neither for the end-user (unless a complete new user interface is developed only using the other applications or protocols in the back-end) nor for the administrator.

Considering the *pros* and *cons*, the decision for a strategy — in my opinion — highly depends on the question how much functionality is already available in existing packages and in the intended use of the final system. If the application has to be designed for one specific company, this combination of available applications might be a good idea, especially if parts of those applications are already in use.

However, we decided to build our *Open Science Workplace* (OSWP) from scratch. The main reasons can be summarized as follows:

- We have complete control over the desired functionality. This is especially important as OSWP offers partly functionality which is not yet available in open source applications.
- It is easier to build a web-application that presents the functionality homogeneous to both the end-user and to the administrator(s).
- The software is easier to install as it is (hopefully) “one package” to be installed in an application server.
- However: some functionality like email is not rebuilt in the OSWP application, since external protocols are used for this purpose to let the system smoothly integrate to existing stable solutions.

Using this approach we hoped to be able to develop a flexible yet easy to use groupware application that offers all the required functionality through a unified web-based user interface.

[4] If you do so (which would be possible using open source systems), the main advantage mentioned above, namely the use of a tested, complete system is lost. Moreover, you would create a new e.g. email application on the basis of an available product, and it would become very hard to use updates of the available email program.

6.2.2. Using Proprietary Systems

An interesting alternative (especially from the technological point of view, but not so much from the "political", as described later) to the scenarios described in the previous section is the use of available commercial systems as foundation for further development. As already mentioned in section 1.4, mainly two systems are widely used: IBM/Lotus Notes and Microsoft Exchange/Back-Office. Both systems, but especially Lotus Notes offers an interesting and powerful groupware base-functionality. Microsoft Office/Exchange is often used, although the groupware functionality is far behind the Lotus Notes system and the continuously changing APIs and application interfaces and more important the Microsoft policy (see 6.3 for more details) make the development of a Microsoft-centric solution not desirable from my point of view.

Lotus Notes on the other hand was pretty attractive, considering the basic functionality of the Domino database and the messaging, workflow and synchronization (!) features. Nevertheless there are also a lot of reasons why Notes was discarded. In fact, the "Notes world" is very different from other usually well-known and established systems like Unix or Windows. This makes it difficult to find developers outside specific (company) domains that know how to develop in the Notes/Domino context. Moreover to use the full Notes/Domino functionality a Notes client for every user is necessary. The notes client is not easily available for every potential project partner and is only running on the Microsoft Windows platform. Also the administration and installation of Domino server is not too trivial either.

Thus both systems may be good choices for solving problems in specific (commercial) domains. Our project domains were: web based learning and distributed project management, as mentioned earlier. Additionally I believe, that projects developed for university or scientific use should try to be built as and upon open systems to support the foundation of an easily available and flexible public IT infrastructure, as will be explained in more details in the next sections. This is even more important if the development is financed by public authorities, because the results should be available for everybody who is interested in the project. More details about the open source policies will be described in the next sections.

6.3. Open and Closed Systems

6.3.1. Protocols and Standards

A definition to open source and open protocol has been given in the first part of this theses and will not be repeated here (see section 5.1 on page 51), albeit a closer look to the meaning of "open" seems to be necessary.

> "In terms of open standards, the most widely agreed use of openness implies

> a willingness to accept external input during the standards development process.
>
> [..]
>
> The companies that have the most to gain from closed source programs and closed specifications are sometimes the quickest to misuse the terms open source and open standards." *Ken Krechmer [53]*

So there is obviously an essential difference between the term *standard* and the term *open standard*. The difference comes mainly from a different process. A "normal" standard can result from commercial, proprietary software, that dominates a particular domain, e.g., because the vendor has a monopoly in this field. A popular example for this are the *Microsoft!Office* formats like *Powerpoint* or *Word*. Such standards are also called *de facto standard*, as there was no (public) process in finding the standard, but more an *ad hoc* decision of one party.

Such *de facto standards* can be exploited by companies that have a monopoly in specific segments, and much more, can create feedback loops, which are very destructive in terms of a healthy economy.

> "Microsoft is giving the people what Microsoft wants because it has a monopoly, which isn't based on the value of the product but rather a positive feedback loop in the information economy: Everything is compatible with Windows, ergo, Windows prevails and continues to prevail regardless of it's liabilities. It's No.1 because it's No.1, period, not because it's valuable" *Ken Krechmer [54]*

It sometimes came so far, that proponents of the Microsoft world noted, that some products of competitors are not "compatible". This sentence, of course, makes no sense at all, in the same way as the formula "$x =''$ is incomplete.

A product can only be compatible to some other product. Using this notation they wanted to demonstrate, that there is nothing else important enough to be considered than Windows or Office, so when writing "System x is not compatible" it must be clear automatically, that the true meaning of the sentence is: "System x is not compatible to Windows, MS Office ...".

What we can learn from this quotation and the arguments above are at least two things:

- Standards are, no matter whether they are open or closed/proprietary, extremely important, as they can generate positive feedback loops.
- If the process, that drives a standard generation is dominated by a single player (e.g., one company), especially when this is done in a closed manner, *de facto standards* may easily be misused for marketing purposes.

As a consequence the development of software (that will serve as information backbone of a company or an institution and will possibly be mission critical) has to take care of *open* standards. This is likewise especially important for the ordering of software, particularly when it is dealing with basic communication or cooperation infrastructure, as well as essential (operational) back-end systems. One has to insist that only software is acquired that is fully compatible to available standards in the field. The reason is simple: information exchange, traceability and *longevity of digital information* as well as easy integration into other IT systems has to be guaranteed. Obviously information access to resources of the system has to be granted even if the vendor or programmer of the particular software is not supporting it any longer. If the software is based on open and transparent standards, it is very likely, that every expert in the field will be able to access the system. This is also the reason, why many companies (especially the mentioned) do not tend to open their data formats, making it through this closed strategy very difficult for potential competitors to work with these proprietary systems.

So this strategy helps consolidate monopolies as well as it keeps *prices high*, *systems obscure* and *competitors out*.

On the other hand, the use of software built on open protocols gives the customer the power back. It is unreasonable — in my judgment — to buy a system, where the vendor tries to build barriers to the access of the data of the customer.

For most areas, there are organizations that try to support the discussion for building open standards and also try to publish the results to the public. One important example is the *World Wide Web Consortium* [125] (W3C). This organization deals with protocols and standards in the domain of the web, data exchange and information systems. Many companies, universities and other organizations are member of the W3C and try to develop useful standards for software information systems. Hence it is highly recommended to check, whether a product *precisely* follows standards defined for the particular domain.

The word *precisely* is important, as there is the so called *embrace and extend* strategy, that some companies try to exploit: This means, that first of all, it is announced, that a specific standard is used, secondly, this standard is modified slightly with the main intention, that the compatibility to other products is no longer guaranteed. Not only, that the product then is no longer standard-conform, even worse, the standard itself can be damaged. Examples for this is the “Kerberos” strategy of Microsoft [79].

> “Microsoft is certainly not the only company that desires to control its markets beyond the value offered to their customers. The desire for such control is quite understandable in a capitalist system. It is the unrestrained use of such control that must be tempered by open standards, competition and government action (if necessary).” *Ken Krechmer [54]*

6.3.2. Open Source

Also the term open source was defined in the first part of the thesis (see 5.1 on page 51). To conclude the properties:

- Open Source software means, that the software sources are available for everyone who is interested.
- The sources need not only be available to everyone interested, but it also should be allowed to be modified if desired.
- "New" products on the basis of an existing OS project may be developed and distributed[5]
- Open Source Projects, if they are successful, usually attract a broad user community, that brings positive feedback and other advantages for the project, that are analyzed in detail in the next sections.

The importance of open standards was discussed in the previous section. Of course, the use of open standards is not limited to open source software, as also a lot of commercial "closed" software is supporting open standards[6]. Nevertheless the use of open source software usually brings more advantages for the user community. It starts with the fact, that the software functionality is transparent, which is favorable for security-critical applications. The fact, that usually a significant web/newsgroup based community is available makes solving problems a question of hours or days.

6.4. Differences between "Open" and "Closed" Development Processes

6.4.1. Software Engineering vs. Open Source Engineering?

Obviously open and closed processes differ in the development- as well as in the business processes. Meanwhile, Software Engineering (SE) is a scientific discipline. But SE tended to deal mainly with traditional software processes, where design, management, testing and other rather formal and organizable tasks are well analyzed and clearly described, as well as supported by process descriptions packages like the *Unified Modeling Language* (UML) [73, 16] and the *Rational Unified Process* and software like the *Rational Enterprise Suite* [85]. The aim of this discipline is to make the development of software an engineering discipline like other engineering disciplines (e.g.

[5] Depending on the specific OS license the "new" project may be used in a commercial context. For details see section 6.4.7 on page 69.

[6] To give in example: Most current graphic software supports scalable vector graphic (SVG) standard, modern office packages like Open Office or Star Office use XML as basis for their data formats.

mechanical, chemical, ...) leaving the unclear and unpredictable "hacker" processes, often depending on the capabilities of few individual developers[7]. The rise of very successful open source projects did even astonish many SE experts as there is no obvious plan behind many of those projects and the development strategies seem to be more "ad-hoc" approaches.

Nevertheless, the open source (OS) scene impressively demonstrated in many projects (though not in all[8]), that their processes are capable, and the quality of OS products tends to be higher then the quality of many conventionally developed systems as the examples of Apache Webserver and the Linux operating system show. Recently studies have been done to analyze the strategies and methods of some successful OS projects. Mockus et.al. [68] analyze the Apache and Mozilla projects in detail. Some interesting findings were published in the form of seven hypothesis. Four of them should be mentioned here:

> "Hypothesis 1: Open source developments will have a core of developers who control the code base. This core will be no larger than 10 to 15 people, and will create approximately 80 percent or more of the new functionality.
>
> Hypothesis 5: Defect density in open source releases will generally be lower than commercial code that has only been feature-tested, that is, received a comparable level of testing.
>
> Hypothesis 6: In successful open source developments, the developers will also be users of the software.
>
> Hypothesis 7: OSS developments exhibit very rapid responses to customer problems." *Mockus et.al. [68]*

This and other studies show, that the approach in the OS development usually allow fast response to problems as well as the fact, that bugs are often faster detected and removed by the inherent transparency in the process. Moreover (and this is an essential finding for this thesis) this study as well as other studies [139, 27] (which focus more on the aspects of cooperation and communication in OS projects) prove, that electronic cooperation and CSCW systems are crucial for the success of OS development[9].

Concluding these findings, there are interesting aspects to note: Commercial driven software usually targets a specific (marketing oriented) goal. The consequence is a clear

[7] Many new suggestions have been made to solve this problem, among others the *extreme programming* concept is currently a very popular one [12].

[8] But it is to remark, that also a huge amount of "commercial" software developments fail. The Standish group reports a finding, that about 30 percent of projects will be canceled before completed, and more than 50 percent cost nearly the double amount of money, then originally estimated [111].

[9] This is also shown by the overwhelming success of the Sourceforge web-platform. Though this is mainly related to the fact, that Sourceforge offers free source code management through CVS and webspace for project pages. The groupware functions are not so often used.

time-schedule for releases (at least they are planned. . .), a specific feature set, a desired market position. Watching OS projects those things are usually not of big importance. Often OS programs are initiated by problems of single developers (consider Linux: Linus Thorvalds searched for a license-free Unix-like operating system which was not available at that time [122]). Then it sometimes happens that those projects create a bigger impact in the community and a OS community arises (even with commercial spin-offs like RedHat or SuSe). Nevertheless those projects are not driven by clear schedules for releases or particular feature announcements. Development always has more of an ad-hoc process[10].

6.4.2. Communication and Collaboration

Yamauchi et.al. [139] analyze the communication and collaboration efforts in OS development which is mainly limited to the Apache and FreeBSD projects. Besides detailed analysis of communication channels an important thesis is expressed:

> "The culture of open-source communities can be characterized by the orientation to a rational rather than lateral approach. Members try to take their behavior logically plausible and technologically superior options are always chosen in decision-making.
>
> [. . .]
>
> One informant said:
>
> That is, we don't meet face-to-face. Then, we need some criteria to decide something, right? The criterion that everyone understands is finally only technologically good or bad." *Yamauchi et.al. [139]*

This is an important consideration. In fact many OS projects seem to be *problem* and *technology*, not *marketing* driven as most commercial products. This is a huge advantage especially for *back-end systems* like databases, webserver, application server and so on, but can be a problem for *end-user systems*. The reason simply is, that writing documentation, adding features for unexperienced users (like the meanwhile well-known "wizards") or programming good installation scripts was not often found in OS products. Like Mockus et. al. [68] expressed, that usually OS programmers use their own software and such features obviously are not required for the expert user. But the good news is, that this has changed in the last years and more and more OS products start learning their lessons considering the *end-user friendliness* of their products.

Nevertheless there are some remarks to be made to the findings of Yamauchi et.al. [139]: First of all, I do not agree with the fact, that OS projects are somewhat perfectly

[10] Other OS project types are existing, mainly driven by companies. See section 6.4.3 for details.

democratic and everyone may pose his or her suggestion and the best technological solution is always selected. Many years experiences in different OS communities show, that tendentious this finding is true, but there are important social factors to consider: It is not true as Yamauchi et.al tend to express, that authorities are not of big importance. In fact *new authorities* form, independent of the “real world” status of the participants: Every newsgroup or mailing list has its “gurus”. Those are usually the core-developers or programmers important by some other reason.

Sometimes also those who are working more or harder have more authority than others only writing comments[11]. Generally, their opinions are of higher relevance than the opinion of “ordinary” users. Though the authority might not be commanding as in “real world” settings, but it still exists. Who would not assume, that a posting in a Linux newsgroup by Linus Thorvalds would have more impact than a posting of myself. And there are good reasons for this. However, sometimes it might be true, that this leads to the wrong way. So I believe — though the basic tendency toward better technological solutions are still valid — that these new social factors are underestimated.

6.4.3. Project Types

The paper of Yamauchi et.al. also focus on a particular project type. So the term open source seems to be too narrow in this article. Nearly exclusively Apache-style developments are considered. To analyze these types of projects is very important, but there are far more types of projects that should not be neglected like:

- Traditional developments: e.g., IBM open source donations to the Apache pool or the mySQL database.
- University (Research) Projects: Teams of small groups often donated to sourceforge finally.
- Multiple group efforts like OSWP.

The first project type usually is developed either by a conventional company development process, but after some time the company decides to donate the sources to the community. Examples are the *journaling file systems* of IBM or Silicon Graphics. Another usual situation is that very recent technologies are developed in research laboratories of big companies. This is especially true for IBM research: Sometimes (parts of) these technologies are then released to the OS community. Examples are the Xalan XSLT processor, Xerces XML parser, SOAP libraries, and so on. In those cases software often was programmed using traditional SE processes, but as soon as

[11] There are e.g. special features in certain newsreader that automatically highlight messages or threads of particular users. This increases the impact of certain community members.

they are open sourced either the process is changed, or at least the positive effects of transparency are noticed.

The second and third type are usually university hosted and problematic in some cases, especially when those projects are mainly driven by PhD or diploma workers, as the risk is high, that the projects are not supported any longer as soon as the PhD or diploma thesis is finished. The reuse of university projects will be discussed in more details later in this thesis.

6.4.4. OS Processes and Support Problems

Another interesting aspect is the question of support in case of problems, security holes or the need for modifications and adaptions. Besides the fact that meanwhile a lot of companies (including "big players" like IBM and HP) offer business-support for certain OS projects like Linux, Apache Webserver and others, it was always a pleasant fact of many projects, that support through the "usual" project communication channels (web, newsgroups, mailing lists) is often better and faster compared to many commercial products. The reason is simple: OS projects are developed *by definition* in public and are dependent on the user feedback. So support and feedback often are the same thing. Furthermore the open source code is obviously an important information source for the expert developer. It allows to solve specific problems or even add bug fixes or new features. This is completely impossible using closed-source software[12].

In contrast to this open communities commercial developers normally are not allowed to make support through such channels, though there were certain exceptions (like the Borland developers reading and answering specific questions in newsgroups in the 90's).

The practical consequence is, that usually feedback in high quality is available in days or sometimes even in hours free of charge. These questions are, as mentioned, sometimes input for the developers to enhance the product, remove bugs or add new features. Hence the process is stamped by shorter release cycles compared to conventional development processes with secret sources.

6.4.5. Security

The consequence from the analysis above is not, that OS software has by definition higher security or quality from the first release on, compared to commercially developed closed source software. But the point is, that as soon as the project has gone through some development cycles and is used for some time, security problems and

[12] At least without performing reverse-engineering efforts that are meanwhile partly illegal by laws like the *digital millennium copyright act* (DMCA). Moreover this code-inspection allows a developer a deeper understanding and control of the system(s) used. This leads to an implicit knowledge transfer as described in more details in section 6.5.3.

potential holes are detected faster and are removed easier. This also has to do with the philosophy behind the development process:

OS development is by definition *open*, *transparent* and *liberal*, which means, that all sides are interested in serious information. Hence also problems and security holes are published fast and patches usually are available after short time. Traditional software companies on the other hand are by nature not too much interested in big publicity of problems or security holes in their software, whereas the open source community is living from this open and transparent setup. Consequences are clear: Many OS products are seen by experts as more secure than commercial competitors like the Mozilla browser or Apache webserver [103, 102].

Even Bill Gates has realized this issue and declared a new strategy of Microsoft, namely *trustworthy computing*. Security should be prior to new functions. Sounds good, but is this believable, considering the policy of Microsoft in the last decades? Or is this simply a new marketing strategy by Microsoft, which is well known as marketing company and less known as technology leader. The outcome of this would be a complete modification of the Microsoft development, production and marketing strategy, and it is doubtful that this will happen, as it would change the company substantially for the next years as the well-known security specialist Bruce Scheider explains:

> “Bill Gates is correct in stating that the entire industry needs to focus on achieving trustworthy computing. He’s right when he says that it is a difficult and long-term challenge, and I hope he’s right when he says that Microsoft is committed to that challenge. I don’t know for sure, though. I can’t tell if the Gates memo represents a real change in Microsoft, or just another marketing tactic. Microsoft has made so many empty claims about their security processes – and the security of their processes – that when I hear another one I can’t help believing it’s more of the same flim-flam. [...]
>
> And they’re going to have to reverse their mentality of treating security problems as public-relations problems. I’d like to see honesty from Microsoft about their security problems. No more pretending that problems aren’t real if they’re not accompanied by exploit code, and attacking the security researcher if they are. No more pretending security problems aren’t caused by bad code in the first place. No more claiming that XP is the most secure operating system ever, simply because it’s the one they want to sell.” *Bruce Scheider [102]*

We have to wait whether this is a substantial change in the Microsoft policy, simply a marketing gag or even worse, simple marketing for new products that allow to get

Name	**Grants**	**Condition**
MIT	copy, modify, redistribute	original copyright lic. must be retained
BSD	like MIT	acknowledgments in advert. and docs
GNU General Public (GPL)	copy, modify, redistribute	derived programs also GPL
GNU Lesser Public (LGPL)	see GPL, mainly for libraries	all necessary object files are to be provided
Mozilla	see GPL	allows use of proprietary CS software
Apache	like MIT	redistribution must contain license information

Table 6.1.: Different Types of Open Source Licenses. Details can be found e.g. at [75]

control over the user and about his or her usage patters and data (Palladium, TCPA) [117].

6.4.6. Versioning

Additionally one should not forget the fact, that OS software usually has a far more realistic *versioning*. As it has become fashionable to make nearly arbitrary changes of version numbers (as it seems not to be acceptable that the product of a competitor has a higher version number than the own product), or remove version numbers at all (Windows 95, ME, 2000 and so on). Moreover commercial software is released often far too early with too high version numbers. E.g. a very popular Java integrated development environment was released with version 1 which was realistically seen at best a technology preview, not even an alpha release. This is mainly confusing for the customer. OS software usually has a more serious versioning. Versions below 1 are often already stable but not seen as production proof. Versions above 1 are intended to be ready for production setup. Additionally alpha and beta and even gamma test versions are available a long time for testing purpose until a real new version is released (e.g., mySQL, Linux Kernel)

6.4.7. Different Open Source Licenses

Besides different "origins" or systems of OS projects, there is also a set of different licenses. A detailed discussion of the different types is not a main focus but this section should give a brief overview. The most common licenses are compared in table 6.1 The licenses have in common, that they give a *person*, *institution* or *company* the right to use the software, modify it, install it on as many systems as needed. Moreover copying is allowed, and the license does not discriminate against fields of endeavor. Usually there are no limits in use for commercial purpose. Though sometimes there is the “limitation” that software based on a specific OS license has to be OS as well.

6.5. Open/Closed Systems and the Knowledge Society

6.5.1. Alternatives and Risks

The differences in the processes between open and closed source developments are described in the previous sections. Those findings have further consequences and new questions are posed: As mentioned, it is usually possible to solve problems with most OS products within days or even hours and receive solutions of high quality. But it is essential to note that this is not *guaranteed* through the normal OS channels. This is a clear consequence of the fact, that OS communities are founded on a *voluntary* basis, where an obligation of some kind is not available by nature. This is unproblematic for many applications, as solutions are usually found in brief time, but for many business applications this fact is not acceptable. Consider software in the financial sector based on open source, where a guaranteed support for specific problems in a defined time schedule needs to be granted.

No responsible manager wants to depend on a voluntary Internet community, lacking some commercial and guaranteed support level. Meanwhile this is available for many systems: from special vendors like Suse or Redhat in the Linux segment, SAP in the area of business software and databases (SAPDB) or even from companies providing complete solutions (from hardware, software, installation, support, education and other services) like IBM or Sun Microsystems. So finally this argument against the use of OS products in critical areas is not valid any longer (for many products).

Nevertheless it is clear, that this aspect of OS software has to be taken into account, when critical applications are planned. Especially it has to be evaluated if commercial support of high quality is available for the desired OS products. Albeit the difference in support between free and commercial software is often not so big as some companies try to make us to believe. Even for many commercial products there is no stable support available[13]!

Additionally there is always the restriction that closed source software may not be modified by the customer, which can create unnecessary delays especially in big projects where time is a critical factor (financial sector, insurance software and so on). This restriction may even be the source of security and transparency problems. In the

[13] This might have to do with the fact, that it is pretty easy for a software company to operate on a global scale in terms of software distribution (mainly over the Internet). But on the other hand it is very hard to provide a global support for software, as this requires expensive local and localized structures. Especially in cases where fast personal interaction might be required.

case of OS software it is always possible for the expert to check what the software is doing to solve specific problems. It is easy to evaluate which data is transferred between what kind of systems and under which circumstances. In closed source software one basically has to trust the software vendor.

In a time where nearly every system is connected to the Internet, there is always the risk, that parts of the software is connected to the vendor to exchange data whatsoever. Lately there were discussions about operating systems sending data to the vendor as well as printer drivers sending detailed usage patterns including IP addresses back to the producer.

Companies should be aware of those intransparencies and potential security holes in CS software products and evaluate whether there is no OS software available that can solve the particular problem in a more transparent way.

As a side note: There *is* a reason, why security experts do not except *security by obscurity*, which means that cryptographic systems are only accepted when the method as well as the source code is open and clear documented. There is, in my opinion, no reason why this should not be true for “ordinary” business software [93].

Another critic sometimes heard is the fact, that a continuous development of certain OS products is not always guaranteed, as the programmer or the team might loose interest in continuing the development efforts. Basically this risk is real, and there are many OS projects that have been terminated. But interestingly enough: the successful and really good ones always found new teams to continue. Having a closer look, this argument becomes a boomerang for closed-source vendors: First of all, continuous development is not even guaranteed in commercial products. There are many products where development and support has been stopped. Either because they were not successful enough, the company changed the policy or because the company ran into (financial) problems. One could bring examples like: Lotus Improve (an innovative spreadsheet application), where many users had problems in getting access to old data with recent other applications.

So the problem of *terminated applications* might pose much bigger problems in commercial closed source applications as there is no chance to get a grip to the application sources and continue the development oneself or at least use it to transfer data to other systems. So if a company decides to stop development of an important application, customers may run into severe problems. If an OS project is stopped, it is especially for bigger companies or projects possible to keep the project alive. There are several examples for this. Moreover it turns out to be a good idea for big projects to work close with the sources of infrastructure applications like application servers or databases, as this allows better insight into the complete system and hence makes problem solving easier and faster, compared to be dependent on external support. This makes the complete system more transparent and gives the developers a better feeling, as they always have the option to look under the hood, compared to using a closed source “black box” system. Additionally one should not forget, that OS projects usually try to stay close to open protocols and formats. This makes it often easier (compared to many commercial products) to migrate from one system to another.

This strategy is especially a good idea for projects of a specific size, typically in companies like banks, governmental institutions, health care system and the like — also because of *high security requirements*. OS projects are perfectly suited to serve such applications. Of course time is needed to get comfortable with the system and understand the details, but as soon as this knowledge is "in the house" development becomes much faster, stable and more flexible and still very predictable. An interesting side-effect is, that costs are lower since no license fees have to be paid — though I would assume, that the "flexibility factor" is often far more important.

Companies like Microsoft detect the importance of the fact, that especially big customers want more transparency and develop strategies like the *shared source* initiative. Albeit the only company, that uses this shared source idea is Microsoft themselves, because customers may inspect *parts* (!) of the code, may *detect problems* and *give this feedback to Microsoft*. Customers neither see the complete code, nor may change the code. So this is by no way comparable to the idea and quality of OS projects!

The outcome of this is, that OS systems can be a relevant factor in a free-speech and a democratic society and economy. Particularly also thinking about second world countries where OS allows cheap and stable IT development without unnecessary economical and political dependencies from some first world countries like the USA or world-dominating IT companies. The fact, that OS projects basically allow cooperation from all interested parties, might be one of the most impressive advances of the Internet community in this area. If some participants would not agree with the development of a particular OS project, there is always the option to start a new OS project on the basis of the old one. Such *forking* has occurred already occasionally. To give some examples: Interbase/Firebird (database systems) or PHPNuke/PostNuke (content management systems on PHP basis).

6.5.2. Economical Consequences

> "Cooperation is more important than copyright. But underground, closest cooperation does not make for a good society. A person should aspire to live an upright life openly with pride, and this means saying *No* to proprietary software. [...] You deserve free software" *Richard Stallman [86]*

Companies like IBM, SGI, Sun, HP, Redhat, O'Reilly [76] etc. show, that open source and commercial success need not to be a contrast. IT systems become more and more complex, and implementation, adaptation, maintenance and education produce a need for support, that guarantees success for companies in the OS environment. Moreover synergies can be used. If IBM supports Linux on all server platforms from z-Series mainframes down to desktop systems (and recently also palmtop systems and watches), then this is a clear strategy to sell complete solutions to the customer. There is a need for open, stable and transparent systems, IBM and others realized this, and offer such systems to their customers. Good business is made by hardware, education and other services.

Maybe there are also other reasons for OS commitment of companies: it seems that supporting OS projects generates a positive image for those companies leading to a further factor for business success. This could be observed at IBM, but also at companies like Redhat or SAP. Also Sun Microsystems tried to support specific projects like the *Apache Software Foundation* or *Netbeans*; but OS proponents are very critical in analyzing those commitments whether this is just a marketing gag or a real support for the OS idea. Sun's policy was not always clear (particularly in the Java licensing) and the consequence was far less success than there should have been possible (especially in Linux environment). This is the same reason, why the shared source idea from Microsoft is doomed to fail from the beginning, as it was immediately clear that the core idea was to use the capacity of the community to enhance products without giving back adequate value.

On the other hand there is the eligible question how the development of OS systems is financed. It is not easy to give a brief answer to this question: Partly this development is financed by companies like the mentioned, either by opening available software, or by supporting communities like the *Apache Software Foundation*. This can be an advantage for those companies as it is often possible to establish a vivid community for their projects that help stabilizing products, increase quality and lower the development costs. Other projects are initiated by governmental financing or are located on universities. Some projects are founded and driven by amateur programmers or professional developers in their spare time. I believe (as will be explained in more details later) that a more stable financial foundation for OS projects is required and should be performed by joint efforts, e.g., as a *European Community* program.

6.5.3. Effects on Society, Political Effects

> "It [Europe] talks about catching up but knows that the technological gap between it and the United States grows larger every day. The continent that invented culture now imports its culture from America. The equivalent of "Intel Inside" could be printed on almost everything new in Europe."
> *Lester C. Thurow [118]*

It seems important to understand, that some market leaders in information technology still try to control the market using secret source code and even more importantly, as explained earlier, using proprietary (data) formats and protocols. This *business with the secret* has nothing to do with fair trade or free markets, but is simply disadvantageous to the customer und the competitors. As a final consequence: it is a serious damage to those companies who try to develop new innovative technologies. This strategy works fine, because politics and legislation (especially the European) has overseen to define appropriate legistic conditions to avoid such monopolistic tendencies. On the contrary, it seems that the next severe political mistake is on the way with software patents, the ostensible protection of copyright and recently, overlooking the serious negative consequences of TCPA, Palladium.

Market and financial power seems to dictate nearly arbitrary conditions to the customers (particularly driven by huge media companies). This has nothing to do with the ideas of a free market and the competition of ideas and solutions, as the basis for innovations. On the contrary: the currently successful companies are forcing laws that guarantee their dominance, despite of the quality of their products. Interestingly enough this negative tendency comes mainly from the United States and supports mainly US companies.

Also we should not underestimate the effects of this problems to the society. We move toward a society in which nearly all relevant areas are controlled by very few companies. This involves the desktop computer, business software, mobile communication or even new areas like *e-government* [5]. Especially the last *buzzword* shows the dimension of the problem: I really doubt that it could be wishful if one global acting monopolist will control the governmental infrastructure (e-government) with proprietary software and protocols. Not only that this is highly problematic because of the intransparencies of these systems, but much more importantly: society would put themselves into dependencies on these companies, with all the resulting negative effects.

There is another interesting factor not mentioned yet: the situation of *developing countries* like China, India, Indonesia, Korea, Brasil and so on. Many of those countries seem to detect (maybe more foreseeing than the countries of the European union), that OS software and commitment to OS projects may be an essential component in building a new IT infrastructure. On the one hand it is cheaper, but more important: it allows to *build know-how* in their countries by analyzing, implementing and supporting those technologies instead of a simple *passive* usage of imported technology. The well-known monopolists try to get into these new markets with aggressive marketing strategies like very low license costs. But it looks like those countries do realize the danger of getting dependent of those products. As soon as the infrastructure of China, India and so on is built upon proprietary systems (usually from US companies), those companies will continuously raise the license and support fees and those countries would — although they are already in a financial difficult situation — run into still bigger dependencies.

As those developing countries seem to realize their opportunities, the countries of the European union obviously did not realize their depency on foreign technology with all negative financial, political and scientific/know-how side effects. Considering the amount of money the countries of the European union are paying year by year for "Microsoft-, Oracle- . . . taxes" und the fact, that Linux and other important OS projects are strongly driven by European participation, the economical shortsighted policy becomes clear. If only a part of this money would be spent to support European OS communities and universities, nearly all products of the mentioned companies could be replaced by OS systems with *high impact on European innovation*. Moreover this would also be a useful implicit support for developing countries (without the need of additional money spent) and a higher degree in transparency, traceability and democracy too.

There are first initiatives, e.g., in Germany where all Windows NT servers of the parliament are replaced by Linux systems [21] and the cities Schwäbisch Hall and Munich replace Windows completely through Linux (in the bureaucracy) [104] (Original Citation see A.2.2 on page 201).

The reason why Linux should replace the NT systems is pretty interesting too (translation by the author, original text in German see: A.2.1 on page 200):

> "With this decision the IuK commission deviates deliberately from the result of the study, because this decision is the basis of a strategic consideration to decrease the current dependencies of products of a particular vendor. This decision should create more freedom for future decisions."
> *Press Release Deutscher Bundestag [21]*

This decision as well as the explanation is evidently long-sighted and considers not only short-term financial aspects. It was expressed clearly, that the decision for an open system has advantages for the future, because new functions may be implemented by different companies not only by one vendor. This might — as a side effect — also support *local* small and medium enterprises. Also the risk is reduced by this decision, the transparency and the competition between vendors increased. Of course this is still a very small step, but at least a first step in the right direction.

The decision of Schwäbisch Hall is again more interesting for different reasons: First of all, Schwäbisch Hall does not limit the replacement to servers but includes also all desktop systems. But maybe the most important decision is to initiate a *Linux competence center* that should support the community as well as other institutions or companies. About one year after this decision, even more impressive the German city Munich decided to migrate all 14.000 computer systems of the community from Windows to Linux. Not only server, but also all desktop systems. The arguments were nearly the same: The importance to remove strong dependencies from monopolies. This is a strategy that should be supported in a coherent plan by the European Union in all countries of the EU to enhance cooperation in the implementation of open and free IT infrastructure.

In conclusion I want to stress an important fact: The question whether a particular product (of a monopolist) is *better or worse* than the product of a competitor (closed or open source) is no longer the most relevant aspect. Even if such a monopolistic product might be excellent, it is not an acceptable situation if *one* company dominates a complete market (segment) with a specific closed source product. This is a very dangerous situation for the public as well as for the economic sovereignty. If — as an alternative (especially as an European concept) — it is considered to finance OS products with significant amounts of money, so that those open products can replace proprietary systems throughout Europe, this would also allow a huge decrease in software costs. Moreover IT competence and innovation can be brought back to Europe.

The other option is continuing the way Lester Thurow expresses it in [118] — as cited on page 73.

6.6. Dystopic Developments and the Knowledge Society

> *"It was the best of times, it was the worst of times, it was the age of wisdom, it was the age of foolishness [...]", Charles Dickens*

6.6.1. Dystopia

Following the ideas of Helmut Willke [129], I *do not* understand the term dystopia as a negative utopia like Georg Orwell's 1984. I see dystopic developments as the *blind spots* of a modern society, which has lost its "classical" and more or less simple and transparent order. The understanding, reception and handling of *knowledge* and *nescience* of this modern society changed dramatically and created new conditions for science, technology, economy and politics. Dystopia also means the believe of many politicians and managers in general (political, economical, scientific) to be able to solve *recent problems* (emerging out of those new conditions) with *inappropriate old strategies*.

> "The current modern spirit is so terrible progressive, that success as well as crisis emerge at the same time. If someone celebrates a success or a crisis, depends on the viewpoint and the selection of the channel.
>
> [...]
>
> A crisis of knowledge results basically because no one can count on the available knowledge. This is true until the complementary nescience is noted and made usable the same way as the knowledge itself. The crisis of knowledge means the inability to deal with nescience in a competent way[14]." *Helmut Willke [129]*

[14] The original citation is in German and is translated by the author. The original citation can be found in the appendix section A.1.2 on page 199.

In this section, the idea of dystopia is treated primarily in a theoretic way. Practical consequences are, e.g., the proposed model for knowledge management as a highly integrates CSCW process (for details see chapter 9)

6.6.2. Knowledge as Resource — a Factor for Success

> “With the advent of the third industrial revolution, skills and knowledge have become the only source of sustainable long-term competitive advantage. [. . .] The knowledge that used to be tertiary after raw materials and capital in determining economic success is now primary.”
> *Lester Thurow [118]*

In many current publications like [118, 129, 39] knowledge is seen as the major factor of economic success. Hence those companies will succeed which have access to most recent scientific research and create best innovations. So the power of a firm can be regarded as equivalent to the knowledge owned by the company.

I doubt this kind of simplification. Those ideas suggest, that there might be some very hard to acquire, kind of “magic knowledge” that the successful companies own. Moreover the perception resists, that knowledge is a rather difficult resource to acquire. As a matter of fact: the opposite is true (at least in most knowledge driven areas).

There is no doubt, that knowledge has passed the traditional resources in importance (at least if we exclude capital as the traditional resource). But one should not forget that the resource *knowledge* is a *highly volatile* resource! As an example consider the already mentioned example: Microsoft [54]. Astonishing enough, in the history of successful Microsoft products, there were nearly no significant innovations, that came out of the company. No technical innovations that would (following the usual theory) create the knowledge foundation of this company. A few examples should illustrate this: Graphical Operating systems were initially developed by Xerox Parc, and implemented successful already by companies like Apple, Commodore, Atari and others — long before Microsoft took over this idea and implemented it into the technologically inferior initial Windows versions. Spreadsheet applications were available before Excel, and this implementation was heavily “inspired” by applications like Lotus 1-2-3 and parts of the functionality from Borland’s Quattro Pro. Similar situation with word processors: Word Perfect was far superior until middle of the 90’s, for Desktop Publishing (Xerox again). Databases like Oracle and DB2 are far older and more mature than MS SQL server (which is again no original MS product), the Internet was completely neglected; then Netscape’s browser was copied, technology and companies were bought. If Windows NT/2000 was more innovative compared to the earlier Windows version, then hardly because of the huge innovative potential of MS, but much more because of the fact that the chief developer auf Digital/VMS changed to Microsoft. I could continue with graphic software (Visio was bought), the new .net strategy, which is mainly a "Microsoft Java for Windows" and so forth. Of course there have been some incremental innovations in different systems, but hardly any substantial ones.

This example is not so much a critic on Microsoft, but much more an illustration about the fact, that it is obviously not true that the *implicit and secret company's technical knowledge* (as resource), that was built up by the company itself, is the foundation of success. Actually the knowledge that drives most *new economy* enterprises is rather ubiquitous, or at least, it is rather easy to acquire or buy the necessary knowledge[15].

6.6.3. Conditions for Economic Success in the Knowledge Society

There is the paradox situation, that knowledge is one of the most important resources in modern economy, hence represents an extremely high value. The *economic value* of concrete knowledge on the other hand, is due to the high transitoriness and the ease to acquire new knowledge highly over estimated. Finally, the modern "production" of knowledge is self-amplifying. Electronic publishing, the easy access of books and Internet resources offer basically the opportunity to get in touch with the most recent findings to everyone. Hence the modern communication-driven world became in a way smaller than about 50 or 100 years ago.

A side effect of this development is the fact, that the importance of the *individual scientist* decreased significantly, while at the same time the specialization and differentiation in science increased[16].

The question is still open: what are the driving forces of success in the knowledge society?

> "Knowledge appears to add more to the value of a company than do physical assets. Microsoft, for example, a company with approximately 30,000 employees, has a market value in excess of $400 billion — many times its annual revenue. In contrast, General Motors, a 600,000-employee company with billions of dollars in physical assets (e.g., buildings, assembly lines, parts, and vehicle inventory) has a market value less than its annual revenue and 1/7th that of Microsoft. The contrast between Microsoft, a knowledge

[15] There are exceptions like some biotech or pharmaceutical companies, where a high technical, financial or mechanical effort is to be taken to acquire new knowledge, or expensive studies have to be financed (medical studies . . .). But even here, it is not so much the invaluable knowledge, that is important, but much more the financial and organizational capabilities to finance such studies and technological developments. Many researchers would have the know-how to build an excellent microprocessor, few have the financial background.

[16] An indicator for this development is, that there are hardly individual Nobel Laureates. Usually a Nobel price is awarded to multiple persons with the difficulty to really track down the originator of one idea or concept.

> company, and General Motors, a physical asset company, is not the exception, but the norm. Yet, it is imperative to recognize that Microsoft will only have exceptional market value to the extent that the firm continues to innovate, i.e., to grow and to apply its knowledge?" *Stewart et.al. [113]*

Also Stewart et.al. argue that knowledge is the relevant factor, but as analyzed above, this seems not to be the main reason. I assume that not the knowledge is the factor of success, but much more the *potential for problem solving.* This potential for problem solving is a sum of the factors: knowledge, information, money, management capabilities, marketing, size (!) and others. The result is a "future potential of profit" and finally determines the market value of a company.

The two factors *volatility of knowledge* (with all mentioned consequences) and *potential for problem solving* may seem to be antagonists in some ways. Many companies react to this problem by *hire-and-fire* strategies. The concept seems to be clear: knowledge is volatile, hence one can buy it when it is needed and fire the employees as soon as the particular knowledge is no longer essential. As analyzed in [120] this strategy might bring short-timed success but most probably will fail in the long run. The reasons are the following: The change in technology and knowledge might be fast, but the real problem is not *speed.* High speed is easy to handle, when it is predictable! This is an often misunderstood fact. The real problem is not speed, it is *unpredictability*, high risk about the *direction* of new developments. Thus companies try to control this complexity — and one strategy observed is to outsource and hire-and-fire employees as needed. But the consequence is often not the desired flexibility, but much more, that the complexity of the complete "system" is increased by those actions, as the employee and knowledge management becomes increasingly difficult with a fast changing company structures! Hence increasing flexibility in the described way, easily might create the opposite effect as desired: the complexity of the complete system increases and with the complexity of the complete system, the risk increases too. And this is obviously not what was intended.

Reading this analysis, one might come to the conclusion, that my there is some inconsistency in the concepts described above. On the one hand the expression, that knowledge is important, but volatile, hence can be bought when needed, and on the other hand, the fact, that higher flexibility might create more complexity and more risk. I believe the truth is, that the described problem itself is very complex and no simple single solution exists. The main issue boils down to reducing the complete (!) system risk, which cannot be done by singular actions, as the system is highly interconnected. It will be required to remove hire-and-fire strategies and build a solid (but maybe smaller) amount of well skilled and trained employees that have a high competence in problem solving (this is: being flexible, but without continuously changing staff). On the other hand: wise outsourcing has to be performed for not binding too much internal staff to specific problems as well as buying new *promising* technologies and knowledge as soon as it emerges somewhere. To do this, it is again important to have good skilled staff that is able to detect whether a new technology is promising or not.

Moreover there is the aspect of binding customers, particularly as customer relationship handling is an essential risk factor. As (to follow the example above): cars are highly standardized, customers can change the car producer from one day to the other. This is not possible in the information technology sector. Particularly Microsoft realized early, that the most essential factors are not technology, not innovation, not even quality: it is the potential to bind customers and to do the best possible marketing[17]. Everything seemed to be allowed (*embrace and extend* strategies, lawsuits against competitors, announcing unavailable products and so on) The whole Microsoft strategy of the last decade(s) was driven by those main factors, and many other IT companies try to do it the same way [79, 54]. The consequence is, that it became nearly impossible to change the software/service vendor without severe difficulties. If capital is available and customers are bound to the product line of a special vendor, every needed technology or innovation can easily be bought and incorporated into the portfolio when needed.

The importance of a solid customer structure is also demonstrated by other popular examples; consider Yahoo: today the Yahoo functionality could be rebuilt by experts in a year or so, but Yahoo is most successful because it has a good strategy in binding customers. Of course there are start-ups that seem to overtake the blue-chip companies because of good and innovative concepts. But the real problem of today's economy is, that those blue-chip companies have so much capital that they easily can buy the new company or if this should not be possible, they invest huge sums to rebuild similar products (see ICQ messenger, AOL messenger, Microsoft Netmeeting . . . again Microsoft had no innovation and just reacted to the ICQ success).

> "Very occasionally, entrepreneurs are the inventors of the new technologies that make change possible — but not often. [. . .] Entrepreneurs are risk-takers, organizers and doers, not usually thinkers and inventors. The characteristics needed to create new knowledge are very different from the characteristics to bring that knowledge into active use. J.P. Morgan built his companies around Thomas Edison's many inventions. Bill Gates has invented no new technologies and was never a creative software programmer. He is, however, an entrepreneur and a builder." *Lester Thurow [118]*

[17] The bundling concept of Microsoft (Office) can be seen as such an extremly successful strategy. The individual applications are hardly sold any longer; so even if a customer might need only the Wordprocessor, often the complete office suite is bought. This reduces the chances of competitors dramatically to sell, e.g., a spreadsheet application, as it is already part of the Microsoft bundle (and most probably already installed). Moreover systems are more and more connected and interdependent: Frontpage for example can only be used in an optimal way in combination with Internet Information Server and so on.

Though I believe, that the next decade will show that the situation proceeded even one step further. Not only the CEO's and managers are "non-inventors" and "non-thinkers", the complete core-companies are. The core competences and activities of most of the mentioned companies will be more and more reduced to *marketing* and (project) *management* as well as *customer relationship management.* The new technologies, the product, the innovations will be created by sub-contractors or bought when needed, but as analyzed above, with a solid core-competence of problem solving in-house. As it will be more and more easily possible to recreate new products on demand (see Microsoft's cell-phone activities). Those active developments will take place as long as there are competitors (like still in the cell-phone sector), but as soon as there is nearly no competition, quality will decrease even more, and prices will increase at the same time. This is what we could watch the last years with Oracle or Microsoft. So this may be the essential consequence of this analysis: When there is no (commercial) competitor left, there is the chance of open source projects to succeed: The reason why Oracle, IBM and Microsoft are under some, although little pressure today is mainly a result of high quality open source competition.

Moreover the often cited "rule" that in modern economy the *faster* dominates the *slower*, not the bigger the smaller is definitely not true any longer. Following the analysis above, the big ones are already that big, that advantage is nearly always on their side. The smaller one can be as fast as possible, as soon as the big one realizes economic chances the marketing and capital comes into play and the small, fast one is simply small and will be taken over or outperformed by massive capital driven developments (as can be seen at Microsoft's Internet developments, or XML additions to available "old" database systems).

As a side-note: *fast* is not automatically *good* (as the dot.com bubble showed): it is possible to run with maximum speed into the wrong direction. At the end of the day, the big players may wait patiently from a safe distance, which one of the "small runners" is successful. Obviously the bigger one are left behind (from a technological point of view). Nevertheless because of the more stable capital situation the big players can easily outperform the smaller ones. What is needed is a solid market evaluation (and preparation) and a good overview about the developments.

An important criterion left is flexibility. Even big companies can be flexible, as demonstrated by many American enterprises. It is notable, that even here the U.S. companies have an advantage over European companies. One reason is, that employees in the United States have far less rights than European employees and even more importantly: U.S. companies much more dominate political decisions than European companies. The consequence is, that in some areas regulations support U.S. companies in an inadequate way (e.g. by military spendings). The real question in fact is, if European companies can learn from this flexibility without dropping the important rights for employees. But considering the earlier analysis about the complete system risk, Europe has the advantage, that hire-and-fire strategies never were established in

Europe. Solid educated core-staff that identifies itself with the company again will be a factor of success in future economy [120]. Europe has advantages here because of the higher legal standards in employment as well as in a higher education of the "upper" third of the society *and* the lower two thirds of the society [118]. We should use those advantages in forming stable and well-sized knowledge companies that are able to react flexible to future demands because of their inner-flexibility not because of hire-and-fire setup. But we have to learn the capability to shut down businesses that are really no longer needed. Keeping "starving" companies alive for a too long time, was a major mistake in the last decades.

6.6.4. Consequences for Open Source / Protocol Projects

> "A strong patent system is by definition a system of strong monopoly rights." *Lester Thurow [118]*

In the previous sections the importance of open source and even more of open protocols/formats for the open society [80, 81] and democracy should have become clear enough. Still the question is unanswered, what the consequences for OS/OP development are or will be, when knowledge and information will be treated more and more like a conventional resource. As mentioned above, the problem is not, that the *resource knowledge* could run short for the open source community by "natural" reasons. The opposite is true. Though the danger appears from other sides: If the bearing of knowledge as a resource reaches to the dimension of resources of the second industrial revolution [118] like oil, steel or crop or even passes it in importance, power struggles will be logic consequence. Power struggles like the ones known in the history of the last century. Today a war is declared because of oil, tomorrow war will maybe be declared because of knowledge[18]. But it is clear, that the *resource knowledge* is far more volatile then traditional resources. Where it is easy to protect a ton of crop or steal, it is (as we know) hardly possible to protect music, videos or books as soon as the information is provided in digital formats. If knowledge once escaped into *open-nature* it will never be possible to control it again. This is the lecture we learned from *Napster* and other file sharing activities.

But there is one essential difference between the *resource knowledge* and the *resource oil* (to select one as an example): When the oil-owner trades oil and delivers it to others, the traded amount of the resource is obviously no longer available for the original owner. Considering knowledge as resource this is not true: giving away or taking away knowledge or information basically does not reduce the amount of information

[18] We might consider, that this is already happening today. War against Iraq or the conflict with Northern Korea could be seen as the first info-wars or at least information-conflicts. If the arguments of the United States are taken seriously, the reason for the second Iraq war is the ability of the country to make specific weapons. This is definitely a knowledge conflict.

or knowledge by the owner. This difference has to be taken into account seriously. It can also be seen from another point of view: There is no limitation in using information and knowledge, and as soon as it is once available, there is usually no particular (high) cost to keep it available. However there is a slight impact to be considered: the information or knowledge of the original owner is not reduced quantitatively but the *value* of the resource might be reduced. E.g., when thinking of experts in specific domains like mainframe operating systems: one reason, why this knowledge is very valuable is, that is is not available ubiquitously. *HTML knowledge* was valuable 5 years ago, today nearly everybody is able to publish information on the Internet (and at least believes that he or she owns the *HTML knowledge*)

Having realized, that knowledge is not controllable like oil or gold, industries with a high knowledge impact seem to run into troubles: high investments are done to create knowledge and evidently many companies are not interested in spreading this valuable knowledge for arbitrary use. In fact this is no real new problem: copyright and patent right are available for more than a century to control these problems. But, again, knowledge never was so relevant and at the same time that volatile as today.

This leads to companies pressure politicians to strengthen patent and copyright laws. But this happens at a speed that a serious evaluation of the consequences are hardly possible and in fact recent political and law-activities seem to reach far beyond any reasonable or understandable action: examples are software patents in Europe or the *digital millennium copyright act* (DMCA) in the United States. Without going too much into details here, the continued extension of the copyright period in the US ("lex Disney") is a clear illustration of interventions that abuse the original and good idea of copyright and patents. The intention was to encourage the creative, the inventor, the entrepreneur to invest time and energy into new findings or new works and to ensure fair wages for creative people. The intention was *not* to create continuous money flows for decades and generations of successors[33].

But the other side is — as the nerves of the current society are more and more depending on this knowledge, e.g., in the form of information technology infrastructure — that the real consequences are the following: Small groups of (open source) developers are evidently not having the resources (financial as well as legal) to pay patent fees or even to check if there might be a patent for a specific solution. Hence those projects will come into danger to run into patent problems. Observing the way (especially big) companies act today, it turns out, that the real function of copyright and particularly patents is not any longer to support inventors, but to increase the power and strength of already huge companies against the free market and against new competitors. Patent actions between companies with strategic patent exchange are the usual business nowadays. Was this the original intention of copyright and patents?

Those laws seem not to bring the desired advantages for the cultural/entertainment sector, but mainly support the already big media-monopoles, and strengthen their power against a healthy competition on the market. The dominance of those firms is often no longer founded on innovative products and services, but on laws. It seems to be a unique development since the planned economies of the former communist countries, that *incompetence* and *lack of innovation* is not subjected to the usual mechanisms of the free market, but that those companies are supported by legal actions. This is only possible, because the equilibrium of power has degenerated. Industrial lobbies dominate legislation, and the rights of the consumers are more and more ignored. Lester Thurow [118] points out proudly, that the success of America compared to the rest of the world is, that "[. . .] it's greatest strength is not it's ability to open up the new. It is it's ability to shut down the old." Obviously not any longer: The "old" (e.g., the music and film companies) became big and influential enough, that they are able to initiate their own laws to protect them in spite of their inabilities. What once was a wise idea, now degenerated to a protection system of incompetent companies. Incompetent in the meaning of reacting to customer needs and wishes. Hence "old" but outdated business strategies shall be kept alive and protected by law.

As analyzed in the previous sections of this chapter, the importance of a stable open software and protocol development *together with* to good conditions for commercial software companies is important. That this vivid coexistence possibly remains also in the future, consequently all required steps have to be undertaken to *stop the abuse of copyright and patent legislation*, as well as the se*vere negative effects for democracy* following to activities like TCPA, DMCA or Palladium.

Additionally society has to decide what kind of knowledge and information is considered as *core knowledge* for the future of mankind. One could imagine knowledge like: AIDS treatments, biotechnology (like the human genome project), IT infrastructures and the like. Whatever might be selected (in the end, this is a political issue): *a decision has to be met* in a joint effort! But the current status is, that politicians are tending not to spend too much money for research in particular areas (as the once mentioned above). Companies then invest high amounts of money into those research areas and consequently want to generate profit from those investments. There is nothing more obvious and logical in a free market than that. But finally, governments accuse such companies, that they are not willing to give their expensive AIDS treatments for free to the developing countries. This behavior is hypocritical: At first, decreasing research funds, letting the companies pay and take the risk, and in the end, when successful (commercial) research has been done, the results should be free, *if the results are critical for the society.* Of course, this is no future oriented policy.

Again: society and politicians have to take a decision for the future: Which (technological) research topics are crucial for the future society: Finally after having decided those key research issues, the needed financial resources have to be provided (at best in a joint European effort). My opinion is, that open IT infrastructure available for everyone is such a key technology, that should never be given away to *mainly* commercial research and development. Again: the same is true for other fields of research.

We have to decide: Do we want to have significant technologies and knowledge like IT infrastructure, medical knowledge, but also cultural heritage(!) as public domain available for everyone, or do we want to pay every time we want to see Mona Lisa or a Bernini sculpture, money to Microsoft's Encarta or whatever other commercial project?

This is the political decision to be taken now.

6.7. The Future of Information

This chapter dealt with the connection of IT infrastructure and society as well as with the connection of (free) information flow and the different development processes. Functionality was not a core problem of this chapter. First of all because there is no principal difference in the functionality of the described open or closed systems and secondly because functionality is, or should not be any longer the crucial criterion when deciding between systems.

New *features* are used as marketing arguments for (particularly) commercial software, which is interesting enough, as it turned out, that most systems offer functionality far beyond the real needs of the users. So why does marketing on the basis of features working? I think the reasons are simple: Still many people believe that *new* means *better*. And new "superficial" features are the easiest way to demonstrate that the software is *completely new* and hence *far improved*. So it seems to be right to change the user interface of all relevant applications with every new version, just because the new UI looks *cooler* than the old one. Is it better in terms of usability? Is this a professional approach to the usability problem? I do not believe so. Usability would most probably suggest gradual improvements and changes. Unless the old systems were so bad, that a complete replacement seems to be appropriate. So everyone might take his or her own interpretations about the software quality that changes the UI every two years.

I think this childish development has to be stopped. Especially in the professional sector a more serious approach toward software development and versioning has to replace these old strategies[19].

The arguments in this chapter should have lead the way to more important decision criteria when turning to a new IT system or infrastructure. And, by the way, it makes no difference, whether the system is developed in-house, or of an application (system) should be bought. Such a decision should not be driven by short termed financial

[19] Different taste is easy to still, as most new applications offer the possibility of skins, which means, that the visual representation of applications can be modified and changed easily. One additional comment: Again, if Microsoft would have been an innovative cooperation, they would have realized, that the Unix XWindows concept was far superior already before version 1 of Windows. Different "skins" were always possible using different window managers. This is a solid solution to this problem, not a playing around with a new interface every year or two.

considerations or a “featuritis”. Features can usually be easily added to every solid system (particularly when it is an open system), so this is definitely not the most important factor.

Decision makers should consider the following factors:

- How is the information handled and organized? Is it well structured, organized with metadata and stored in open formats?
- Are those open formats well documented, so that reuse of this information is easily possible (in other available as well as in future systems)?
- Is there a clean separation between data and processing?
- Does the system offer mechanisms to communicate with other systems, to exchange data and functionality?
- Are those mechanisms implemented in well documented open protocols?
- Is the development cycle of the system solid? Is there a clean versioning, open discussion about problems and solutions?
- Is there a user/developer community of significant size?
- Do the in-house administrators/developers understand and appreciate the system, or is it a management driven decision?

Of course many more could be found out, but those decision factors should show the direction. But maybe the most important decision factor is not explicitly mentioned above: it follows the perception, that every IT system will be outdated sooner or later: The more open, well documented and open-standard compliant a selected system is, the easier the migration to other newer systems will be. This is not only interesting with regard to a complete replacement, but also offers the possibility to change support companies. Expensive long-term bindings to specific hard or software or support companies is no longer necessary; Free competition on the market would be possible again. The best competitor should be able to handle the new system. But also access to data of older systems is a crucial factor (not mentioned in most marketing statements), and this topic will be discussed in more details in chapter 7.

7. Longevity of Digital Information

7.1. Introduction

> "In fact, the record of the entire present period of history is in jeopardy. The content and historical value of many governmental, organizational, legal, financial, and technical records, scientific databases, and personal documents may be irretrievably lost to future generations if we do not take steps to preserve them." *Jeff Rothenberg [90]*

Longevity of digital information is a key issue in contemporary information infrastructure, nevertheless the importance is usually highly underestimated, if seen at all. The last decades brought a fast change in the way information is managed and stored. Whereas in the 80's databases were already in use, nevertheless most essential information was stored conventionally in books, on paper in traditional archives. With the advent of faster systems and bigger storage units as well as better access technologies (like Internet or *Intra*nets) many companies, individuals and also communities turned to store most of their data in digital form. This seems to be a logical development and a wise decision as access becomes easier and more flexible; there is no need to send paper files through companies or even from one location to another. Many new possibilities like versioning, access control, backup, cooperation on documents and so on came up. Of course, those developments were also driven by a significant increase in productivity using such information management systems.

Generally this trend toward digitizing information is an extremely useful one, but certain aspects should be considered [58]. I always found it remarkable, that societies might be rather traditional in many ways, but nevertheless technology is implemented at an incredible speed at all places even very sensible ones[1]. Some considerations should be taken when implementing digital data storage. Not only on implementation for "big" company systems, but even for individuals. As Jeff Rothenberg expressed in [90]: "the record of the entire present period of history is in jeopardy" — hence I believe, that measures to ensure the longevity of digital information should be part of each IT system installation, even if it might not be easy to argue, as costs arise, with no immediate positive *short-term* effect.

[1] This can be observed even in rather fundamentalist countries like Iran, where information technology and modern communication facilities like mobile phones are ubiquitous although society is still very traditional.

Some of the most vivid issues will be discussed here, as well as some suggestions for managing important arising problems. Starting with the specific problem of fast changing hard- and software, archival issues, specific problems in the science and project management context and the future use of information in the knowledge driven society.

In the chapter about *information management* (see page 99 *ff*) the solutions mentioned here are described from a more technical point of view.

7.2. Hardware and Software Issues

Working with digital information obviously requires computer hardware, software and storage media. Unfortunately all three parts create different types of problems with respect to longevity of digital information. Hardware and software is to a certain amount connected, at least hardware and operating systems. Because of the extremely fast speed of hardware innovations, computer systems are considered "old" and practically obsolete after a period of about 5 years (currently). The reason mainly lies is the changes on the software (operating system) side. The consequence is, that outdated hardware has to be removed (this side of the problem is not discussed here) but more important for these considerations are the fact that information still might be stored on those devices: This creates (at least) the subsequent problems:

1. The outdated system can be in use, hence someone is responsible for it, or the system is simply turned down by someone and was "forgotten".
2. Data will be stored on internal storage devices like *harddisks*.
3. This data might need *particular applications* to handle.
4. Parts of the data will be considered important for future use, others not.

To the first point: if the system is removed from service in a "professional" context, the following points will be considered. In the case, that the system is a left-over by some former employee or found in a heritage . . . the system may be old, the passwords unknown, the know-how how to handle the system unavailable. We might consider a VAX/VMS system in 2025, found in some lab arguing that important data and/or applications are stored on it. I truly believe, that it will be very difficult to find someone who is able to access the system appropriately (if it is physically in order!).

This might be a *worst case scenario* but it is important to see the range of the problem. Even considering a professional scenario, where data is migrated from older to newer systems, problems occur: old applications might not run on new hard and software; this problem sometimes requires huge efforts in migrating the data to new formats. This is especially problematic when proprietary applications with proprietary

Medium	practical physical lifetime	avg. time until obsolete
magnetic disk	5-10 years	5 years
digital tape	2-30 years	5 years
optical (CD)	5-59 years	5 years

Table 7.1.: The practical physical lifetime and time until media become obsolete (taken out of [90])

data formats are used. Especially older software often does not support standardized interfaces.

Even under those "professional" circumstances, where continuous migration is done, there is one significant drawback: often it will not be possible, or not considered as necessary to migrate the complete data from the original system to the new target system. Because of the cost-factor, only data regarded as essential at this instant will be processed. This means, that some technician or employee decides which data is required and which data might be obsolete (possibly this data will be backuped, but this only postpones the problem for some years into the future). The consequence is, that useful data for future use might be destroyed. It is important to note here, that this *destroying* of data is *not* an explicit one! So it is not as people were used to: papers and books are evaluated, and if found outdated or unimportant those books or papers were dumped. Nowadays we detect the inverse scenario: data has to be *explicitly selected* for survival! This is the other way round: Up to now, energy was necessary to remove information, now energy has to be spent to preserve this information. It seems to be clear, what the consequences of this inversion are and will be.

7.3. Archival and Migration

> "Digital documents [. . .] have the discouraging characteristic of being software-dependent.
>
> [. . .]
>
> the translation approach suffers from a fatal flaw. Unlike ancient Greek and English, which have roughly equivalent expressive power and semantics, digital documents are still evolving so rapidly that periodic paradigm shifts are inevitable. And new paradigms do not always subsume their predecessors: they represent revolutionary changes in what we mean by documents." *Jeff Rothenberg [90]*

7.3.1. The Medium

Looking back in the history of mankind, we notice, that these kind of problems are completely new. Up to now, information was written down in stone, papyrus, paper, books and so on. This information has been accessible for multiple thousands of years. The advent of new technologies brought new problems. It started already with media like audio tapes or film; however, those technologies are yet simple enough to get information from those sources with rather simple devices. But already the film example shows, that the media does not survive for more then few decades! In this context film is irrelevant, but other storage media are not: table 7.1 illustrates the lifetime of some important currently used media.

7.3.2. Migration of Digital Information

One could argue, that especially the example of the lifetime of specific media like film illustrates the advantages of digital information, as copying of digital data is possible without loss of information! This seems to be true, at least from a low-level technical point of view. But problems show up immediately. Besides the already mentioned problems problems in "keeping old data alive", there are even more substatial questions:

1. What is a backup? A "simple" backup of the digital byte-stream?
2. What is a copy? When is a copy or a backup similar to the original?
3. What is the backup medium?

To start with the first question: Digital information usually is closely connected with hardware and software. Consider a text document produced with a word processor in 2003. This document contains a lot of information: Text, other elements like images, sound, hyperlinks, marks and notes by different users, format commands, paper sizes and setup, document history and so forth. What is now understood when the term *backup* is used? Usually *backup* is a strategy to ensure security against hard or software failure and means a simple binary copy of the data bistream. This is a well-suited strategy for the case, that the necessary hard and software are still available for restoring the data.

This strategy is obviously highly unsuited to keep digital information for years, decades or even longer by many reasons: first of all, the required hard and software to read the backuped files will not be available any longer; eventually not even the hardware to read the backup media will be functional. Having a look at table 7.1, with a high probability the backup medium itself will be unusable.

An alternative to this strategy was already mentioned in the previous section: The documents and data need to be migrated. But this sounds far simpler than it is as the second question illustrates: What is a copy? To migrate data means to make a copy of the information into a format that can be used on the new system(s). But what is a copy [58]? Considering the example above: The text of the document? The text including the images, including the formatting commands; and what about the change history of the documents. . . . Practice shows, that it us usually never possible — even with digital data — to make an exact copy when migrating data. As the information often is a combination of the data *together with* the appropriate application that uses logic to interpret the data. The problem increases when the need to export and migrate data in future is not originally planned with the application. A lot of such examples are existing (files from outdated word processors, spreadsheet applications, databases. . .) As a rule of thumb: The older the data is, the more is lost in migration (except when data is kept in very simple formats like ASCII text-files maybe also enriched with metadata, this will be analyzed in detail in the next sections). So the consequence is, that one (at best the author) has to decide what is worth to be copied and should *stay alive* when migrating the data. Ideally, the system should be prepared for migration already from the beginning[2]!

7.3.3. Alternatives to Data Migration

As migration of data has several problems as mentioned above, several research projects tried to find alternatives. Essentially two main strategies were described: (1) Emulation of Hard and Software on new systems and (2) a clean meta-description of the currently available system so that the concrete implementation is only a matter of a fitting virtual machine (as I would call it) for a new hard and software.

First of all, the latter proposition will be discussed, because already a brief view shows that it is practically useless. Raymond Lorie from the IBM research center Almaden proposes a so called Universal Virtual Computer (UVC) [61]; an interpreter could then be written for any target hardware platform. He concludes his article with the following points:

> "1) In 2000, for each platform, the manufacturer needs to provide an emulator of M2000 written as UVC code. Manufacturers of devices in 2000 need to provide the UVC code that emulates the device control unit.
>
> 2) In 2100, every machine manufacturer needs to produce a UVC interpreter, and every manufacturer of an I/O device needs to produce an implementation of the abstract device on the real 2100 device." *Raymond A. Lorie [61]*

[2] Meanwhile several standards and recommendations are approaching. E.g., the *Reference Model for an Open Archival Information System (OAIS)* [69] developed in the domain of *space data systems*.

These two propositions are already the main reason, why this strategy will never happen in the near future. There is no producer in sight (neither hard- nor software) that is planing anything like this. Moreover: to be successful, this would have to be a joint effort of the majority of vendors. And finally: such a system is basically already available: the *Java virtual machine*. So why should anybody (except Microsoft) make a copy of an already working platform independent virtual machine system? And although such a VM is already available it is not really a solution for our problem, as there are many detail problems that are still not solved: Even if everyone would produce only Java applications: there would have to be a guaranteed continuity of the JVM for the next decades. Moreover also applications designed for a *virtual machine* are not running in *virtual space*: they are generated for a *specific* domain, e.g. a web-application, or a desktop GUI application and integrate with other applications like webserver, databases, and so on. Who knows how the computer systems in 50 years will look like and if those applications would make any sense then, even though they are developed for a virtual machine. This, besides the fact, that not every application is programmed in Java and a UVC is not in sight and this is no solution for already available systems, we clearly need other strategies.

So the first point initially mentioned is emulation. In fact many researchers and companies are working on system emulations, and so there are emulators available for many old systems like the C64, Atari Systems or DOS, which allow to run old applications and access old data. At least theoretically. Those systems will be valuable in future, but to be clear: they will definitely be *no solution* to guarantee the longevity of digital information, particularly not of scientific or cultural heritage data. The reasons are simple:

- Even if an outdated system could be emulated precisely with a new computer, will there be an expert in 50, 100 or 200 years who knows how to handle this system? Most certainly not.
- Even if the emulation is available *and* an expert exists who is able to handle the system: has the data been copied to up-to-date media? It is unbelievable, that a C64 disk will be readable in 50 years.
- It is improbable that the emulation technology will be continuously developed for generations, and if not, a highly complex avalanche of emulations would have to be initiated: e.g., an emulator that emulates a Windows PC to start the C64 emulation inside the emulation and so on[3].

So the lesson learned is, that emulation software is useful, if the users forgot to migrate the data and no running hardware is available. Then the data has to be

[3] And you will need a windows expert to install windows and to install the C64 emulation and another C64 expert that knows how to handle that system and so on. . .

migrated immediately using the emulation software, as long as the data media as well as the knowledge to handle the system are still accessible. So the emulation enlarges the accessibility gap to older systems for, say, five to ten years, but are definitely no long-term solution. Hence the core strategy has to be a planned migration strategy as described in the last section.

7.3.4. Mission Critical (Continuous) Data

Mission critical data and data used continuously is usually no significant problem: Databases as well as client applications are permanently kept up-to-date and migration happens seamlessly. Problems occur, when bigger changes (especially in hardware and infrastructure) are to do. Nevertheless it might be very reasonable — also for mission critical systems — to follow the suggestions made in section 6.7 as well as in section 7.6; maybe by even other reasons: Using open systems, well documented data (exchange) formats and clean interfaces instead of proprietary and "secret" systems allows more flexibility for the customer in selecting vendors for new systems as well as more partners for support and training become available instead of expensive system specific support.

7.4. World Knowledge Heritage or The Lost Knowledge?

> "We risk a dark age where it is impossible to reliably state what occurred or why because the information that documents the business has been lost."
> *Waugh et.al. [126]*

In the *systems* chapter (see page 57) the importance of knowledge in the society (since the third industrial revolution) has been pointed out. Knowledge is based on information which is based on data. So the circle closes again: data, especially scientific research (published as well as unpublished!) and cultural artifacts were in a form, that lasted usually with no problems for centuries. Materials like stone and paper offer no specific problems in maintenance, the conservation of paper is well understood and mainly under control. But in the last century the materials changed dramatically: starting with the paper quality which is in some cases far worse than the paper quality in the 19's century, but most problems arise with the new materials like film, audio tapes, optical disks...

First of all these materials are not usable *as is*; they need specific machines or logic to be useful (Figure 7.1 illustrated the processing from media to information). Also the durability of those new materials are very limited. Only few know — to give an example — that classic movies from the 1950's, and 1960's are running into problems because of the properties of the film material. To give an example: the recovery of Alfred Hitchcock's *Vertigo* took multiple persons for more than a year and cost about

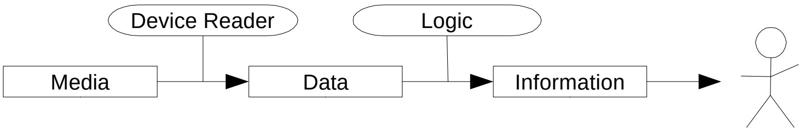


Figure 7.1.: Processing steps from data-medium to information. For sake of completeness it has to be noted, that the audience has to fulfil specific requirements too. E.g., a “cultural” background may be necessary as well as the capability to understand a specific language and so on.

one million U.S. dollar. How many other movies will be lost in the next decades, because no one has enough money and interest to recover them?

Additionally also the risk of loosing huge amounts of data is increasing permanently. When data was stored on paper, the loosing or destroying of a few books might have been unpleasant and even a real problem. But today harddisks are available with a storage capacity of 200 gigabytes and more, and the terabyte media will be available soon. When no well-planned backup is done, the damage of one single medium — a terabyte disk for example — might destroy the value of a complete library!

So it’s even worse with digital media: On the one hand, copying of digital media is basically possible without loss, but the durability of the media is maybe restricted to decades. It is questionable if DVD players will exist for more than 20 years. What about the movie (DVDs) or music CDs bought today? But there is not only a media problem: As the media companies will not spend a lot of money to rescue music and films that were not successful, the problem gets even worse with recent legal actions like the *digital millennium copyright act* (DMCA) and the encryption and copyright protection of DVDs and audio CDs. I wonder how this could be accepted by politicians: Customers buy films on DVD and have no guarantee, that these movies can be played in ten years, because the customer officially has no possibility to migrate the *legally bought digital information* to a new media! A solution for this problem would have been required to protect the rights of honest persons buying their media and not copying them from the Internet or from other illegal sources!

The same problems can be detected in the scientific research community: In contrast to the Anglo-American tradition, in Europe as well as Asia, knowledge never was treated as a product to bring some companies big revenues; with knowledge becoming a resource, more and more scientific results tend to become the property of a firm, of a commercial database or a patent and it will most probably lead to big problems for the capability to scientific innovation. Recent announcements and activities show, that media and science companies try to protect their data using complex encryption techniques were even special hardware is required (TCPA, Palladium). To be clear:

this is not only a *science issue*, it is also a *cultural issue*: questionable copyright laws, complex encrypted data formats etc. bring problems in terms of high-quality teaching (knowledge transfer), spreading of literature, film and paintings and even worse, it will pose huge problems in terms of longevity of digital information, as the logic steps to decrypt and transform the data to information (as outlined in figure 7.1) become very complex and intransparent.

All those new (mainly imported) tendencies pose huge threats to the free development of societies, science and democracy. Now, even beyond all negative effects on economy, we can observe the paradox situation, that cultural heritage, history of science, in short: *the human knowledge*, that was accessible for everyone, might become a closed, restricted and commercial resource. The situation is paradox, because the rise of information technology promised to be a factor to allow access to recent knowledge on a broad basis for "everyone"[4]. Current activities as described above may destroy this freedom of information and knowledge with benefit for everyone, and create an ever closer circle of those who *have*, opposing and controlling those who *have not*. Not to forget the developing countries, that come more and more into unnecessary dependencies because of patent rights, closed knowledge bases and closed science[5]. This is, of course, very much a political issue, but I am afraid, that politicians with a sense of responsibility still are unaware of those described consequences!

7.5. Characteristics of Project Related Data

The importance of longevity of digital information in different application domains have been illustrated above. But project related data and information poses specific problems that should be illustrated with an example (following the ideas and concepts of OSWP, see chapter 13): Project data is highly interconnected: There are users, projects and tasks. Tasks are related to projects and tasks (in a tree-like data structure) and to users (who are responsible for specific projects or tasks). Additionally other information is available like different types of resources (files, URLs, databases) or "news articles" — both are again related to users and tasks/projects and so on. Parts of the data are hard to interpret without additional logic (like database resources, XML resources with connected stylesheets . . .). A potential user of this data will be confused if he has no additional application logic to work with this data. The general problems are similar to the problems of cultural and scientific data described earlier, but with the complication, that project related data is a bunch of different

[4] At least for everyone with the necessary technological equipment, which is already restricting enough, considering the fact, that a stable information technology like the Internet is still not available in developing countries on a broad basis, and most probably will not be during the next decade.

[5] Those findings were also a basis for the development of the German literature and language projects in cooperation with the University of Salzburg (for details see chapter 12).

structures, types, access and storage methods and all of this highly dependent.

However, the need to give access to project data in a format that is not primarily dependent on a particular application seems important, yet crucial. Many reasons can be given: Rus et.al point out the value of a so called *post mortem analysis* of projects [91] for project and knowledge management. On the one hand, to learn from failure, and on the other hand to extract information to be reused in new projects. Especially the latter function seems to be a little bit neglected in (university) research:

> "At a higher level, organizations and industries must analyze multiple past projects to improve their software developing abilities." *Rus et.al. [91]*

Of course, also in the industrial domain post mortem analysis can be of highest value. To give an example: IBM encourages its employees to write brief "lessons learned" reports after a project is finished into a specific database. If new projects are started, project managers should browse this database searching for similar projects already done with the idea to avoid typical pitfalls in the project domain.

So the problem is, that on the one hand, *open access* to the project data is highly appreciated, on the other hand the data *as is* might be hard to use because of the mentioned ontological problems. To find a simple solution to this problem is hardly possible, but some suggestions will be made here (in a more general/theoretical way and in a practical part in the concrete implementation of SWP and OSWP, where some of the suggestion made have been implemented and tested — see chapter 10 and 13).

Another difficult problem is data from communication channels. In every project cooperation different (digital) communication channels are used, like email, maybe discussion forums, chat and instant messengers. The data exchanged through those channels might not be a primary information source, as the results usually are concluded in reports and documentation, but nevertheless (in case of problems) they might offer an extremely valuable resource. One could consider the case, when one project partner is leaving and is replaced by a new one. It would be of great help for the new employee to be able to trace back certain developments and decisions.

Unfortunately the current situation is problematic as all those mentioned systems often use different systems like Internet email, NNTP or web-based discussion, Yahoo or ICQ instant messenger, some Java chat applet. Integration of this data is very hard. It is highly recommended to be aware of this fact, when a (corporate) communication system is planned.

Those observations lead to the consequence, that we tried to implement a unified access to the most important communication features into the OSWP system. This leverages the use, and should be the first step to a homogeneous data basis also for communication data. Nevertheless it is not easy to find a useful solution. E.g., the email system was considered to be reimplemented in OSWP, but this would have the disadvantage, that users do not like to use new systems, when their available (email) system is well functioning and useful. Implementation of such functionality is often walking on a thin line between good integration and bad usability. So other strategies were taken and will be explained in detail in the third part of the thesis.

7.6. Policy and Suggested Solutions

7.6.1. General Considerations

The problem of longevity of digital information is not only and may be not primarily a technical one, it has: "three aspects: *physical conservation*, *functional preservation* and *organizational preservation*" [126, 63]! A clear concept has to be worked out, at best by a specific task force to develop a strategy for handling data, information and knowledge. This complex problem has already been detected for at least a decade; specific commissions have been formed and strategy papers have been written. E.g., in 1995 the *Task Force on Archiving of Digital Information* commissioned by *The Commission on Preservation and Access and The Research Libraries Group* (United States) [57] or a study for Australia [126] performed an extensive analysis to the problem. Among others, the commission "envisions the development of a national system of digital archives, which it defines as repositories of digital information that are collectively responsible for the long-term accessibility of the nation's social, economic, cultural and intellectual heritage instantiated in digital form." As one could expect, also libraries realized that the current situation posses a lot of potential risks for the scientific and cultural heritage, as described in Barkstrom et.al. [11].

So this problem is well known for years, but it seems, that short-time innovations have a higher impact then a clear information management strategy that goes beyond a simple backup. Waugh et.al. [126] names the cornerstones of long term preservation of digital information in five points:

> "- Encapsulation; that is, wrapping the information to be preserved within descriptive metadata and keeping it at a single location.
>
> - Self documentation; that is, the ability to understand and decode the preserved information without reference to external documentation.
>
> - Self sufficiency; that is, the minimization of dependencies on systems, data, or documentation.
>
> - Content documentation; that is, the ability of a future user to find or implement software to view the preserved information.
>
> - Organization preservation; that is, the ability to store information that allows an organization to actually use the preserved information." *Waugh et.al. [126]*

The technical parts of those suggestions (encapsulation, self documentation, self sufficiency and content documentation) will be discussed in details in chapter 8, including the use of open formats and metadata like XML (extensible stylesheet language) as well as metadata specifications like *resource description framework* (RDF) or *topic maps*. This allows to store information with meta-information. However the problem persists, that also the meta-information might not be clear or unique. Hence additional research activities are done trying to define more specific ontologies. A brief overview to recent standardization efforts in the research area of ontologies will be given (like the *semantic web*). Additionally it is recommended by many reasons, to use open operating systems and open source applications were available (for details see chapter 6)

One additional comment has to be given, particularly to data management according to project related data: All activities for long-term data preservation must be started at the very *beginning* of the project, thus has to be part of the project and the project plan and finances. This is very important as after a project is finished, often no additional money is available to pay the additional activities to preserve valuable knowledge. Particularly in the academic field where personal fluctuation is very high by definition, the responsible person(s) for the project might not be available any longer. So it is clear, that all archiving steps have to be part of the initial planning.

7.6.2. Political Activities

As Waugh et.al. [126] point out, migration of data is always a significant challenge and might even break the cardinal rule of preservation: *minimize harm*. As migration always means: data modification, information degradation and even worse: it might subsequently not be clear what has been lost after migration! Hence this necessary future migration step should be an integral part in each new IT installation and part of policies, e.g., in software installations in the public sector. By adding this activity into the requirements for new software, a necessary pressure is put onto vendors to consider the problems mentioned here. And no one has a better insight into data and functionality then the developer during the development cycle.

The importance of open standards has already been pointed out on multiple locations, but this is also an issue for political activities. First of all, in the public sector only systems should be used that fullfil certain minimum requirements with regard to long term preservation of data. Secondly, as a consequence of this, politics could increase the support of open and respected standards for information interchange as well as for information preservation. This is an urgent issue not only for data in the public sector, but also for universities, health care, insurances, statistics and so on.

8. Information Management

8.1. Information Abstracts Data

8.1.1. Degrees of Abstraction

The definition of information given in the first part of the thesis (see page 41) was a rather general one and is the foundation for the following thoughts: As information is *relevant data*, decided by any *operational system* information can also be seen as the first degree of abstraction of data, the second degree of abstraction would then be knowledge.

Abstraction means generalization, reducing the not relevant aspects, where some operational system defines the condition of relevancy. At this place the circle is closed again. To give an example: A data collection is: names, addresses, income of all people of one town. *One* possible chunk of information *could* be the finding, that the distribution of wealth is not homogeneous, but is concentrated at specific places in the town.

Why is this concept important? Information management has to start with *data management* in a way, that *flexible information abstractions* are easily possible. Ideally with not too many a priori restrictions of the domain, as it is never known in what context data might be used in the future. Following the example above: The data should be organized in a way, that allows easy access for all possible types of information generation processes, even such not known at the moment of implementation.

In reality, such an optimal way to manage data does not exist. Several compromises have to be accepted, depending on the type, order and structure of data and information. Those aspects will be analyzed in detail in this chapter. A basic consideration shows systems with different degrees of abstraction in terms of information-generation out of data. In Table 8.1 real-world-systems are categorized in systems of first order, second and third order abstraction. This categorization is not only interesting from a theoretical point of view, but has very practical implications (as the table illustrates) with regard to access, generation, reproduction, transmission, longevity, searching and archiving of data and information. To make this concept clear examples should be given:

	First Order	**Second Order**	**Third Order**
Access	direct	depends, usually additional transparent steps required	special decoding required
Generation	direct	using transparent tools	using special encoding machines
Reproduction	usually difficult loosing quality	not too difficult, loosing quality	Easy usually without loss of quality
Transmission	difficult/impossible	difficult, loosing quality	Easy without loss of quality
Longevity	depends on material, evt. very long	depends on material, usually medium time (decades)	depends on data storage and decoding machines, typically only years
Searching	difficult	difficult	easy
Archiving	difficult	depends on medium, medium	see longevity

Table 8.1.: Information Properties according to Order of Information. Note that this table has to be understood as qualitative, not as normative expression. The values are meant as “typical” properties.

8.1.2. First Order Information

A first order data/information system could be a stone-painting, a conventional book or a conversation (a speech of a person). Access is directly possible, reproduction, transmission, searching and so on are depending on the “object”, but usually offer some problems[1].

We are surrounded by systems of first order, and knowledge as well as states, administration, science, . . . were based on those systems and still are.

8.1.3. Second Order Information

Systems of second order usually need some *additional treatment* to allow significant information to be abstracted: Examples could be: a conventional (analog) film, a record (not a compact disk!), a microfiche and so on. But it is important to see,

[1] It is here to note, that the term *direct access* is not unproblematic. It assumes, that every human is able to abstract information from the data. This *might* be possible in the case of a simple stone painting, but not necessarily in the case of a book (See also Weizenbaum’s critic in first part on page 42). This is obviously depending on the language and content of the book. However, to keep it simple, it is assumed as direct access, when *there are people*, that are able to access it directly.

that access to the information is not too complex, and the method needed is pretty transparent to any well-educated human.

Dealing with systems of first and second order will not be a main topic in this thesis, though it is essential to keep them in mind when generating a project related knowledge and information repository! The “solution” from the technological point of view is rather easy as nearly all first order systems can be migrated to second order systems, and second order systems to third order systems.

8.1.4. Third Order Information — Some Consequences

Systems of third order are systems where specific encoding/decoding steps with probably massive knowledge and/or technology are necessary to allow information abstraction. Examples could be a compact disk (CD) or a digital versatile disk (DVD), a computer hard-disk containing a database, a message encoded with an Enigma machine, or any other encryption process.

As mentioned above, all first and second order systems can be migrated (even with some loss of information) to third order systems, so this thesis and this chapter in particular will focus on systems of this type. But also among third order systems we observe differences: data can be managed in open formats like docbook (XML) or in proprietary and complex formats like a computer game, that requires a special computer hardware and software setup, where it is hardly possible to migrate this to any other context.

So in fact the classification provided here is a simplification: from *stone-paintings* to *encrypted DVD* content, there is a continuous increase in terms of abstraction. Hence the definition of three orders is somewhat ad-hoc, but is useful to categorize and analyze data/information systems. Yet it is important to see the consequences of this view toward data and information. In everyday’s life and work, we are confronted with increasing higher orders of information. Blessing or danger, this is the question hardly asked: On the one hand, as a consequence of this “abstraction revolution”, dramatic increases in productivity can be observed. On the other hand, many short-termed productivity gains will be paid on future cost.

Very often in this thesis I discuss the problem of too abstract representation (Longevity of Digital information: see chapter 7, Systems: see chapter 6, knowledge management see chapter 9 and in this chapter). Hence the suggestions given here should help to setup ICT for long-term productivity gains, and to avoid loosing information and knowledge non-retrievable.

8.2. Structure

8.2.1. Highly Structured Information

Highly structured information (HSI) is information as defined in section 4.2.1 on page 44. The management and handling of HSI can be seen as non-problematic; it is solved from the theoretical point of view with the *relational data model* and with schema-based XML formats, as well as from the practical point of view (see, e.g., [24])

A huge amount of relational database management systems are available, open source as well as commercial systems. The maturity of the mainly used ones (Oracle, DB2, PostgreSQL, mySQL, Interbase) is very high as they are on the market for at least a decade. Access is possible with all relevant applications and program languages using different driver systems (Java: JDBC, Windows: ODBC, native protocols . . .)

So if possible, HSI should be treated following the relational model and stored in relational database systems (RDBMS). Nevertheless there is a potential drawback using RDBM systems: Although there are various standards (SQL 92, 99) [26] the mentioned database systems implement those standards only partly and add different proprietary functionalities. Starting with particular SQL commands, different data types, different languages for stored procedures and others. The consequence is, that an application that uses the specific functionality of a RDBM system is highly dependent on this product. Moreover it becomes a difficult task to extract the data out of the system, as parts of the meta-information like relations, foreign keys, triggers, checks might get lost.

As a matter of fact this might not always be the best solution as there are sometimes good reasons using the high level functionality of the systems. From the data/information point of view, it seems to be a good idea to plan (from the beginning) some kind of “exit-strategy” that allows to access the data in the RDBM system in a platform neutral way, e.g., as well documented XML data, maybe also by adding webservice interfaces (more details to those data exchange problems will be given later in this chapter).

As a consequence of those problems it is often suggested to avoid using high-level RDBM system functions, particularly things like stored procedures, triggers. It is clear: as soon as an application depends on database stored procedures or triggers, the application is no longer portable and moreover also the access to the data may become critical. So more and more middleware products like application server (J2EE server) and object relational mapping tools rebuild the high level functionality and use RDBM systems only as data-stores.

Of course structured data formats like SGML or XML (XML databases) can also be used directly as data store; but in many cases it is more comfortable to use the RDBM systems as performance, multi-user access, scaling, clustering, transactions, synchronisation and much more is already a reliable part of the system and can instantly be used. In the case of the SWP and OSWP project, application server were used (Enhydra, JBoss with OJB) and those application server and object relational mapping tools respectively, use only basic functionality of the RBDMS engine and offer higher level functionality through the application server API. Hence SWP as well as OSWP can be used on nearly any RDBM system (Interbase/Firebird and PostgreSQL/mySQL were used and tested).

For data management in the literature projects [99] a content management system was developed that was capable to store different types of information (articles, lexical information, multimedia content, ...)[2]. The data model is rather difficult to understand as it consists (as a consequence of normalization steps) of many entities as well as of complex relations, and foreign keys between those entities. This normalized model is very useful for the implementation of the content management system, but problematic for persistent data storage as mentioned above. Even a platform neutral database export (e.g., as comma separated values or the like) would be problematic as the complex relations and the meta-information necessary to understand the data (and rebuild the information) are difficult to document, and even more difficult to rebuild without the initial knowledge of the domain.

As a consequence, the content management system allows to export the complete information from the normalized RDBM model to denormalized XML documents[3]. Those documents are better suited for long-term preservation of information as will be explained in the next sections. This is particularly relevant as in those projects valuable cultural information was collected which has a far longer lifetime and far greater importance than information in the IT domain.

8.2.2. Semi-Structured Information

8.2.2.1. History of Standards and Formats

The term semi-structured information was defined in section 4.2.2. Nevertheless it is necessary to remark here, that this categorization between structured, semi- and unstructured information is blurred, and in many practical cases one has to decide what kind of model and tools should be used for a specific problem. This unclear situation is particularly existing between structured and semi-structured data. However, the categories as outlined here, should demonstrate the "ideal" conditions for using a specific model and specific tools.

[2] The system is described in more details in part 3 and in the cited scientific papers.

[3] This is done in the end, as a final step of the project, and this "future proof" data should be archived properly!

From a pragmatic point of view there are several typical problems where semi-structured model and tools are appropriate:

- *Document oriented information*: That means, the document schema can be precisely defined, but the sequence of elements is not predetermined. A typical example of such document oriented schemes is the XHTML definition [131] or the Docbook specification [32].
- *Hierarchical Information*: This is a very important case. Hierarchical (tree-like) information like the content of a file system, configuration files of complex applications, a pedigree or an organigram is easy to map to an XML or SGML structure, but does not naturally fit into the relational model.
- Information with *regularly changing structure*: This is typically the case when data is acquired where either the structure of the information is complex or it is not known at the beginning therefor one does not want to "squeeze" the data into a too close schema.
- Documents with *"irregular structure"*. Those are documents that need not even have a precise document definition. Nevertheless the semi-structured information stored is clearly structured and enriched with metadata.
- A special type of information, that should be called semi-structured is *application logic* and *program source code*!

Seen from a "historical" point of view the situation of storage and handling of semi-structured information was very inhomogeneous. In the Microsoft DOS/Windows dominated areas different types of (mostly binary) proprietary file formats were used. Nearly every application and system used its specific data-formats, and all of those data formats were hardly documented and data exchange difficult — if possible at all. Situation became a little better with more recent Windows versions as Microsoft tried to unify data exchange between applications using clipboard formats and *object linking and embedding* (OLE) technology. Besides the fact, that OLE was limited in functionality and had a lot of bugs (hence was hardly usable for a long period of time), it had a more serious drawback: For complete functionality, embedded OLE objects need the host application of the embedded object installed on the machine (ideally in the correct program version). If this is not the case, only a limited access to the data of the embedded object is available. Windows clipboard formats on the other hand are used only for data exchange during work with the applications as both sides of the data exchange parties (source and target applications) have to be installed on the system. Consequently (Windows) clipboard and OLE are useful for some tasks, but do not really leverage the problem, that semi-structured data was and still is scattered in various bad documented binary documents.

On the other hand, on the Unix (and partly OS/390 Mainframe, OS/400 ...) side the situation was slightly better as data formats were usually text-based. So at least access to the data was easier, whereas access to the information was not always easy, as data without meta-information and other guidelines is often hard to understand. Moreover, even if data formats are text-based specific parsers have to be written to access data.

In the 1980's a new standard was developed to solve the mentioned problems: the *Standard Generalized Markup Language* (SGML). SGML (and other suggestions like HyTime in the 1990's) had some impact in specific domains: SGML was successful mainly in the document management/publishing field, where also good tools like Framemaker/SGML and others existed. Yet SGML was not really successful in other applications. The reason was mainly, that (as so often) the developers of SGML wanted to achieve too much and generated a too formal model. As a consequence, SGML was hard to learn, hard to implement and hard to use. The same is true for SGML related technologies like the stylesheet language DSSSL.

Hence in the second half of the 1990's an SGML related technology (see ISO Standards [49]), the extensible markup language (XML) was proposed [133]. XML succeeded by many reasons: first of all — on a basic level — XML is very simple and easy to understand. Moreover many developers were fed up with the history of HTML which started as specific SGML application as a standard for document exchange and developed in a "wild-west" manner by two companies fighting for market shares. XML was clean from the beginning and even though Microsoft tried to implement the *embrace and extend strategy* also to XML they luckily failed up to now. So XML was the right specification at the right time and the right place (World Wide Web consortium [125]).

Still there is to note, that meanwhile XML is far more complex than SGML, at least considering XML and all related specifications (XSLT, XSL:FO, XLink, DSSSL, XPointer, XHTML, Topic Maps, Webservices, Namespaces, XPath, XQuery, ... to mention just a few). But the advantage is that XML and related specifications can be seen as modules or like parts in a construction set where everyone learns and uses just the components needed to solve a particular problem.

As a consequence of the success of the XML specification and related technologies many application developers (in the open source as well as in the commercial field) started to modify their systems to work with data formats based on XML. This makes access to such data easier as XML is human readable and metadata is included in the data format. But equally important is the fact that a huge amount of XML tools emerged in the last years like XML editors, XML parsers for all relevant programming languages, transformation languages and processors and many more. This makes access to XML related data again easier as it makes no principal difference whether an *Open Office* document or an *Apache configuration* file is parsed.

Besides the mentioned formats also other systems established: Object oriented programming generated (memory) storage mechanism in the form of objects that can be seen as semi-structured data too. Clearly persistence mechanisms for this kind of in-memory-objects were needed. Those systems will be described and analyzed in the next section.

And finally it should not be forgotten, that maybe the greatest part of semi-structured information is not yet available in digital form but written down in *natural language text* like books, notes, documents and so on (this would be information of first order as categorized above). This is also true for natural language text that is digital, but not structured with machine-readable meta-information (like emails, scanned books, web-documents without metadata and so on). Retrieving machine-readable and understandable information is a highly complex task, mainly caused by semantic problems. Even documents of one specific domain are very hard to analyze and process. Problems in this field of research are not part of this thesis, but are investigated, e.g., by Mattox et.al. in [66]

8.2.2.2. Storage Mechanisms

File Based The "simplest" solution for the storage/persistence problem is of course storing the (XML) data in a file of a file-system. This is the usual methodology for all kinds of *document based* data that is (write) accessed only by one person (at least at a time). Moreover this way of storage is perfectly suited for backup mechanisms. In case of multiuser (write-) access, very large documents and specific needs (like indexing, querying and so on) a file based storage mechanism is often not the ideal solution. There are solutions for these applications like concurrent versioning system(s) (see also section 8.2.2.2).

Additionally, also program source code can be seen as semi-structured information. This kind of data is usually managed file-based. This need not always be the case, though (particularly in very recent systems): code generation on the basis of XML descriptors or produced by UML (unified modeling language) modeling tools like Rational Rose have to be taken into consideration.

Object Oriented Databases As mentioned above, there is an affinity between *object oriented* and *semi structured* data (XML). Hence the idea is to think about using an OODBM system for storage of semi-structured information and data. In fact this is hardly done in practice. There are many reasons for that: First of all, there are hardly any OODBMS available. In the 1990's with upcoming OOP, it seemed to be a good idea to develop OODBM systems. It turned out not to be such a successful idea for the everyday business. One major drawback often was, that a database that stores objects (in a kind of serialization) is usually strongly bound to that specific application. A flexible usage of the database for different purposes is difficult.

The relational model on the other hand is a flexible model rather independent of a concrete application (this is also a disadvantage in using it with OOP languages). However, it turned out, that users and developers did obviously not like OODBM systems too much, moreover the RDBM systems were dominant already at that time and extended their engines with OO features, and different open source projects and commercial products dealt with the problem of OO/RDBMS mapping. Considering semi-structured data and XML new competitors are on the way: *native XML databases*, as will be illustrated later.

In short OODBM systems seem to be dead in mainstream applications, but are still used in specific niche domains (like embedded systems). Hence many OODBMS are already seen as legacy systems and their importance as storage mechanism for semi-structured data and information is not really relevant.

Relational Database Management Systems As mentioned earlier, RDBM systems were originally not designed to store semi-structured data. Nevertheless all major database vendors developed extensions to their systems that allow to access a broad range of data types including XML data, multimedia data and so on. As an example: IBM's DB2 extender allows to store XML data in a specific XML data field in a regular table, or to define a mapping between arbitrary XML structures and database tables. Oracle databases on the other hand offer the functionality to access data in form of tables (using JDBC and other drivers), via ftp or WebDAV "file access", as webservice, as XML and so on.

Also in the scientific community [28] and the open source scene many products were launched like OJB (an object/relational mapping tool), the XML database Apache Xindice announced functions to integrate RDBMS and XML data and Apache Cocoon allows to integrate XML data and conventional RDBM databases. Many more projects are on the way.

The advantage using a conventional database system to store semi-structured data is the same as described above comparing RDBM to OODBM systems: flexibility. As soon as data is stored in the RDBMS, it can be accessed using "conventional" applications using interfaces like ODBC, and at the same time application servers like JBoss, Websphere or publication frameworks like Cocoon can use and work with the data.

Moreover systems like DB2, Oracle, SAPDB or PostgreSQL are very mature and stable and offer many tools for administration, multiuser access, backup, import, export and so forth.

Native XML Databases A rather new option in storing semi-structured data are native XML databases [95]. There are cases, where the data is best stored directly in XML. Using a file system has the mentioned disadvantages, like lack of multiuser capabilities, transaction safety, and so on[4] Thus a *database engine* is advantageous, but a relational database system might be not ideally suited.

Recently native XML database systems came up, in the commercial domain (Software AG: Tamino) as well as in the open source scene (Apache Xindice, eXist) and scientific research is done extensively, e.g., [4, 124, 95]. Using such a system offers the advantages to have multi user access, indexing mechanisms, different APIs (Java, Perl, Webservices, . . .), querying mechanisms like XPath, XQuery, standards for data updates (XUpdate) and so on. So this approach is particularly popular in the field of web or cross-publishing problems and web applications.

Future will show whether native XML databases will find a market *beside* conventional RDBMS database products, or if new *XMLized* RDBM systems will take over this market share as it was done in the OODBM market.

“Special” Systems Besides the mentioned storage strategies there are data storage and management tools that do not belong to any of the systems described above. For special data like program source code applications for managing these sources have been developed: most popular examples are CVS (concurrent versioning system) in the open source domain or Microsoft’s source safe. Those tools were developed to manage different versions of applications and parts of applications as well as to organize code between larger groups of developers. As systems like CVS work (essentially) on text-based documents, the consequence is, that one can use CVS systems not only for application sources, but for all (hierarchically) stored file repositories. Examples are: management of websites or XML/XSL documents. This option is often forgotten, when system decisions are done. Using CVS has many advantages, particularly when versioning and platform independence is an issue.

Finally it should be mentioned, that this storage system description should be seen as a rough analysis of possible solutions, as there exist several other storage mechanisms not explicitly mentioned here (like directory services, e.g., LDAP based and so on).

8.2.2.3. Management

As mentioned already, management of semi-structured information poses more problems than management of highly structured information. The reason is clear: Although standards like SGML and XML allow the creation of tools that access every type of SGML or XML document from the syntactic point of view, the semantics and structure of the documents are for more variable then the structure of a database table. This is clear consequence, as a higher flexibility automatically creates a higher degree of freedom.

[4] Database theory calls those factors the *ACID* model, see [31].

So we have to distinguish between management from a pure technical/syntactical point of view and information management from a semantic point of view. For the first case different tools are meanwhile available: native XML databases, unified parsing strategies like SAX, DOM models, XML to RDBMS mapping tools and so forth. Further aspects will be analyzed in the next sections (Access, Query, Transformation, Webservices).

The latter problem: information management on the semantic level is much more difficult to achieve. In fact, these problems are hot research topics, and some progress is achieved (topic maps, web-ontologies, semantic web), but still all suggested standards and concepts are in an experimental stage. The currently most important ideas and concepts will be discussed in section 8.3. Future will show, which concept will succeed.

8.2.2.4. Access to Semistructured Data

Introduction Besides the complexity of the standard, SGML also suffered from lacking API support. Access to SGML documents was not easily possible from different programming languages and systems. This was maybe one of the reasons, why XML's popularity was increasing so fast: One of the initial XML developers explained the major ideas of XML with the example, that *an undergraduate student should be able to write an XML parser in half a week.* In fact, I doubt that this is true any longer, considering, that even the "simple" XML became pretty complex with namespaces, internal and external entities, document-type and schema definitions, encoding issues, Unicode design and so on. Still there exist a lot of XML APIs for all relevant systems, starting from C/C++ over Java, Perl, Python, PHP to Visual Basic. This was doubtless one of the most important factors of success.

Standards Moreover, in contrast to many other systems, the XML community is highly standard driven and not vendor driven. Even "the usual suspects" were not really able to hurt available standards. One consequence is, that not only APIs are existing to access XML documents, but much more those API concepts are standardized. In fact XML API's can be categorized into the following classes:

- SAX (*simple API for XML parsing*): This is the basic parsing technology and is usually the underlying strategy of all other API's. A SAX parser uses a callback mechanism to transform elements of an XML data stream into events sent to the registered application object. This application is then triggered by those SAX events and can react in an appropriate way.

- DOM (*document object model*) based: The (hierarchical) XML document is mapped to a analogous representation of tree-like organized objects[5].

[5] Usually those objects are kept in memory, but recent parsing strategies also allow to keep only parts

- XML *binding*: This is a new strategy that allows a closer binding between application classes and data (XML), meaning, that XML elements are 1:1 mapped to application objects (including data-type conventions), which makes access very easy for particular problems.

- XML *serialization*: Special libraries allow object (tree) serialization (comparable with the Java Serialization API) to XML data formats. Partly this is even configurable.

As a consequence, application developers have a broad toolset of XML API's that leverage access and creation as well as validation, query and transformation tasks.

8.2.2.5. Query Standards and Systems

Another key requirement in every database management system is an appropriate mechanism to query the data [134]. A similar mechanism to SQL in relational databases was needed for semi-structured data. Different specifications were developed and still are under development. The most important specifications in the XML domain are XPath [134] and XQuery.

Currently XPath Version 1.0 is the dominating technology, it allows to address parts of an XML document using a (Unix) path like mechanism, moreover a set of specific functions is available for tasks like text processing.

XPath is by different reasons not the optimal solution for querying documents or XML databases (from the database point of view), but is the technology used in XSLT for addressing (e.g. in templates). This is the reason for a far larger user community, compared with XQuery. Current XML databases like Xindice, DB2, Oracle are using it to query XML collections.

The second specification XQuery Version 1 is somewhat similar to XPath, but has no widespread use. The W3C developers obviously detected those problems (particularly as XPath and XQuery are too similar in many ways) and plan to unify those specification in version 2, which seems to be a wise decision.

8.2.2.6. Webservices

Introduction As explained already at many locations in this thesis, current data storage specifications and concepts should always consider interoperability between different platforms and systems. Webservices are one answer to this problem-field in the XML world. Webservices means in short: data exchange on the basis of web-standards like http (hypertext transfer protocol) and SMTP (simple mail transport protocol).

of an XML document as tree representation in the main-memory to allow the manipulation of even large XML data.

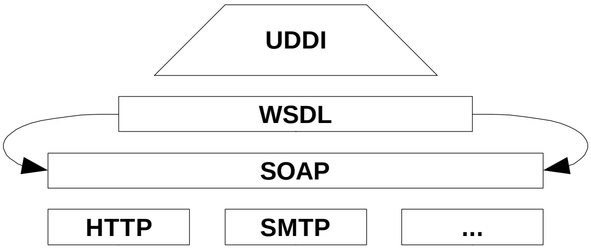


Figure 8.1.: Webservices are a virtual component model using Internet standards for remote procedure calls. The basis are transport protocols, the next level is a message format like SOAP. The properties of the webservice is formally described using WSDL, and UDDI is a registry mechanism for locating webservices.

XML chunks (SOAP messages) are exchanged in a decoupled question/response mechanism. This is similar to RMI (Java) or CORBA/DCOM (Windows) remote procedure calls, but in a platform independent way. Additional mechanisms and standards in the webservices domain are: SOAP, WSDL and UDDI.

SOAP — Simple Object Access Protocol SOAP is the underlying messaging protocol, it is the XML representation of the question sent from the client to the server demanding specific information, and encodes the response sent back from the server to the client. This SOAP encoding is platform neutral and allows to integrate systems of different vendors, as well as different hardware and operation system platforms. However, SOAP is a rather new specification, but already the first SWP research version test-implementations was using SOAP to add administrative functions to the system were performed.

WSDL — Web-Services Description Language As a server might offer different services, it is useful to have a *definition language*, that allows to specify what kind of services are offered and the specific parameters of the services like:

- All functions the client may invoke
- All function signatures (parameters)
- The response (format) of the server

Implementation of a webservice usually starts with the writing of a WSDL file. This file is the basis for the next steps: Tools are available to create the server classes for the concrete implementation at the server side, as well as the client classes to access the SOAP server.

UDDI — Universal Description, Discovery and Integration This last specification is set on top of the SOAP/WSDL/UDDI pyramid (see also figure 8.1), and is at the same time the most experimental. It is thought to be the world-wide *yellow pages* of all public webservices. The idea is obviously important, but there are currently no stable and mature implementation, hence we have to wait whether this standard will succeed.

8.2.2.7. Transformations

The last but highly important aspect in data management is transformation [4]. As (XML) data is enriched with metadata and there are always a lot of possibilities to store similar types of information, it is often required to transform data based on one schema to another schema. This is not only useful for data migration/conversion, but also for publication issues, as data based on arbitrary (XML) schemas can be transformed to the XHTML schema or *formatting objects* schema which can be used for web or print publishing.

Also here, different specifications and tools exist. The most notable ones are DSSSL and XSLT (extensible stylesheet transformation language). The latter is the widespread one in the XML domain; DSSSL comes from the SGML and seems to be (as a Scheme derivate) hard to learn for many users. Yet both languages allow to transform XML data from one schema basis to another. Additionally query mechanisms like XPath (XSLT) can be used.

8.2.3. Structure with (Application) Logic

Conventionally, application logic and data were clear separated issues. With the advent of new technologies like the World Wide Web, those things started to mix up. Just to give an example: Consider web-pages written in HTML: this is data, no doubt. What about web-pages with a Java applet, a Flash animation or even "worse" Javascript Code, that is embedded in the page, or Office documents enriched with Visual Basic scripts or access databases with VB and macros.

Is *this* page data, or a program? The question is of great importance as more and more "data formats" are mixed up with logic, not even in systems as the described ones, even in "clean" XML formats: think of scalable vector graphics (SVG). Those systems have the advantage of *richer content*, that means this content is easier to produce ad-hoc, it allows more functionality without the need to construct a complex application with a clean separation between content and logic.

It is not possible to answer the question here, whether this development of "enriched" data-formats is wishful or not, but it is to note, that the advent of such formats create comfort on the one side, but maybe problems on the other. The first and most obvious consequence is, that it becomes much harder to access the information in these data formats, as information then means also correct execution of application logic. This does not work all the time as it is "demonstrated" by all the Javascript problems between different browser types[6].

So the conclusion, that there is a high probability, that many web-pages produced today will not be accessible in 10 or 20 years, is obvious[7]. Particularly as logic is more and more used at crucial parts of the system like in the navigation. The short term advantage of richer formats turn out to produce side-effects that are most probably not wishful like:

- Archiving this type of content is very difficult: *what is the content*? Already to answer this question is hardly possible. Archival would mean the need to archive the data formats *including the complete application infrastructure*, which will most probably not work in future systems.

- Even if one would decide only to archive the "pure data" without code, it becomes difficult, as the missing logic (e.g., Javascript, Word Macros, ...) can make the access to the information impossible.

- Embedded application logic may even modify the data — so what data "snapshot" is to be archived?

- Traceability of decisions and information flow can become very difficult when even a few years later systems are no longer accessible.

The consequences are clear: The terms data and logic become rather obsolete under such conditions. If possible, even in *rapid application development scenarios* a clean separation of content and logic should be provided, and if logic is necessary to understand how to transform data into information, these steps should be outlined in an abstract, not too concrete logic. This is also the reason, why many authors do not recommend to use syntactical structures like *processing instructions* in XML. It should be replaced by a more general description inside the XML document. So it can be seen as best practice to avoid structure with logic, when possible, and a tight coupling between data and logic.

[6] Those problems occur not only between Netscape/Mozilla and Opera and Internet Explorer, but also between different versions of Internet Explorer!

[7] As a matter of fact: many webpages are not properly produced, not even working with standard conform browsers today.

8.2.4. Synthesis of Storage Mechanisms

As seen in the previous sections, there is no longer a clear separation between different storage/persistence concepts. Relational databases provide support for semi-structured information, XML databases start to offer bridges to relational databases, and middleware components try to build an abstract layer over all persistence mechanisms.

I believe that the gap between the "two worlds" of highly structured, normalized relational data and the (often) semi-or unstructured, sometimes redundant XML data will be harmonized by the next generation database and XML tools. As examples I outlined the functionalities of products like of DB 2 XML extender or Oracle at the RDBMS side and of the Xindice roadmap. The same is true for middleware and O/R mapping tools.

8.2.5. Ad-hoc Structures (in Publication)

Another problem in the current information, content management and particularly in the publication process is, that more and more the classical roles of publishing workflows are removed. 30 years ago, it was clear, that there is an expert in typesetting, who is responsible to finish a book layout, or a printer who finished the book from the technical point of view. Software available today like Pagemaker, Quark Express, Microsoft Publisher or Word *suggest* to the "normal" user, that this very software enables him or her to make the publication without the need of some additional experts.

The consequences can be seen: Typographic quality was never that bad then in the 80's or 90's, but even more important, the lack of know-how of the usual workflow creates sheer amounts of ad-hoc publications that can hardly be reused outside the initial context. No meta-information is provided, generic markup is not used properly, applications used are not understood, websites are created, that are not even syntactically correct (and wide-spread word processors generate unusable HTML code).

Particularly when talking about publication in firms or in science, this becomes a big problem. Cooperation is difficult, reuse of information as well as including such documents into a database becomes a serious problem. Consequently universities, companies, the public sector, ...should be aware of this problem and provide clear and easy to understand guidelines how to produce specific documents and how to collect information in a way, avoiding such situations. The bottom line of this chapter (to conclude the publication example) should be to re-think and re-implement the classical publication roles and apply them in the technical workflow.

Clearly, ad-hoc structures are not desired and a clear separation of structure and publication has to be planned. Thus the editor should edit content not design, the designer should manage design not content and the publisher should deal with the technical issues of publication.

8.2.6. Unstructured Information

The term unstructured information was introduced in section 4.2.3. In short, unstructured information cannot be *directly processed by machines*. There is research in this field to create systems, that allow at least a (semi)automatic conversion of unstructured information of a specific domain into machine readable meta-information enriched formats [66]. Semantic knowledge and a complex processing is required, though.

In short, this very problem is not part of the research described in this thesis. Our focus is to avoid producing unstructured data. This was one reason of starting to do research in the field of collaboration, project management and monitoring. If it is possible to provide useful tools to the project-user, it should be possible to avoid the situation, that relevant information is recorded in an unstructured way.

Although one should not forget, that this is not only, and may not even be primarily a technical problem, it is also an organizational and management problem. To give a popular example: things like "post-it" notes are the first step toward an information-catastrophe. They are *very* unstructured, located at a specific place and not accessible by other collaborators. Hence to avoid problems concerning unstructured information in critical areas, a consequent strategy including management and tools support is required.

8.3. Integration and Ontologies

8.3.1. Interfaces for Integration

Integration of information and data is not only a technical problem of data formats and access strategies. Evidently standards like XML and webservices are a good foundation for integration of data repositories. Those problems are more or less solved (at least all technologies are available to create clean and transparent solutions). But even when data is structured using meta-information, the problem stays, that there are different possibilities to name semantically identic entities and elements. Not only considering different languages (where the element <book> would most probably be identical with <buch>) but also different meanings of words in the same language (<title> as book title or <title> as academic title). Additionally there are synonyms or different ways of writing the same element including "case-sensitive" problems (<Book> is not the same element as <book> syntactically, but from the semantic viewpoint most probably).

The *XML namespace* specification allows to define XML elements and attributes that are *unique* world wide. Hence one part of the problem is solved as elements like <a:book> and <b:book> can be distinguished as *a* and *b* are unique URIs. Nevertheless the problem is not solved, that even <book> elements in *different namespaces* might have the *same semantic meaning*! So it is desired for data exchange as well as for search and retrieval problems to be able to identify the semantics of elements. Unfortunately this is not a trivial and still unsolved task (at least, no technology has succeeded an a broad base), and in the following sections the current research in this area is outlined.

8.3.2. Metadata Initiatives

> "Metadata has been with us since the first librarian made a list of the items on a shelf of handwritten scrolls. The term 'meta' comes from a Greek word that denotes 'alongside, with, after, next.' More recent Latin and English usage would employ 'meta' to denote something transcendental, or beyond nature. Metadata, then, can be thought of as data about other data. It is the Internet-age term for information that librarians traditionally have put into catalogs, and it most commonly refers to descriptive information about Web resources.
>
> A metadata record consists of a set of attributes, or elements, necessary to describe the resource in question. For example, a metadata system common in libraries — the library catalog — contains a set of metadata records with elements that describe a book or other library item: author, title, date of creation or publication, subject coverage, and the call number specifying location of the item on the shelf." *Diane Hillman [45]*

Having understood the meaning of metadata, it becomes clear that an essential component for success of any metadata strategy is a global used standard. A fragmentation into different standards again limits interoperability. This means, that the way resources are described as well as the meta-information itself should be unified:

> "[...] One of the major obstacles facing the resource description community is the multiplicity of incompatible standards for metadata syntax and schema definition languages. This has lead to the lack of, and low deployment of, cross-discipline applications and services for the resource description communities" *Renato Ianella [48]*

Several activities are researched, the most important probably is the RDF (resource description framework) [87]. RDF is an XML based standard with the following basic ideas:

- RDF is a lightweight standard based on web-technologies
- RDF is designed to leverage interoperability between different application exchanging data, and a clear expression of semantics is needed

- A *resource* is described by a set of *properties* called *RDF description*, where each property is a *type/value* pair.
- Though RDF is an XML based standard it can be used also in HTML documents.

Additionally non-profit organizations try to standardize the semantic in specific usage domains. The idea is to build a vocabulary that should be used for description of specific resource types, like the Dublin Core standard [34], to give an example (cited from Diane Hillman [45]):

```
<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://media.example.com/audio/guide.ra">
        <dc:creator>Rose Bush</dc:creator>
        <dc:title>A Guide to Growing Roses</dc:title>
        <dc:description>
            Describes process for planting and nurturing
            different kinds of rose bushes.
        </dc:description>
        <dc:date>2001-01-20</dc:date>
    </rdf:Description>
</rdf:RDF>
```

This example shows the use of *RDF* combined with *Dublin Core vocabulary* describing a resource, in this case a multimedia resource. The RDF namespace (http://www.w3.org/1999/02/22-rdf-syntax-ns) is used to specify, that all elements in this namespace, e.g., <rdf:Description> can be uniquely identified as belonging to the RDF standard, all <dc:xxx> elements belong to the Dublin Core namespace (http://purl.org/dc/elements/1.1/), hence are identified as DC vocabularies. *creator* for example is the DC term for the intellectual property identification, *title* is the title of the resource and so on. In the terminology of RDF, <dc:title> for example is the *type* of the description and "A Guide to…" is the *value*.

8.3.3. Topic Maps

Topic Maps (TM, XTM — for XML Topic Maps) — also called the "GPS" (global positioning system) of the Internet — are a standard that can be seen as a competitor to the RDF initiative, although there are some differences from the basic concept [36].

- Topic maps are seen to be more powerful and flexible compared to RDF, which is a more focussed specification.

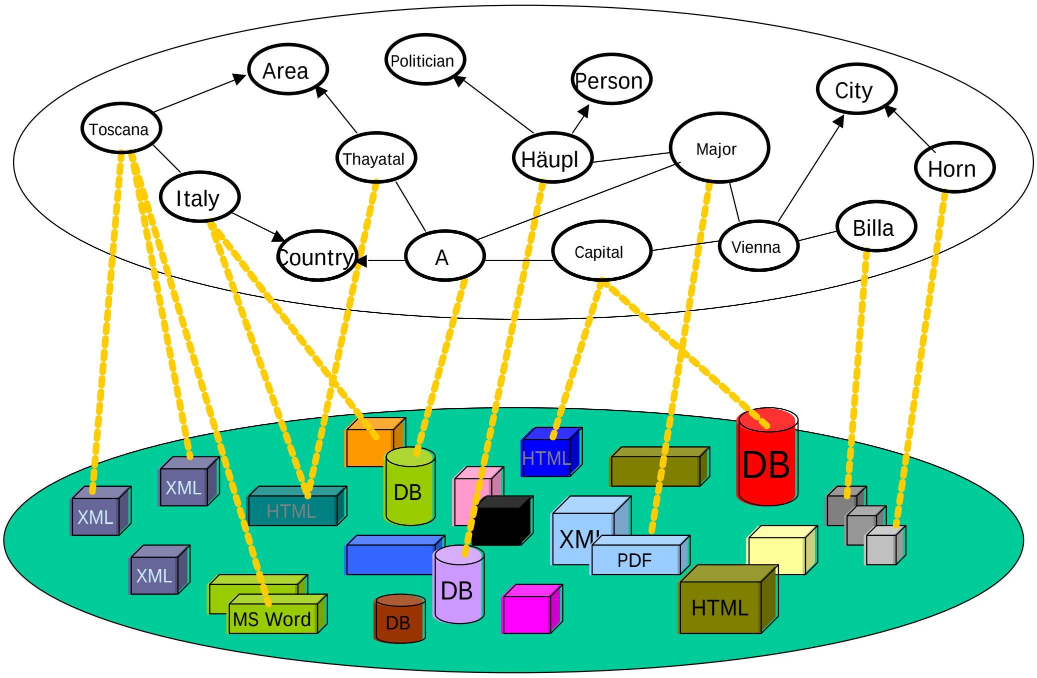


Figure 8.2.: Example for a Topic map: The upper part is the TM, the lower part are the concrete resources, that are connected to meta-information by the TM.

- Topic maps can be defined *external to particular resources*, much more: TMs can be seen as integrators of different resources, where in opposite RDF is meta-information *attached to resources* (This is illustrated in fig. 8.2).
- TMs can define multiple levels of metadata (that is: metadata of metadata . . .)

The design concept of the TM specification expressed some goals for TM:

- TMs should be easy usable and a straightforward application over the Internet is desired.
- TMs shall be applicable to a wide variety of problems
- TM have to integrate well into existing (XML) Standards, particularly standards like XLink.
- XTM documents should be easy to create, human-legible and reasonable clear.

TM Term	Example
<topic map>	graph on top of the figure
<topic>	country, Toscana, area
<occurence>	points to a resource
resource	lower part of figure, e.g. XML document
Published Subject Indicator	URI, JDBC address...
<association> of <topic>s	connection between topics

Table 8.2.: Topic Map terminology referencing the example in fig. 8.2

Table 8.2 gives an overview over the TM terminology, following the TM specification[121]. Additionally Topic Maps are supposed to be a critical element building the Semantic Web, along with RDF and ontologies.

8.3.4. (Web) Ontologies and the Semantic Web

All mentioned metadata initiatives still have the problem, that different persons, institutes, domains and so on are often using a different terminology even for the same real-world-entities. This is particularly problematic, when systems should be integrated. The *semantic web* is a rather new concept initiated by Tim Berners-Lee, and is essentially a combination of open (markup language) standards (*XML*), metadata initiatives (RDF and/or topic maps) and (web) ontologies.

> "In philosophy, an ontology is a theory about the nature of existence, of what types of things exist; ontology as a discipline studies such theories. Artificial-intelligence and Web researchers have co-opted the term for their own jargon, and for them an ontology is a document or file that formally defines the relations among terms. The most typical kind of ontology for the Web has a taxonomy and a set of inference rules.
>
> The taxonomy defines classes of objects and relations among them. For example, an address may be defined as a type of location, and city codes may be defined to apply only to locations, and so on. Classes, subclasses and relations among entities are a very powerful tool for Web use. We can express a large number of relations among entities by assigning properties to classes and allowing subclasses to inherit such properties. If city codes must be of type city and cities generally have Web sites, we can discuss the Web site associated with a city code even if no database links a city code directly to a Web site." *Tim Berners-Lee [13]*

Definition of ontologies [43] and clear standards how to access and how to connect resources is also a basis for agent based systems. The idea of software agents was hyped in the middle of the 1990's, but the high expectations were not even reached in test-environments [72, 100].

8.3.5. Semantic Web continued: Web Services and Agent Technology

A part of the semantic-web vision are so called *autonomous (intelligent) software agents*. The idea is, that on the basis of such *open data exchange standards*, a "biotop" for autonomous software agents could spread. The main characteristics[8] of software agents in a semantic-web scenario are:

- Software agents are small, autonomous programs, that are instructed to solve a particular problem (e.g., a complex search over various systems).
- This agent "travels" autonomously through the network and tries to communicate with information systems to acquire information or solve a mission.
- And finally, the agent should be able to *negotiate* with those information systems as well as with other agents (e.g., in "EBay"-like-scenarios where some agents sell things and others try to buy them).

The drawback is, that there are so many real-world problems unsolved until agent-based systems could become reality. Starting with the problems mentioned above: ontology problems, different information systems with different interfaces, incompatible platforms and so on. Problems that could be addressed with the strategies combined in the semantic web suggestion. Berners-Lee's vision is also connected with a concept of ubiquitous computing, and IT assistants that are "everywhere". Such a scenario can only be really successful, if those assistants are able to exchange information with a lot of very different systems. To give an example following Berners-Lee's vision:

1. You are at home want to see a specific movie with your partner.
2. You start the personal agent (PA), and give the instruction to organize this.
3. The PA starts searching the Internet for movie-theaters that are near your location.
4. Additionally it detects that you have a member card valid for specific cinemas, hence it restricts the results to those cinemas first of all.
5. It compares the movie-start times with the entries in your and your partners personal calender (PDA) and selects a fitting cinema and start time.

[8] It is to remark, that sometimes systems are called *agent based*, even when not all of those attributes are fulfilled, e.g., agents that do not "travel" over system boundaries, but "simply" do Internet queries.

6. It contacts the cinema computer and reserves two seats (as it knows your favorite places in the cinema)
7. Then the agent contacts the traffic service to find out how long the travel to the cinema will take according to the current (or expected) traffic status.
8. At time, it remembers you to leave home, and does this 10 minutes earlier because the car computer registered low fuel, and refueling is necessary too.

This described task(s) are completely usual for everybody, and pose no problems for a human actor, but it is clear, that such a strategy requires a huge effort in ICT to become reality. Of course, my example might be special in some points (fuel of the car), but then again: this is one of the major issues of agent-based scenarios: if they should be useful, they have to be very powerful. Otherwise you always would have to double check all agent solutions and such a system would make no real sense, because in the end searching and booking by yourself would be much more efficient. Moreover, such scenarios pose big problems in terms of security and privacy: agent scenarios assume, that potential hostile code is executed on different information systems (e.g. the personal agent might be executed on the cinema computer system). And the privacy problems might be even more threatening. Again, to be useful, the agent has to access a lot of private information (possibly even including health data, financial data and so on), and this is obviously a risky situation. Finally, *even if* it should be possible to construct such complex systems that are able to do such complex tasks and even negotiate and book, will we — the user — trust those systems?

Even far simpler "search-only" problems will pose enormous problems according to the intelligence, that would be required. A simple example: I was searching for a refrigerator of a specific size. The problem was (1) to find all potential vendors (about 10), that (2) sell in Austria a (3) refrigerator of (4) a specific size. This is a problem rather easy to solve for a human actor, but practically impossible for a software agent, as there are so many things needed to be standardized before a software agent could solve such a task. It is questionable if such a standardization will take place in the near future.

Furthermore, consumer requests show, that most people are not willing to buy, for example, cars that drive autonomously; even though they might be more secure, faster and have many other advantages. Even such a basically stupid action as driving is too important for many people to release it to a computer. Will they allow personal agents to decide to which cinema to go, which doctor to select and at what time to meet friends (because the PDAs took the decision)? I personally do doubt those scenarios, at least for the near future.

However, even as my description sounds rather negative, I believe, that interoperability will become more and more a critical task, even if no agent based systems are involved. I assume that maybe the agent and ubiquitous computer scenarios of

Berners-Lee might be obsolete concepts for the next decade, but the semantic web is not. A wise implementation of standards like RDF or topic maps, web ontologies, web-services and XML may leverage many problems of today, starting with data exchange, longevity of digital information [63], search and retrieval, reuse of information and so forth. And not to be forgotten: all these scenarios will become crucial for project cooperation, especially in a globalized scientific and industrial cooperation.

8.4. Publication and User Interface Issues

8.4.1. Introduction

Publication and User Interface design might not be seen as a core problem of *information management*, but a pragmatic analysis shows, that information that is useful, has to be published or accessed in a way the desired user community can handle. This is trivial but nevertheless very important. For planning an information management system or strategy using certain system(s) one should have the access/publishing problem in mind. This means essentially:

- Besides a clean data storage and management strategy (as described earlier), publication and UI access should be part of the initial planning phase.
- Publication is (as defined in section 1.2.4) one-way, data exchange, but may target different audiences and formats. Hence multichannel/cross publishing is desired.
- User interface design systems should provide support for different systems (e.g., interfaces for databases (JDBC, ODBC), webservices, connectors to different programming languages). Systems that are limited to only one specific environment should be avoided.

This section describes mainly the publication issues, by two reasons: First of all, the UI/programming problems have been part of earlier analysis (e.g. in chapter 6) and additionally the area of cross publication the last years showed an increasing speed in new attempts to new standards and frameworks. Moreover, UI/access and publication problems get closer, especially driven by web-oriented applications, where document oriented structure is often mixed with user interfaces. This is a new paradigm in application development and it turns out, that a clean strategy for cross publishing usually also allows a good access to the information from the UI point of view.

8.4.2. Markup Revisited

Markup languages like HTML or XML, or similar concepts like templates in word-processing or desktop-publishing applications allow to enrich (textual) information with meta-information. Unfortunately the difference between *generic* and *visual* markup is often misunderstood and even complex documents are “structured” using only visual markup like font settings, font faces color and the like. Basically these markup problems have been analyzed earlier as a clear separation between data/information and logic was demanded.

The same is valid here: A publication process should take care to separate information entry, processing and finally the visual publishing. This is the reason, why HTML is more and more replaced by XML. Alhthough HTML can be used for generic publishing, XML is far more powerful in this respect. It allows to collect information, enrich this information with metadata and separate this step completly from visual publishing problems. In succeeding steps, those visual informations necessary to publish the information is added (e.g., using stylesheets) and the publication process is finished.

Similar problems arise in “publishing” information using applications and graphical user interfaces. It turned out, that writing “hard-coded” user interfaces has several disadvantages. Among others: they are not easily portable, modifications are often difficult, visual appearance cannot be changed without problems. Also here, several attempts can be seen, that try to solve this problem. As examples: The Java/Swing or Microedition UI system, web-frameworks like Struts, or most interestingly (though not very widely used), the XUL concept introduced by Netscape and implemented in the Mozilla browser [137]. XUL is an XML language that allows to describe user interfaces in an abstract way using an application generator or interpreter which produces the logic and generates the concrete UI for the desired platform.

8.4.3. Reusability

Particularly the development of the last years in Internet publishing demonstrated that publishing on electronic media is fast and easy. At least this was the impression a few years ago. Soon it turned out, that “fast-and-easy” publishing means that possibly thousands of documents are part of a website. The problem increases with the concept of hyperlinking information. In an ad-hoc publishing process problems may arise, that can be hardly controlled:

- Hyperlinked documents easily run out of control, when “simple” HTML linking is used. The reason is, that one never knows whether links point to a particular document. If such a document is removed or changed, eventually dead links may be the consequence.

- Design changes might become a problem, when thousands of documents have to be modified; the same problem arises when the site has to be restructured.

- Multiuser access might be a problem considering versioning and multiple persons working on the same document(s).
- Corporate identity should be guaranteed: Usually it is not desired, that every user creates documents with his or her personal style.

The consequences are clear: Publishing should be well-planned from the beginning. Many tools exist on the market; starting from "Web Editors" that allow to manage websites up to complete content management systems. Whatever tools are selected, the planning has to start before the first publishing activities.

However, pure HTML publishing or print publishing might be problematic when reusability is an issue: Publishing strategies should be considered that follow the suggestions in this chapter and allow to organize the information in a generic way (in the database, in XML, or in a mixed form), and has a clear separation between data and publishing stylesheets. This solves most of the problems mentioned above: Data can be changed without impact to design, users are "forced" to unique styles, data can be reused as well as other sources can be integrated into the publishing process. Moreover interactive websites are easier to generate.

Nevertheless it is to mention, that a solid planning is required. This is a time-consuming task and requires experts in the field. Bad planning of a corporate (web)information system might be even worse than ad-hoc publishing. So the initial phase is time-consuming, but when a good strategy is selected the quality of the daily publishing process is enhanced dramatically.

8.4.4. Standards

Several standardization efforts have been mentioned already. In fact the problem is, that there are too many standard suggestions available. Starting with recommendations from World Wide Web consortium like XML, DTD, Schema and XLink, XPointer, XSLT, Namespaces, XPath and so on, going to suggestions from various other organizations like Docbook or Topic Maps.

Some technologies are meanwhile accepted like XML, XSLT, XPath, formatting objects, namespaces, others unfortunately not. This is particularly a problem in areas like *linking resources*: Standards would be urgently needed and are basically available (XLink, XPointer), but are not used. The XLink standard for example, allows to declare hyperlinks outside the specific documents in special linking definition documents, which makes management of linked information easier. This is basically an old idea of hypermedia, but unfortunately, up to now, no such system has become popular enough to succeed. The main reason is the lack of supporting tools. Without tools the standards are not used and vice-versa. Similar problems can be seen with standards like docbook. This is basically a complex schema describing book-like documents and can be compared to LaTeX. Even here, a lack of (quality) tools is the problem.

But this might change as demonstrated by SVG (scalable vector graphics). Meanwhile it is supported by all major graphic applications, and as soon as native browser support will be available, I suppose SVG will be used on a broad basis probably even replacing the proprietary Flash in many domains.

So the consequence is, that we have to reserve time to check whether standards are available and if they fulfill the requirements before new concepts are invented. This enhances interoperability and reuse. The importance of this consequence can be seen at standards like RSS (rich site summary, RDF site summary), where nearly all newsfeeds are powered by RSS and every portal tool allows easy integration of RSS feeds. This demonstrates the self-amplifying power of open standards.

8.4.5. Frameworks

As a consequence of the complex design of current publishing concepts, which target multiple devices, should manage content, allow database integration and so on, publishing frameworks were developed in the last years to leverage those processes. This is good news, however the other side of the story is, that all frameworks I know are pretty complex and evaluation of a contemporary framework like Cocoon [25], Zope, PHPNuke . . . takes a lot of time. But selecting the wrong tool may generate a lot of work later on. A suggestion should not be made here, because development is so fast, that every suggestion will be outdated very soon. Nevertheless I believe, that some points should be taken into consideration, when evaluating publishing frameworks:

- Selecting any framework usually is a long-term decision! Changing to another system is often a lot of work.
- Is the system proprietary? If so, support might be available, but if problems arise, the system might become expensive and a migration could be difficult (as this is obviously not supported by the vendor of the “old” system).
- Is the system designed for a very specific purpose? This is true for systems like PHPNuke: they might be a good selection, when they fit precisely, but a demand slightly beside the usual usage pattern typically creates enormous effort and rewriting parts of the system.
- Is the framework designed for a general purpose (like Cocoon): Then the limits are not so close, but on the other hand, even for simpler problems more work has to be done.
- Is there someone in the company/on the institute that is able to control the system and solve problems? If not, an easier/other solution should be selected.
- The (technologically) second best system will be the best fitting, if it is under control, compared to a technological superior one, that lacks knowledge in-house!

- Is the system a dynamical or a statical one, meaning: are e.g., statical HTML/PDF documents produced by the system and served by a normal web-server (Apache ant XSLT) or is the publishing process database or XML based and dynamical. The latter case is more flexible but even more sensitive to technical problems and performance issues.

- Is the system platform independent or is a specific proprietary hard/software operating system required: if so (and as this is a long-term decision), one becomes very dependt on the future policies of a specific vendor.

Of course this list is not complete, but it should give an idea in which the direction an investigation for a content/information management system should go.

8.4.6. Complexity

So many frameworks exist and every one promises to solve the publication problems of today and of the future This is clearly not the case. In fact, publishing frameworks may be a big aid in solving complex publishing tasks, but one should never forget, that the problem of "all" frameworks is, that they create an additional level of complexity. The question to be answered in the concrete problem-solving process is whether the complexity reduced by the framework is bigger than the complexity raised by it. The answer of *extreme programming* [12] would be simple: Never add features you might use in the future. Add them, when they are needed.

On the other hand in a publishing process many people might be involved and a permanent change in the publishing system will confuse those users (particularly the unskilled ones): So I suggest to evaluate frameworks and use them. But is is extremely important to be aware of this additional complexity of those systems and have some skilled personal that is able to control the system!

8.5. Information Management — A Conclusion and Reference

8.5.1. Introduction

Considering the suggestions made in this chapter, information management should become a controllable effort. Nevertheless there is to consider that one has to distinguish between different "scales" of information domains. To give a brief categorization:

1. personal information management (IM)
2. IM of small groups (workgroups)
3. IM of company scale

4. “meta-IM” in company scale
5. knowledge management

This list shows increasing size and complexity. Not only in terms of data size and storage space needed, but more importantly in terms of complexity, e.g., when cooperation is desired, versioning becomes an issue and complexity of backup strategy increases, as well as availability and access control becomes more important.

8.5.2. Personal Information Management

Personal IM is typically not very problematic. A single person can easily organize his or her information using standard technologies like file system, desktop database, email application, and so forth. When no multi-user access needs to be granted, versioning, backup and availability of servers and services is no big issue.

Nevertheless there are things to consider: Personal work has to be “up-sized” at a specific point (e.g., from a personal software project to an open source multi-programmer project) and longevity of digital information might be of importance! This is not only in the field of commercial work, even considering private “stuff” like digital photos or other multimedia documents.

As a consequence, even individual users might want to consider (at least partly) suggestions made for corporate users. It is not always an easy decision, though: Starting personal programming projects using management strategies like used for big open source projects (build tools, versioning systems, documentation tools ...) will most probably increase the quality of the personal work, but also adds an additional layer of complexity and effort to the project(s), and requires additional knowledge acquisition for the individual. As it is no problem in a company to have an administrator that deals only with this specific tools, this has to be done by the individual programmer him or herself. The same is true for multimedia data management.

A solution could be searching for available tools on the Internet. Just as an example: There are email providers available, that offer SPAM mail filtering as well as virus checking and there are services for open source programmers that offer versioning support (e.g., Sourceforge[108]).

8.5.3. Workgroup IM

Even for individuals or very small workgroups (say, 3 to 15 user) it might be a good idea to consider to spend a little more time into the planning phase, not only because a transformation of projects to bigger scale becomes easier, but also because the organization and quality increases. When starting projects on a workgroup scale a complete IM strategy is already suggested. However issues like service availability and scalability are not of such importance. Moreover one will not necessarily provide the services and server(s) by oneself, but use available tools, provided by Internet companies or Internet providers. A lot of information about such systems is available in open source communities (web) or discussion forums and mailing list. In the planning phase those resources should be taken into consideration.

8.5.4. Corporate IM

In the corporate IM, all suggested methodologies in this and in all related chapters should be evaluated and implemented. At least one person (or one group of persons in big companies or institutions) should be available to control the servers (hardware) and the services (software). Particularly the following aspects are part of this group:

- Selection and definition as well as customization of open standards (like XML, Docbook, metadata, . . .), see this chapter and chapter 6.
- Integration, metadata and reuse of information, see this chapter and chapter 11.
- Cooperation Issues, see chapter 10.
- Security and access control issues.
- Consider the risks of IM, KM and other efforts, see chapter 9 and, e.g., [113].
- Backup.
- Longevity of digital information (see chapter 7).
- Prepare the IM system to be ready for “meta-IM” like *data mining* [56] and *data warehouses* [51] (which is not part of this thesis, but a lot of information and literature is available) and knowledge management; for suggestions see chapter 9.

Also the next section might give some ideas about the infrastructure of coorperate projects with a brief conclusion of the strategies used at the literature and *Open Science Workplace* projects.

8.6. Strategies in OSWP and Literature Projects

8.6.1. German Literature and Language Projects

In the literature projects as well as in the OSWP projects all types of information occurred. It is interesting to see, that although the literature projects started in 1999, a large development in terms of semi-structured information management and handling took place since then. Experiments were made, to use XML directly to handle the content in the literature projects, but in 1999 it turned out, that tools and standards were not stable enough for productive use. Hence "conventional" database technology was used to manage the content, although the mapping of the hierarchical and semi-structured data was not always easy. Neither from the model nor from the user interface point of view.

Four years later with more mature tools in the domain of semi-structured data and XML processing, probably another strategy would have been taken. However: with regard to longevity of the cultural heritage, it is possible to extract the data to XML format for future reuse. Project documentation was done in HTML. The same here: some new standards matured during those years, but experiments with XML based content management and publication systems were performed, but not used because of the instable status. Additionally first tests using RDF/Dublin Core Metadata in XML and HTML were done. Also naming in different areas (database) were tried to keep Dublin Core conform.

8.6.2. Open Science Workplace

The OSWP project(s) started in 2000, and during the project work (especially until the second part started in 2002), the situation changed a lot. Different types of information has to be handled when dealing with project related information and data. First of all there is "typical" relational data like user information, project information and the like. But considering the project and task structure (see sections 3 and 13 for details) this is typically hierarchical information. As we decided to use a middleware (application server), the mapping of this tree-based structure was leverages by object relational mapping tools (as mentioned above). So storage is done in relational databases, but access through O/R mapping API.

Moreover there are other types of information involved: resources for example: in the first project file resources could be organized with the SWP tool. To store these resources a mixture of relational database (storing the meta-information) and file system was used (storing the files). Considering Metadata research is currently still on the way: In one paper [100] we suggested a new combination of (RDF) metadata and webservices (SOAP) to increase interoperability in agent based applications. Also in the resource management and XML-export part and of OSWP considerations about wise strategies to ensure longevity of project information as well as easy data exchange by using metadata is planned.

Additionally the (O)SWP projects were different to the literature projects in another important way: The literature projects were developed by one individual programmer, which made some source code-management issues easier, whereas (O)SWP projects were developt as a team. So learning the lessons of SWP in the second OSWP project a clear code-management strategy was implemented using concurrent versioning system (CVS) and the Apache Ant "make" utility. Both systems are standards in the open source community and there are many good reasons for this. It is not recommended to start any project (except trivial ones) without the support of those tools. Furthermore, CVS was used not only as source-code management tool, but also to store and manage the project website, XML based documentation and all related work like diploma and PhD thesis . . .

And finally, project documentation has to be done: user documentation, website for the open source community and so on. As content management systems and publishing frameworks matured, a new strategy (compared with the literature projects) was chosen: Project (user) documentation was written in docbook XML and published using XSLT/FO to target formats like HTML or PDF, and also the website was done using XML/XSL publishing using the Cocoon publishing framework.

9. Knowledge Management

9.1. Introduction

Knowledge Management was not intended to be a core topic of this thesis. However, depending on the point of view (see also section 9.6 on page 140) most activities described in this work can also be seen as part of a knowledge-management infrastructure. Or to put it that way: from a KM point of view, a CSCW system is already a specific part of a KM system, hence this aspect has to be analyzed in more detail (see especially section 9.6.2 on page 141).

Besides this general aspect of the relation between CSCW and KM systems, a new communication/question based KM approach will suggested. This is particularly interesting as it would integrate seamless into CSCW/communication systems. This is an essential factor as Stewart et.al. analyze:

> "Unfortunately, contemporary technology for knowledge management is a hodgepodge of executive IS, group-support systems, intranets, decision-support systems, and knowledge-based systems." *Stewart et.al. [113]*

So all efforts that try to integrate systems (communication, collaboration, project management and knowledge management) as also described in chapter 11 are a significant progress to the contemporary situation.

Finally further perspectives like the ideas about *artificial intelligence* and *intelligence amplifying* systems will be discussed, besides the possible risks of KM systems.

9.2. The Draw-back of Knowledge Management

> "Employees often do not have time to input or search for knowledge, do not want to give away their knowledge, and do not want to reuse someone else's knowledge." *Rus et.al. [91]*

This is an observation often made, when companies try to implement new communication, information- or knowledge management systems. There are may reasons for this obvious failure, and in this thesis approaches are suggested, that should leverage the steps toward a universal (project based) information and knowledge management strategy. The concept described here should overcome the problems mentioned by

Rus et.al. as the individual employee should see concrete advantages in using the KM system (by getting as well as be adding information and knowledge).

Whatever difficulties will arise when dealing with KM, it is an important topic for universities as well as for companies and might create a huge increase in productivity of institutions. So what is *knowledge management*? One common definition is the following:

> "The objectives of knowledge management (KM) in an organization are to promote knowledge growth, knowledge communication and knowledge preservation in the organization. (in [30])" *L. Steels [112]*

(I will discuss the various different interpretations of knowledge management after developing my specific ideas in section 9.6 on page 140.) Nevertheless traditional knowledge management concepts often show problems in acceptance of the end-users. There are several reasons for this:

- Knowledge acquisition is usually a "proactive" process. This means, that each expert user has to give input into the system without having an immediate use or benefit from the system.
- Even if knowledge topics are entered and managed properly, knowledge management systems often only help to provide a contact between the person who has the problem and the person who might find a solution. The solution itself often is not included in the system.
- A person who might solve a problem might feel not highly motivated in offering help as it (1) disturbs his normal activities and (2) as he or she realizes no advantage for him- or herself.
- Information in knowledge management systems should be updated regularly to be useful.

9.3. Crossing the Gap: Back to Nescience

> "As already seen in science, also in the financial sector, more knowledge does not produce more truth and more security, but in a paradox way more options, more insecurity, hence more specific nescience[1]."
> *Helmut Willke [129]*

[1] The original citation is in German and is translated by the author. The original citation can be found in the appendix section A.1.4 on page 200.

The term *knowledge management* obviously suggests the necessity to deal with information and knowledge, but as often detected (and also discussed here), direct access to knowledge is difficult by many reasons. On the other hand, living in a *knowledge society* means, that development is driven by new detections/findings/research, new information, that needs to be processed and finally new knowledge to be created. But at the same moment, when knowledge is generated and applied (!) society proceeds one step higher in system complexity; the nescience, the insecurity increases, new problems arise, and generally spoken the system risk grows [39, 129].

Following those considerations, the new concept suggested here assumes an approach to the problem inspired by Willke [129].

> "The crisis of knowledge ist cognitively driven by the new relevance of nescience. Operationally it is driven by the necessity to make the right mistakes faster than the competitors to intensify learning processes, what means developing expertise in handling nescience[2]." *Helmut Willke [129]*

According to the problems described above, a knowledge management system (KMS) should fulfill at least the following requirements:

- A *re-active process* is assumed to be more useful than a proactive process. The reason is, that people are easier to motivate to act, when they have a problem, not when having a (possible) solution. Especially when it is not clear that there is a reward for the knowledge added pro-actively.
- A KMS should not work simply as a medium to enable contacts between people who know and such who don't. Moreover it is desired to build a *knowledge repository* that keeps relevant information for more than a single usage case.
- Persons who have *relevant* knowledge must be motivated to provide this knowledge and share it with others.
- A KMS has to be a living system with frequent interaction. Hence it is useful to integrate the KMS into existing CSCW or communication systems.

Following these prepositions, the core idea of the suggested concept is to put the management of the nescience into the center of interest, or in other words: nescience can be expressed in the form of a *question*. This question shall be the starting point of the knowledge acquisition and management. Moreover, as described in [1] knowledge management should be a highly integrated task. So an implementation in the OSWP workplace can lead to synergistic effects:

[2] The original citation is in German and is translated by the author. The original citation can be found in the appendix section A.1.3 on page 199.

> "Coordination and collaboration support must be a first order citizen of KM [...] information retrieval and management systems must deeply be interwoven with the collaboration-oriented everyday work." *Abecker et.al. [1]*

and

> "Software development is a group activity. Group members are often geographically scattered and work in different time zones. Nonetheless, they must communicate, collaborate, and coordinate. Communication in software engineering is often related to knowledge transfer. Collaboration is related to mutual sharing of knowledge." *Rus et.al. [91]*

Furthermore, I will show, that this approach fulfills also the idea of *Corporate Knowledge Management* described in [30]. Dieng. et. al describe the building of a corporate memory as relying on six steps: *(1) Detection of needs in corporate memory, (2) Construction of the corporate memory, (3) Diffusion of the corporate memory, (4) Use of the corporate memory, (5) Evaluation of the corporate memory (6) Maintenance and evolution of the corporate memory.* All steps can be found in the proposed concept in a very natural and user-friendly implementation, as will be shown in the following sections.

This idea of a question based KM system has also be described in [96].

9.4. The Question

The first goal to be achieved is to motivate users to use the KMS. This can be done as the system allows the users to pose questions[3]. This is a good concept by many reasons: First of all, the users are motivated as they can use the system to solve there own problems. Secondly only topics are included into the system, that are really relevant to the persons involved in a project, a company... Moreover the (project) manager can receive an idea about the open problems in his or her division or project(s) by watching the problems posted to the system. Steering activities like getting knowledge from outside may be a consequence.

More generally spoken, the question can be seen as a *crystallization point for knowledge*. Questions show interest as well as problems; questions can also start communication, bring up new ideas and initiate projects.

[3] From a technical point of view, the question has to be "normalized" in a way, that similar questions can be detected as such, even if they are posed slightly different. However, this specific problem is discussed in other research areas like [17] and [74] and is not part of this research.

9.5. Closing the Gap: The System

9.5.1. Introduction

Figure 9.1 shows the basic ideas of the nescience management — question driven concept as use-case diagram. In the diagram the following roles are introduced:

- Project User: This is a person in a project, company... who has access to the system. This is a “normal” user.
- Administrator: This is a person who administrates the system. To keep the diagram simple, this role is not included in the diagram, as the administrative functions are not central to the functional ideas.
- Project Manager: This person manages a project. This user has advanced functionalities compared to other common project users.
- *System:* This role is implemented as software component. Automatic administrative tasks are performed by the system.
- *Intelligent Agent:* The system should have an open design using open W3C standards like a webservice interface [125]. So software agents may be implemented to support specific KM and integrative tasks.

The next sections describe the details of the use-cases and the implementation.

9.5.2. Documenting Problems

If a user has a problem, he/she can pose a question to the system. The system checks whether this question or a similar question is already in the knowledge repository. If the system detects possible identical or similar questions (no matter whether they have been answered already), the user is asked if those questions are similar to the one he or she asked. If not, the new question is added to the repository of open questions.

If no answer is given to the question by some other users, the system asks after a specific time if the question is still relevant or if the user has solved the problem already. If so, the user is asked to write an answer to his own question. The reason is simple: this system should build up a knowledge repository, not only a question repository.

9.5.3. KM Portal and Evaluation/Ranking System

Every time the user visits the KM “portal” the recently posted questions are shown and the user is encouraged to answer questions if possible. Moreover a user can register

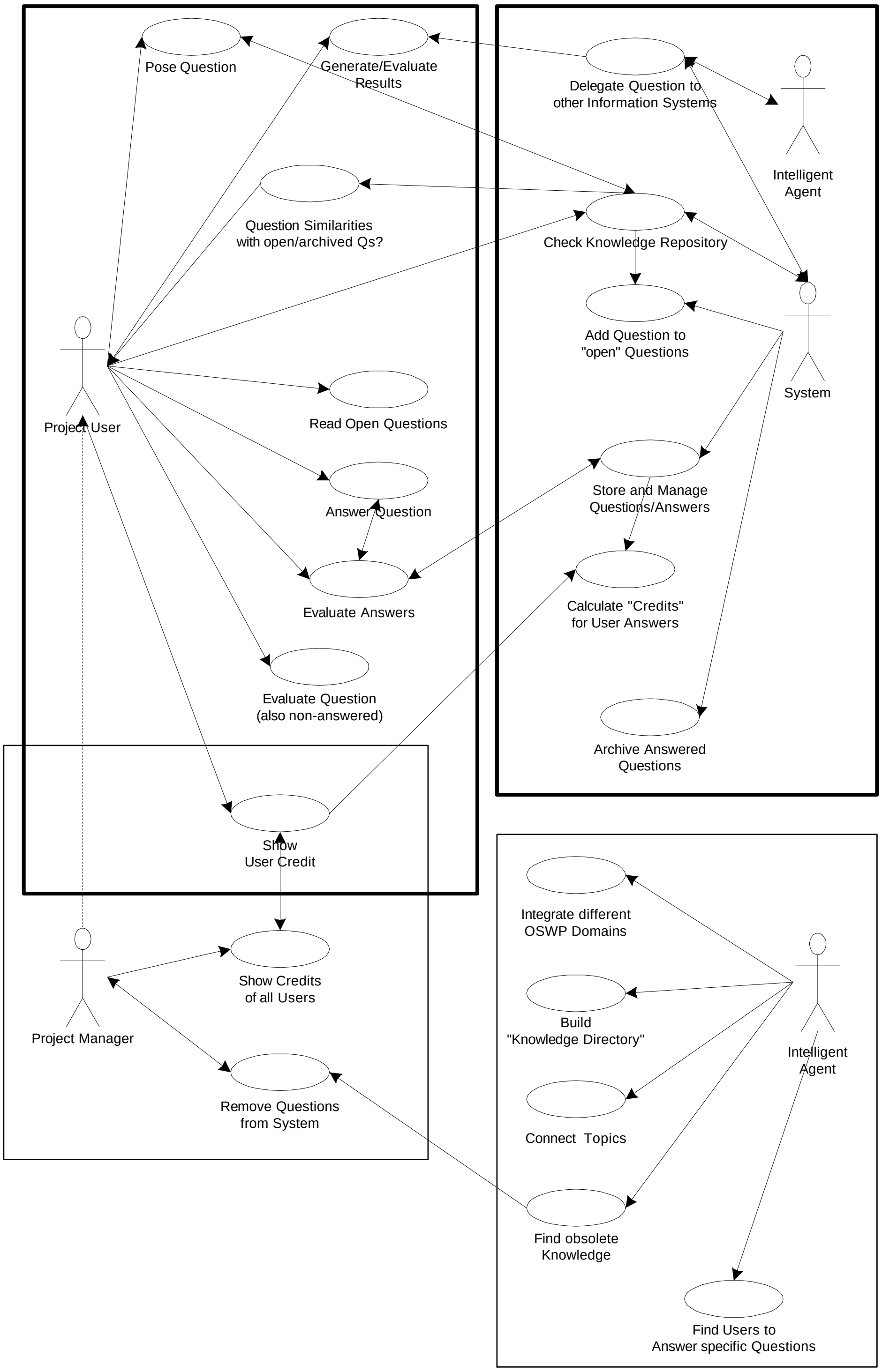

Pose Question
Generate/Evaluate Results
Delegate Question to other Information Systems
Intelligent Agent
Question Similarities with open/archived Qs?
Check Knowledge Repository
Add Question to "open" Questions
System
Project User
Read Open Questions
Answer Question
Store and Manage Questions/Answers
Evaluate Answers
Calculate "Credits" for User Answers
Evaluate Question (also non-answered)
Archive Answered Questions
Show User Credit
Integrate different OSWP Domains
Project Manager
Show Credits of all Users
Build "Knowledge Directory"
Intelligent Agent
Remove Questions from System
Connect Topics
Find obsolete Knowledge
Find Users to Answer specific Questions

Figure 9.1.: Use Cases for Knowledge Management Concept

him/herself to open questions to demonstrate that he/she is interested in the answer to this question, too. This increases the importance rating of the pending problem.

If the user answers a question, this answer is added to the repository and the persons registered to the question receive a notification, that the question has be answered. Then all users should evaluate the quality of the answer. This information is very important as it helps ranking the answer in the KM system as well as it helps to give credits to the person that wrote the answer.

It is important to remark at this point, that first of all the questions should help to get immediate answers, but secondly this also can be regarded as crystallization point for ideas and concepts. As Sunassee et. al. remark:

> "The chief knowledge officer needs to establish both *pull* and *push* factors to force employees to share knowledge. An example of a push factor would be to force employees to search through the knowledge repository before starting a project or a business venture." *Sunassee et. al. [114]*

This is guaranteed from the technical viewpoint of the system, but the users must be encouraged to see the opportunities of the system used this way. Hence it is important that the users search the repository, post question but also rank unanswered questions and eventually discuss about answered as well as unanswered questions. This gives managers the chance to detect problems and support the coworkers.

9.5.4. The Question as Bridge to Other KM Systems

KM literature often emphasizes the importance of strategies from data mining area [30, 110]. This means, that certain KM systems focus on search, retrieval and integration of different information resources like documents, databases, . . . to build an "organizational memory" using (among others) ontologies. This is a very important strategy. The prerequisites for this reuse of information in different other locations of this thesis are already described, mainly in the chapter "Information Management" (see chapter 8 on page 99).

However, one point not mentioned there is how to integrate the "prepared information" into a knowledge repository. I believe, that this question-based system could be a well suited integrative tool (see also Fig. 9.2). As soon as a question is posed by a user, the system should evaluate the question resources as mentioned above, but also start a "broadcasting" of this problem to other registered information subsystems like the database pool, the document repository and the like. An "intelligent" analysis tool should then present the user a selection of hopefully useful resources already available in the system. Additionally also agents could be written, that search outside the system, e.g. by using web-search engines or newsgroup search engines. As soon as those results are collected, the user should decide whether these resources already

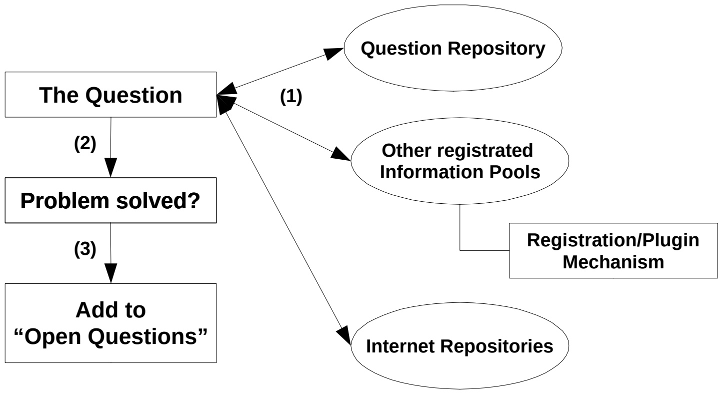


Figure 9.2.: "The Question" based approach can also be seen as smart integrative approach in integration of KM tools/concepts: (1) Question is posed and the question repository as well as "pluggable" *other information pools* are queried. (2) The user is asked if the problem is solved by the results delivered and (3) Either the open question or the solved question eventually is added to the KM repository.

solve the posed problem, and if not, the question should be put to the "open question" repository.

If the problem is already solved by the resources provided, it could be a good idea to let the user select and mark the resources that helped him solving the problem and put this question as solved into the knowledge repository including the marked resources.

9.5.5. Credit System: KM as Marketplace

This kind of KMS can be seen also as a knowledge marketplace. As users rank answers to questions, this ranking will be calculated to credit points for users who answer questions. Those credits can be seen as a money equivalent, where multiple strategies can result, depending on the company structure or the intention of the system. Just to mention a few:

- Users with high credits can be published at the KM portal.
- The credits can be exchanged to real money or other beneficiaries to encourage users to share their knowledge [91].
- Questions from users with high credits could be handled with priority (e.g. on the portal page) to encourage cooperation.

- A combination from user credits, question ranking and evaluation can be used to enhance the quality of the KMS repository.

9.5.6. Project Manager

The (project) manager(s) can browse the credits of the colleagues and eventually react to the credits like giving beneficiaries to high credited users. Additionally the manager can remove irrelevant or obsolete questions from the systems knowledge repository. Eventually he/she can help categorizing topics.

Especially the aspect of removing (or at least marking) obsolete messages (as also written in the next section) is a critical aspect of any KM system as analyzed in details by Stewart et.al [113]. Knowledge can be obsolete by many reasons: Especially in the IT domain, certain knowledge is often only useful for a small period of time. Moreover new knowledge might replace older knowledge, e.g. because specific products are replaced by new ones. Hence the KM System has to support the project manager (and also the user) to remove such outdated information.

Additionally on posing new questions the user should optionally have the opportunity to add constraints to the question. Such constraints could be: How long is the question relevant? Is the question related to a specific system or product (e.g. this question is related to a problem occurring with Oracle 8i; when updating the system, this question will be most probably not relevant any longer).

A further interesting aspect is, that an analysis of the questions, answers and discussions in the system can give additional information for management needs. To name a few: The type and number of questions (and answers) can give hints about the core knowledge but also about lack of knowledge. Moreover the management can view which questions are unanswered and how important those unanswered questions are. Significant knowledge and information deficiencies can be detected that way.

A key question in KM is where knowledge is located, which gives the management the opportunity to promote colleagues or help find fitting co-workers for projects. This is concluded in:

> "These knowledge-able people are also very mobile. When a person with critical knowledge suddenly leaves an organization, it creates severe knowledge gaps
>
> [. . .]
>
> Knowing what employees know is necessary for organizations to create a strategy for preventing valuable knowledge from disappearing. Knowing who has what knowledge is also a requirement for efficiently staffing projects, identifying training needs, and matching employees with training offers." *Rus et.al. [91]*

9.5.7. The System Role

The system has to perform several functions like evaluating new questions (as mentioned above), managing the repository, calculate credits and to perform archival tasks for old questions. Moreover an open interface should be implemented (Webservice) that allows to develop software agents for specific purposes like:

- Integrate different KMS domains (servers).
- Extract a “knowledge directory” to ease the access to the knowledge repository.
- Suggest connections between topics (make references between similar topics)
- Try to find obsolete topics.
- Automatically build a “newsletter” for the registered users.
- Even enhanced “intelligent” features could be imagined: The agent can try to find a user to answer open questions by analyzing similar questions and who answered them.

However, the interface should be generalized and open, so that it is easily possible to “plug in” new agents that cooperate in analyzing the knowledge repository. Of course each agent has to have an responsible user to report the results to. Also some problems can not be solved by the agent and final decisions have to be met be this person.

So to conclude: The system role is to support the project manager and the user as described in the previous sections and additionally support an open interface to the KM system, that allows flexible addition of functionality.

9.6. Other Aspects and Definitions of Knowledge Management

9.6.1. Different Viewpoints toward Systems

In fact the term KM can be seen from very different perspectives. The concept mentioned here is a rather narrow perspective of KM. In other publications, the KM term is used in a much broader sense, including cooperation, communication [91], document management, ontology aspects including also the important aspect of *knowledge preservation* [2]. So from this perspective many concepts in this thesis can be seen under the light of knowledge management, including the very critical problems of preserving digital information (see chapter 7) or the ideas to unified information access (see chapter 11). Another term used is *knowledge capitalization* and should be cited here, as it describes an important topic. It starts with:

> “[. . .] to reuse, in a relevant way, the knowledge of a given domain previously stored and modeled, in order to perform new tasks.” *Abecker et.al. [2]*

and continues with

> “[. . .] an Organizational Memory[4] should also support knowledge creation and organizational learning.” *Abecker et.al. [2]*

The first is one of the central ideas in this thesis and I describe the problem and suggest concepts in many chapters. But I believe, that the term KM is not really appropriate, and that is why I include this topic in the chapter 8: “Information Management”. The second one is a clear example of what my idea of KM is, and what the concept described in this chapter should fulfill. (In fact the system suggested here goes beyond this function, but knowledge creation and organizational learning are key functions).

Moreover there is the aspect of information reuse e.g.:

> “Case based reasoning allows to reason from experiences and cases already encountered, in order to solve new problems.” *Dieng et.al. [30]*

This is a second central idea of this thesis, even if not seen in the first line from the KM aspect. Information reuse and keeping the access to older information resources available is a daunting but very important task. I believe, that there is much to learn from previous projects, especially also in the university context, where the “corporate memory” is not so highly expressed as the personal fluctuation is by principle very high.

However, as nearly all CSCW efforts can be seen as KM [91], I see two uses of KM: first of all concrete actions taken to acquire and manage knowledge of persons in organizations, this is mentioned here, and secondly as an idea flowing parallel to all CSCW ideas. I like to keep them separate as the second use can be regarded so generally that it might be set on top of every system, so that every IT system might become a KM system. It might be doubtful, if this is a desired goal.

9.6.2. The CSCW System as Data-Basis for KM Systems

Yet another way of approaching to the KM problem is possible, or even necessary. Even considering, that the question based nescience KM system is not implemented or part of the OSWP or any other CSCW system, the data stored inside of a CSCW system is highly interesting for KM purpose. It contains information about successful

[4] This is a term introduced by Abecker et.al. and describes a specific KM scenario

projects as well as about less successful projects; Following the tasks or the activities of the collaborators it allows to extract information about the knowledge and skills of the users of the system. Moreover, as such systems are intended to be used as resource of management applications, these resources (which might be office documents, application code, databases and so forth) include essential knowledge of the institution waiting for reuse.

What problems do we expect, when trying to reuse the mentioned data in a KM context? First of all, there is the question how to reuse data from a technical point of view; this means: is it possible to extract the information out of the CSCW system automatically without manual activities? The problems and concepts concerning these technical and semantic problems will not be repeated here as they are analyzed in detail in the chapters 6,7 and 8.

The second and maybe more complex problem is to offer the KM (or any other system that is "interested" in the stored information) appropriate search and retrieval mechanisms. This is particularly a problem as various types of informations (with completely different structures) are stored and managed inside the CSCW application(s). Just to mention a few: project data (project core information, task information, financial data, etc.), resources (office documents, XML documents, databases, etc.), communication data (discussion board messages, email messages, instant messenger data), structured data (data of users and customers like contact information, skill information, and so on). Unnecessary to say, that the mentioned information is very different in the syntactic as well as in the semantic perspective. To open such a system to a KM system a unified access (search) mechanism is desired, or even, required. This means, that a general search and retrieve interface has to be defined, which allows a unique query to all parts of the CSCW system.

Unfortunately the problem is not solved by implementing such a unique search/query interface[5]. The next problem appears, when we think about the results sets returned by the search/query interface. A unique result set will be difficult as the data is very different in semantic and syntax. I suggest a solution that combines different approaches:

- Most important: the answer has to be given in an open and easy to parse format, hence XML is suggested, possibly consider the use of *webservice interfaces* [127] .
- This XML format has to be enriched with metadata information using a standard like *resource description framework* (RDF) [87]. This allows to enrich the content with standardized metadata.

[5] Not to mention all technical problems of such a global search, starting with keeping indices up to date, performance aspects, ...

- “View information” should be provided. As the KM application receives data in a format that might not be supported directly, information how to display this information should be available. As the data comes in XML format, *extensible stylesheet language* (XSL) [136] stylesheets could be defined that support visualization of the information.

This, or a similar approach could help to connect KM systems to the data pool in CSCW applications. Even if the KM application is part of the CSCW application, these problems exist and a similar implementation could be considered.

9.6.3. The Next Step: AI, Expert Systems. . . ?

One might argue that the next step (or following my description above — also seen as the next viewpoint) of information and KM systems might be expert system based, or more generally the implementation of AI “artificial intelligence” to such an information system, whatever exact meaning the AI term might have. I think the perspective is correct, that structured knowledge and data bases might offer an interesting playground for “intelligent tools” that try to extract new knowledge or new relations not deduced before. However these fields of expertise (Data Mining, AI) are already too far away from the core topics, so I will not add specific ideas about these possibilities except two remarks:

The first remark is a technical one concluded by an insight: As I described the KM system above, open standards in general are important, and more specific an open interface (e.g. using webservices) is suggested including the idea, that “intelligent agents” could plug in there and perform whatever desired. This mechanism can be exploited by any tool. The insight might be, that the suggested project information and KM system can be seen on the next level of abstraction as a resource system for high level tools (like databases are today). Maybe we will use such systems in 5 to 10 years comparable to the ubiquitous use of databases today, which are seen more and more as cheap basic infrastructure available everywhere.

The second remark considering AI is to ask a question about the goal of such a system:

> “It is time to recognize that the original goals of AI were not merely extremely difficult, they were goals that, although glamorous and motivating, sent the discipline off in the wrong direction. If indeed our objective is to build computer systems that solve very challenging problems, my thesis is that
>
> $$IA > AI$$

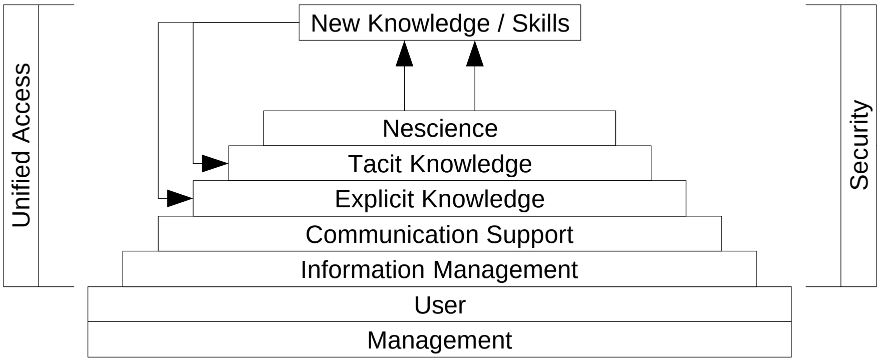


Figure 9.3.: This figure illustrates how a complete KM setup fits together. The basis is a clear concept of the management as well as good interaction with the potential users. Information management and communication, are the next essential factors, followed by explicit and tacit knowledge. Nescience is a clear consequence of all knowledge based processes! Based on the ideas described here, the KM system manages and generates new knowledge. Security and a unified access are factors to be taken into consideration at each step of a KM solution.

> that is, that intelligence amplifying systems can, at any given level of available systems technology, beat AI systems. That is, a machine and a mind can beat a mind-imitating machine working by itself."
> *Frederick P. Brooks, Jr. [19]*

I believe, that this is a pragmatic but a very engaging statement. It summarizes the intention of this thesis, namely to build complex IT infrastructure that supports groups of collaborators to manage and organize project knowledge and resources by providing universal access. Universal access in terms of a highly integrated workspace, access from anywhere and access for non-expert users. Hence this system can be seen as described by F.P. Brooks as an intelligence amplifying system for project collaboration.

9.7. Risks of KM Systems

As Stewart et.al.[113] note, there are assumptions underlying the management of knowledge, that are not often discussed, but may be critical in deciding whether KM strategies are useful and might support project work. Four basic assumptions are analyzed: "(1) knowledge is worth managing (2) organizations benefit from managing knowledge (3) knowledge can be managed (4) and little risk is associated with managing knowledge."

To go into details: Following the arguments in section 6.6 it is assumed here, that the questions whether knowledge is worth managing and if organizations benefit from managing knowledge is true in many cases. Of course it should be remarked, that there are situations where the installation of a KM system does not seem to be appropriate. Just to mention a few: KM approaches like the one suggested here need a "critical mass", a minimum number of users in the KM "community". Moreover there are surely enterprises where knowledge is not a system critical factor.

The issue if knowledge *can* be managed is difficult to answer. As mentioned and analyzed in [113, 6] "the management of knowledge is substantially more difficult than managing physical assets." To answer this question, many authors suggest to differentiate between two kinds of knowledge:

> "There are two types of knowledge: tacit knowledge and explicit knowledge [...] Tacit knowledge is the form of knowledge that is subconsciously understood and applied, difficult to articulate, developed from direct experience and action and usually shared through highly interactive conversation, storytelling and shared experience. Explicit knowledge, on the other hand, is easy to articulate, capture and distribute in different formats, since it is formal and systematic." *Sunassee et.al. [114]*

So the question can be divided into two questions: the management of explicit knowledge seems to be mainly a question of successful implementation of an information and resource management strategy. This is described mainly in the "information management" chapter 8 and in the description of the OSWP system in chapter 13. More difficult is the management of tacit knowledge. So the knowledge management as suggested in this chapter is mainly designed to deal with the practical problems of managing the latter kind of knowledge. Furthermore it is obvious, that KM does not come for free [91], which is the reason why many big companies have meanwhile installed specific teams as well as *chief knowledge officers*.

Besides technical issues many "psychological pitfalls" exist. The most critical is the question of user acceptance: This is a problem related to the described CSCW implementation, as there is always a momentum away from new systems. So a new system has to be propagated in two ways: First of all there should be a clear advantage to each user compared to the old system and secondly there must be a clear position of the management to use this systems[6] Already problematic for CSCW systems, these arguments are even more critical in KM system implementations, as the user might be afraid, that his or her knowledge should be included into the KM system, and then his or her value in the firm becomes less important.

[6] This includes, that the management itself uses the system. It is not acceptable, that e.g. the management implements a new CSCW system and does not use it by themselves!

Therefore each employee must have the feeling, that the KM system is useful for him or herself (directly) and a "bottom up strategy" is suggested. The implementation must be done carefully — also because the KM system will not be able to replace the work of skilled employees — the opposite is true: A KM system is a vivid system and needs continuous input as well as evaluation of the content. The system will stay as good as the users are who work with the content. This fact must be clearly disseminated to all employees.

A second psychological factor is the problem of possible information overload. This is a concern of many managers as analyzed in [6]. The design of a CSCW and/or KM system might not be simply to install *yet another desktop/web application.* As already discussed at various locations in this thesis, CSCW and KM are "holistic" approaches. The user should be integrated and well trained, and the particular situation of the company must be taken into consideration. About all it is important to build a unified access "portal" to all groupware applications. The user must be able to get a clear and clean (not overloaded) overview over the currently available new and important information by starting one application or open one intranet portal. Otherwise the user will either get confused, overloaded or will not use one of the systems.

A last risk factor should not be omitted: Information or knowledge, that is stored in computer systems can be stolen, abused or might get lost by technical problems. Especially the first issue is a complex one and may seriously damage a knowledge-based company. No simple strategy can be suggested here, except that this factor is very important and the implementation and installation of any KM system has to take those security problems seriously into consideration.

A illustration of the complete KM setup is shown in figure 9.3.

9.8. Popper's "3 Worlds" — A Digression

Karl Popper introduced a concept of *three worlds* into the discussion of the mind/body problem (see for example [92]). As far as I understand it, this concept is less a new contribution to an ontological discussion lead for centuries, with references from Descartes over Kant to the "Wiener Kreis" and positivism, but much more a concept to discuss recent problems in theory of science, brain research and epistemology [83][7].

Popper's theory is to categorize the world into three parts he calls "World 1", "World 2" and "World 3". Figure 9.4 illustrates this idea. World 1 contains all physical objects including the relations between those objects. World 2 is the world of personal experience (the *ego*). This World 2 is not accessible by others then by one-self. Most important is the concept of World 3: it contains ideas, (scientific) theories, the cultural heritage. Of course, those ideas and theories might be written down in books or are

[7] I refer to the german translation [84].

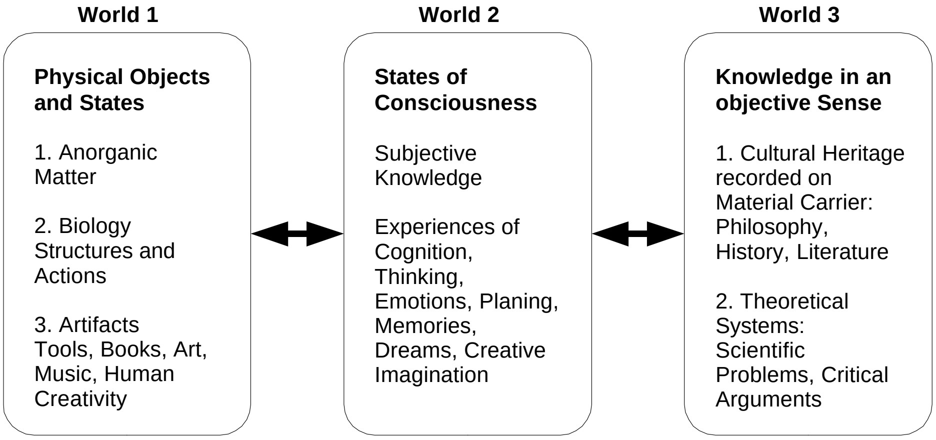

World 1
Physical Objects and States
1. Anorganic Matter
2. Biology Structures and Actions
3. Artifacts Tools, Books, Art, Music, Human Creativity
World 2
States of Consciousness
Subjective Knowledge
Experiences of Cognition, Thinking, Emotions, Planing, Memories, Dreams, Creative Imagination
World 3
Knowledge in an objective Sense
1. Cultural Heritage recorded on Material Carrier: Philosophy, History, Literature
2. Theoretical Systems: Scientific Problems, Critical Arguments

Figure 9.4.: Karl Poppers concepts of three "Worlds"; Note the interactions between the three worlds! (Figure translated from Eigen et.al. [35].)

stored on a harddisk, which are again World 1 objects. World 2, as representing the *ego* can be seen as a *mirror between World 1 and 2*!.

This very idea attracted the brain researcher John C. Eccles, and he tried to locate functional representation of the three worlds in the brain [84].

Besides those philosophical and neuro-scientific ideas, the parallelity between the concept of data, information (as outlined in section 4.1.2) and knowledge (as described in this chapter) and Popper's three worlds was appealing to me. Figure 9.5 should illustrate this idea. The first step in the diagram is not completly correct: in fact the term *data* is introduced a little more subtle in section 4. So the illustration here is a little simplified. However, World 2 is the data influenced by the *ego*, which creates information out of it. As defined, information is data with meaning, created by an operational system, and in this case, the human is the operational system. This information is not accessible directly (which is the main problem in teaching).

The next step from information to knowledge is *tacit knowledge*, as explained earlier. Tacit knowledge is somewhere in-between World 2 and World 3. Most importantly is the transition from World 2 to World 3: As soon as this transition is done, information and tacit knowlege become *explicit knowledge*. As Popper expressed it: "By application to World 3, our dreams are corrected continuously, until they finally can become concrete."[8] *Teaching* on the other hand would be the creation of a valid World 2 by using World 3 knowledge eventually recorded in World 1 objects.

The ideas in this chapter as well as in chapter 11 will hopefully help perform this transition from World 2 to World 3, and the suggestions made in the earlier chapters

[8] Citation translated by the author. Original text see section A.3 on page 202.

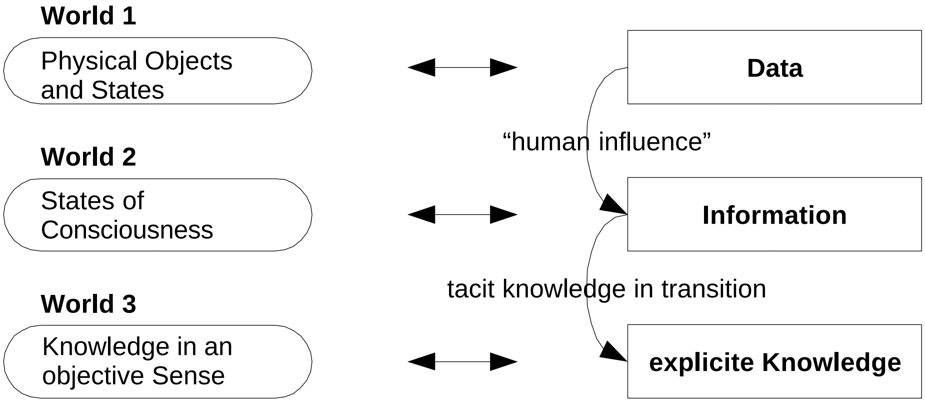


Figure 9.5.: Relation between Popper's three worlds and the concept of data, information, tacit and explicit knowledge..

about systems, information management and longevity of digital information will support World 3 expressed in World 1 artifacts to stay accessible during a long period of time.

10. Project Cooperation in Dislocated Environments

10.1. Introduction

Project cooperation, management, coordination and controlling in distributed environments offers particular problems unknown in traditional projects where all project partners are located at a specific place. Especially the Internet increased the number of such projects and will continue to do so in future. This chapter will give a brief overview about the problems and challenges of such projects as well as suggestions for solutions to the most common problems including a reference to the *Open Science Workplace* software.

As introduced in chapter 3, projects with dislocated scenarios increase the complexity of management and monitoring significantly. At this point it should be explained in more details what distributed scenarios are:

- Projects where co-workers are located in different cities or countries.
- Projects with inter-disciplinary character; as often people from different institutions, even when located in the same city, work in different offices.
- Projects where home-working/tele-working is involved, or an essential part of the project has to be done outside the office (Consider external working IT consultants).
- Projects where external companies or specialists have to perform specific essential tasks and need to be integrated.

In short: As soon as it is not sufficient to walk to the next door to communicate with the co-workers one might see the scenario as dislocated. As a consequence, many projects done in the high-tech environment can be viewed (at least partly) as dislocated and could hence profit from supporting (IT) infrastructure.

Additionally one will detect, that the lessons learned from distributed projects can also increase the quality of very traditional projects, at least with regard to traceability, quality and risk management and project monitoring. The increased traceability again allows easier integration of new co-workers, hence more flexible human resource management is possible. So this chapter will give some side-notes and analyze experiences that have not been mentioned in the previous sections, as well as analysis of usual pitfalls and suggestions to avoid them in selecting fitting CSCW systems are given.

10.2. Types, Sizes and Workflow of Projects

Projects can be distinguished by different properties. E.g. by domain issues: IT/KM projects like software projects, cultural projects like a virtual museum or e-learning, "pure" science projects (e.g. in particle-physics). Or a categorization following the organizational type can be done: Projects at universities, open source organizations, business projects.

However, I believe that the underlying essential principles are the *project size* and the *workflow setup* of the project. The parameter *project size* is immediately evident as with increasing project sizes communication and decision processes become more complex. This is also true for steering and keeping overview over the work of all participants. The more subtle, but equally important parameter is the *workflow* setup of the project, unfortunately this parameter is harder to describe. What is meant by this? Workflow setup means: how are important processes organized in terms of persons involved and rules defined. Some examples:

- New resources like computers, software are required for the project: who decides this, who is responsible for the buying and setup, how formally is the decision.
- How is the hierarchy among the co-workers of the project: are there multiple hierarchies with strict reporting schemas, or is there a very flat hierarchy with rather "chaotic" information exchange.
- Is there a clear top down steering of the project or are there several equal partners that have to agree with certain decisions.

Those are just examples. In reality it can be seen, that traditional business and project planning is rather strictly organized and localized. More flexible strategies evolved recently, supported by moderating communication technology like the Internet. This is a result of the need to react (even within a project) to changes on the market. Flat hierarchies, homeworkers, distributed teams (also to include specialists that might not be available in the company) are a consequence.

The experiences with the Iranian/Austrian cooperation also showed, that two equal partners with equal rights for decision are sometimes necessary and can work but under specific condition. These are, e.g., one-to-one communication between the groups, good communication channels, one person decision on each side. In some situations, this turned out to be a problem in the *Open Science Workplace* project with the Iranian partners. A more formal hierarchy is still existing there, and sometimes even small and rather unimportant decisions have to be taken by the project leader, who is not always available immediately

As a consequence it turns out, that the workflow conditions of a project is a crucial factor, even when it is hard to describe formally. Hence well designed IT support for modern projects cooperation with flat hierarchies, high mobility of workers, dislocated scenarios, complex resources can leverage the daily work in such projects. The *Open Science Workplace* as described in more details in part three of this thesis, as well as in the scientific articles in the appendix, is a tool that supports particularly such project types.

In the next sections the detailed requirements of such typical modern workflows are analyzed and described. CSCW tools should support the project members in those areas. Traditional management and traditional tools can help for projects with a high degree of centralization. Todays problem solving often required de-centralized setup, hence the keys of such a methodology that supports this concept is described here.

10.3. Communication

10.3.1. Communication Channels

10.3.1.1. Introduction

In the daily work of a project it turns out, that there are at least three crucial factors for success:

- Clear responsibilities and motivation
- Good information and data management
- A well planned communication strategy with appropriately integrated user interface

While the first aspect is discussed at various locations, and the second was the main issue of the previous sections, integrated communication support was not yet focused. First of all, it should be noted, that the communication channels should be part of the project planning. If this is omitted, ad-hoc communication channels establish with the consequence, that users use the systems they know, some users are cut-off from important information and others are flooded with unimportant data.

10.3.1.2. Synchronous Communication

Synchronous communication are personal meetings, telephone, chat. The problem of these communication events are obvious: all participants have to be present at the same time; with more than two participants technical problems might emerge. So as a consequence in many projects either synchronous communication is avoided as far as possible[1] or alternatively — especially in formally organized projects — there are held so many meetings that the "real" work is lost in discussions.

Besides the mentioned disadvantages there is the problem of missing traceability of personal meetings and telephone talks, unless protocols are written. Chat systems also might offer the possibility to store the discussion and keep it accessible.

Nevertheless cultural problems (as described in the next section), *require* regular personal meetings which are extremely necessary, especially when project partners from different countries are involved. So the importance of synchronous communication should not be underestimated. Reducing communication entirely to asynchronous channels will most probably lead to misunderstandings and problems in the project. Hence it is recommended to define a clear meeting schedule also taking milestone dates into account. This procedure gives the project a clear communication flow as well as a transparent structure for all collaborators.

10.3.1.3. Asynchronous Communication

In asynchronous communication we have again to differentiate between *push* and *pull* systems. Push systems are, e.g.: email (personal), mailing lists, short message services (SMS); pull systems are, e.g.: instant messenger, nntp or web discussion forum, and also web-based-publication systems like the recent Wiki applications[2] and versioning systems like CVS.

It is interesting to watch that simple email is the most used asynchronous communication systems in projects. The consequence is, that often some persons are excluded from important information, or others are flooded with unnecessary emails [123]. Moreover it is difficult to enter discussions that are already running. Consider the following example: A and B discuss a problem via email. After some mailings they find out that it is eventually a good idea to include C into the discussion. Now it is hard to include C, as it is difficult to share all sent emails. Again more problematic is the situation, when project members change, or new project members have to get into the project.

[1] Particularly when considering personal meetings, as they are accompanied with usually large expenses and cost much time.

[2] A Wiki system basically is a content management system, where every user may edit all webpages online, and information sites are generated in a collaborative effort in a very democratic way. There are Wiki systems for developer and user communities of open source projects as well as a complete encyclopedia.

Even worse is the common usage of email to share resources by sending documents or other binaries. A lot of network traffic and storage capacity is wasted, versioning is not possible, and users are distracted by tons of email, that are not read, and dozens of binary attachments that are not opened.

Not to be misunderstood: email is an excellent medium, but it has to be *used appropriately*! What can be seen as "appropriate usage"? This is communication, that is *really* only a matter of few persons. Essentially — as a rule of thumb — information exchange, that could also be done via telephone; but email is preferred as others are not distracted by a telephone call. In all other cases pull systems should be implemented and used! The practical problem is, that there are often project members that are not familiar with pull systems. Even worse, often managers are in this group, and if the manager starts sending huge documents via email, and abuses push systems for discussions, the project communication is doomed. This was observed at one of the literature projects, where implemented pull systems failed by that reason. *If* a pull system is established, many positive effects can be seen:

- Everyone has access to the message and resource repository, also from everywhere (web), depending on the system and security settings.
- It is possible to trace back discussions any time, when messages are not automatically deleted
- The repository is usually search-able.
- Project members can browse the repository when it fits to their workplan, and are not continuously interrupted by email and other push messages.
- Also different versions of documents and discussion about the contents can be traced easily
- Waste of network capacity as well as unnecessary storage space is avoided.
- Server based systems can be administrated far better then email systems[3], and also backup is possible without problems. Hence a client destroyed by a virus or hardware defect does not affect the work of the person.

Consequently *before the project starts*, not only email accounts have to be setup, but also a proper pull client installation has to be done[4]. This has to go hand in hand with

[3] At least when POP email protocol is used. The situation is slightly different when using IMAP or Notes/Domino.

[4] If it is definitely not possible to install and/or use pull systems, an alternative could be the use of a properly configured mailing list system. This has the advantage over "normal" email, that it is easier to filter mailings from a mailing list into specific mail client folders, hence have a discussion like thread view. Moreover many mailing list systems offer archival and search of mailing lists. Though a pull clients is more advantageous in my opinion, this is a valid alternative.

training for unskilled users, particularly of managers. It is of highest importance, that project managers and middle management uses the communication channels properly!

10.3.2. "Two Cultures" Revisited

Having implemented stable and integrated communication facilities, there are still important risks to expect: Communication in projects with the described characteristic has different meanings, as it becomes more and more important to mediate between cultures. Then again, the question arises, what is understood as *culture*. The first thing that comes into the mind is *culture in the sense of different countries, religions* and the like. But the second point of view is: *cultures as a result of academic tradition*, of "schools". Having interdisciplinary projects between technicians and German literature and language scientists for example, show dramatically, that here different cultures are opposed and different languages spoken. On the other hand, the projects with Iran demonstrated the differences in communication virtually from the "original" definition of the word "culture". At the end of the day one has to realize, that both definitions are important and both types of potential culture differences have to be taken into consideration in a world with the necessity and the will toward of interdisciplinary work[5] as well as in a globalized community. C. P. Snow put it like this (see also [88]):

> "I believe the intellectual life of the whole of western society is increasingly being split into two groups. When I say the intellectual life, I mean to include also a large part of our practical life, because I should be the last person to suggest the two can at the deepest level be distinguished [...] Literary intellectuals at one pole — at the other scientists [...] Between the two a gulf of mutual incomprehension — sometimes (particularly among the young) hostility and dislike, but most of all lack of understanding [...] This polarization is sheer loss to us all. To us as people, and to our society. It is at the same time practical and intellectual and creative loss, and I repeat that it is false to imagine that those three considerations are clearly separable." *C. P. Snow [107]*

This ideas of Snow were published in the 1960's and often criticized since then. We might add today, that the "the young ones" not even feel hostility and dislike, because this would assume, that there is some interest in *the others*. I have the feeling, that (particularly from the side of the technicians), there is simply ignorance and lack of knowledge. Who might have guessed, that it is possible to study physics and chemistry

[5] This is definitely not only an issue for information technology as engineering discipline for different user groups, but for nearly all scientific disciplines. One might think of medicine and ethics, economics and psychology, brain research and philosophy and so on.

without having the slightest idea of epistemology. The *other side*, the philosophy, sociology . . . seems to be more and more interested, but science became so difficult and distracted, that it is in fact very hard even for the interested, say philosopher, to get an idea about contemporary physics, cosmology, brain research and science topics like that.

These arguments might be true in detail or not, I believe Snow at least describes a vivid danger, maybe becoming more problematic today, then when he was writing it four decades ago. This is unfortunately not the place go more into details about the necessary interaction between *literary intellectuals*, as Snow calls them, and *"real" scientists*, as the technicians are mostly seen today. Technological breakthroughs are valuable, philosophical and sociological seem not to be of such importance. But if we come to the conclusion — and I do — that there is the necessity of the interaction between different cultures as defined above, one has to take those issues seriously if the project should become a success.

Having realized the potential problems it is nevertheless difficult to find the right answers. The most important thing might be to address the problem of *different languages and cultures* from the very beginning and even more important: even if the other side, the project partner, signals, that he or she completely understood the concepts suggested, one has to be very skeptical about it. I realized the following scenario occurring several times — particularly in the German literature and language projects: technical suggestions were made, and the technicians tried to explain those technical details on a very basic level. The literature and language scientists replied to have understood the suggestions with all consequences. But when for security reasons, additional questions were posed from the technical side, we realized, that not all consequences were fully understood and parts of the problem had to be explained again. Of course the same happened in the opposite direction!

Consequences are clear, but not always easy to execute: *far more time* has to be reserved to discuss essential concepts and even more important it turned out, that it is very valuable to have *prototypes*, or examples, even if they are very rough. But having a prototype allows a much better translation of technical details into non-technical speech and vice versa: having simple examples (e.g., how do the illustrations really look like, how is the quality of the multimedia material, read a text, not describe it . . .).

Having tried to address all those inter-cultural problems, one might have forgotten even the most important one: The *culture clash between generations.* This is not only a sociological problem, but it is very important for technical issues like: Will this or that communication technology be accepted or not? The best CSCW tool will fail, if the managers (of a generation not accustomed to computer systems) will not use those systems, the communication media, do not read emails and do not respond to instant messengers.

10.4. Management and Steering

Following the idea, that project characteristics have changed dramatically and will continue to do so, this must have consequences for the management and for steering activities. A centralized setup can be very functional with traditional located projects, but will most probably fail, when projects are de-centralized, very flexible and there are many partners involved. Besides factors like flat hierarchies and the capability of project managers to really delegate responsibilities, the greatest danger might be, that the project manager looses the overview. This is especially a problem, when one manager has to control multiple projects. To get some details, at least the following parameters should be held under control:

- Human resources: who is available, best suited for what tasks, how expensive...
- Project size, tasks, subtasks.
- Clear responsibilities for tasks
- Progress status of tasks.
- Finances (time and cost management of co-workers)
- Resources: is it possible to access the current versions of all relevant resources?
- Versioning of Resources.
- A brief (visualized) overview about the project status is helpful, particularly when multiple projects have to be kept under control.
- Communication channels, including announcements that reach all required members.

One should not forget, though, that meanwhile not all projects need or allow explicit steering like traditional projects do. This is particularly true for open source initiatives like BSD, Apache software pool and others. Although there are project leaders, they usually do not have the option to steer, simply because most programmers work voluntarily and for free, hence pressure has to be avoided and is often contra-productive. Controlling such projects is usually a matter of getting a critical mass in the developer community, to avoid the dependency on one particular programmer for solving a particular problem. This is, what happened, e.g., with Linux in many areas: different suggestions for solutions are made, and steering means to select the best one. Nevertheless it is always necessary, that there is (usually) one main developer or project leader who does integration and programming of parts that are not done voluntarily. Those developers often are located on universities or paid by companies like Linus

Torvalds. A good analysis about the processes in such big open source communities can be found in [139].

And last but not least, the project manager has to be aware of the problem of possible information exchange. Projects might need to include new groups, or companies might need to exchange data. So the project data, information and knowledge should ideally be managed in a way, that cooperation with others partners as well as with other IT systems is possible even if they will be integrated later on.

In the *Open Science Workplace* (OSWP) project, the organizational backbone is the structuring of projects in a tree-like form: In one OSWP instance multiple projects can be defined. Each project may consist of tasks and each task again may be composed of sub-tasks. Additionally one-person responsibility is an essential idea throughout the OSWP system: There is *one* project manager and *one* responsible project member per task. This structure is not only useful to organize multiple projects: Other modules like the resource management, communication (discussion) facilities and others are associated with this project/task tree. This is a natural way to organize project-related data without the need to implement different structures for different systems.

10.5. Workflows and Hierarchies

10.5.1. From Traditional Management to Virtual Companies

As mentioned earlier, traditional workflows often base on three facts

1. Multiple levels of hierarchies in the organization.

2. Central storage of information and data.

3. Co-workers are located in one office, or at least in one building.

This is still true for many projects in enterprises, universities and other institutions and seem to continue. Very progressive theories suggested ideas like "virtual companies", particularly during the "dot.com bubble" in the end of the 1990's, meaning, that only a very small head of the company is available with fix staff to organize and steer projects and the operative business, the "real work" is done by groups of freelancers that are organized in an ad-hoc manner [120].

In fact it turned out, that those *virtual companies* do not work as expected. The amount of coordination becomes too big, company specific knowledge and tradition decreases. Moreover the expected flexibility was not achieved, as a flexible staff located in the company is ever faster, then fresh recruited staff for every new problem.

10.5.2. Working Flexibility

Finally it seems, that the current management strategies are oriented somewhere between a very traditional setup and virtual companies. For day-to-day business, traditional management usually works fine, but complex interdisciplinary work as well as globalization require new strategies as mentioned above. Dislocation is often a consequence of such attempts. Moreover some companies (and universities) tend to create spin-off companies for very new technologies or products. Also it is often impossible to avoid fast changing teams, when the business is to build recent technology.

All those developments require powerful and flexible strategies with regard to information, knowledge and human resource management as well as the described mediation between different cultures. It is interesting, that companies have realized, that it is necessary to have specialists to mediate their products and services so that potential customers become interested and understand the value of the product. This is the business of marketing and advertising agencies, including graphical designers, web experts, musicians, writers and so on. But it seems to be hardly understood, that current projects (especially big ones) would require a mediator who leverages communication and human resource management between different parties and interest groups. This should not be mixed up with the role of the project manager, though he or she often has a similar role, but in bigger projects it could be beneficial to have a project manager for operative business and a project mediator/consultant responsible for communication and mediation.

Some companies at least understood, that someone should be mainly responsible for information infrastructure and communication facilities and created the position of a *chief of the information office* CIO [6]. Maybe it will be realized soon, that also a CPMO will be a good idea — a *chief of the project mediation office*? Although it might be assumed that in some cases, this job could be better done by external consultants as they are not too far integrated in the company's structures.

10.5.3. Changes in Management Principles — Flat Hierarchies and "Democracy"

Drawing the conclusion of the dynamic behavior in current project work, it turnes out that clear responsibilities are required, whereas strong hierarchies are not. Projects that are successful and work in a fast changing unpredictable market[6] should base on a more democratic model. This requires excellent selection of project members and even more important the willpower to let the project members take decisions in their specific domain. However, it is very important for each member to distinguish between

[6] Speed *per se* is no problem, as long as it is predictable. The problem of many projects today hence is not the speed of the development, but much more the unpredictability of the direction of the developments.

problems that are better decided by oneself, and problems that are far reaching and should definitely be discussed in the project group.

All of these problems require IT support, that makes decisions traceable, allow discussion on a democratic level and allow the project manager to detect whether specific steering mechanisms are required. Moreover the IT infrastructure should be open, insofar, that practically each project member can access all informations of the project. I do not believe in strict access right policies. My experience is: Either you can trust the project members, than a sloppy policy is the best idea, as anyone can easily contribute and retrieve necessary information, or you cannot trust the project team, then you have a problem anyway. Bureaucracy should be avoided as much as possible.

Strict access policies may be required in operative businesses, where either many persons are accessing the system or many different roles are available. This is usually not the case in projects: the project information system should be secure for outsiders, but very open for insiders. Transparency is value, access restrictions not. Those core ideas were also important considerations in the *Open Science Workplace* planning.

10.6. Management of Human Resources

As mentioned above, there is a need to rebuild teams (even if recruiting is mainly done inside the same company) and CSCW systems should support the management of human resources. Particularly in the following areas:

1. Selection of project members (following skills, experiences, current work,...)

2. Necessity of an overview about who is doing what and about the status of the project in terms of task-person relations. As mentioned above: this should be easily possible for each project member, as problem solving requires the fast possibility to find responsible co-workers for a particular part of the system

3. Financial planning.

4. Resource management: Relations between resources, different versions of resources and project members.

The suggestions made in the second part of this thesis, as well as the implementation of *Open Science Workplace* try to support those needs.

10.7. Resource Management

10.7.1. Resource Types and Unexpected Pitfalls

Among all considerations in this chapter, one of the most important might be resource management, where particularly two kinds of resources can be distinguished: hardware resources (like computers, cars, rooms, telephones) and information/data resources. The latter was analyzed in detail in this thesis, and some references are given in the next section. The first one is obviously essential, as needed hardware, for example, should be provided fast and functional, and problems in supporting those needs can slow down a project dramatically. But it is mainly a question of good staff that deals with system administration, buying hardware, cars. . . and facility management. As this is normally a problem of the operative business, it is not a core topic of this thesis.

Nevertheless experience shows, that this is an issue often forgotten in project planning: Projects are usually located in some company, university and need resources of this institution. As those resources are understood as “available”, they might not specially considered in the planning phase. Finally it turns out, that system administrators, facility managers and others do not feel that supporting of project groups is their core task, and problems arise.

Project management has to define precisely what resources and what kind of support is required from the “hosting institution” before the project is launched; this is necessary to guarantee, that the responsible managers of those institutions provide explicit support for required services. Maybe even contracts should be signed.

10.7.2. Information and Data Resources

So having a good project planning and hosting, those resource problems should (hopefully) not occur. The main issue for the daily project work is dealing with information and data resources. The first problem of resource management was already addressed in the communication section (see page 151). Many problems can be avoided from the very beginning, when communication channels are used in an optimal way and combination. Each communication method has obvious advantages and disadvantages, and as mentioned in the communication section, pull clients are often forgotten.

For all other information resources a clean planning should be done before the project starts, to avoid ad-hoc information storage, probably even on client side. Server side storage is always suggested, as it allows central mechanisms like backup, search and retrieval and so on. It seems not to be necessary to go into more details here, as the problems of information management (see chapter 8), longevity of digital information (see chapter 7) and system decision (see chapter 6) have been analyzed and discussed in detail in previous chapters.

But one comment has to be added: as already mentioned in in the communication section, managers have to set a good example in using systems as designed! Clear strategies have to be defined and executed. Simply installing the best CSCW software (whatever this might be in the concrete project problem) is definitely not sufficient. Especially when unskilled project members are part of the project. Users have to be trained to use the systems even if it might be easier to write a note on a post-it instead of entering it into the PDA or the CSCW system, and soon the monitor is full with post-it notes, and the overview is lost. So a “post-it-forbidden” strategy (as a simple example) has to be forced even against initial resistances! Users will find out soon, that a clean strategy is more comfortable for everyone.

In the *Open Science Workplace* project, the problem of organizing resources was solved by associating them to tasks. As projects and project tasks are structured in a tree-like way, and reflect the project setup, this kind of resource management seems to be one good and natural way to solve the resource management problem.

10.8. Cost Management

Cost management is not a core topic of this thesis, but awareness should be directed to this factor, required in many dislocated projects. Usually cost controlling (in the operative business) has clear rules, where employees are located and structures of work are defined. In dislocated scenarios the situation is different in many ways:

- External experts may be included in the project.
- Different project members have different salaries.
- The financing of the project may also be “dislocated”, meaning that eventually not only one partner is involved.
- Tele/homeworkers might be involved.
- Different tax systems and other legal problems may arise, when project collaborators are located in different countries.

Those mentioned points are just a brief inspect into the various aspects of the problems to be expected. Of course, many of them need professional consulting from, e.g., tax-experts (like the last one) to be solved and are obviously not primary an IT problem. Others should be supported by CSCW systems. Particularly the issues with dislocated home and tele-workers and the problem of “different expensive” project members should be supported by ICT.

Ideally, CSCW software allows to leverage cost planning, in a way, that selection of appropriate collaborators includes the finances as a decision factor. Additionally, a very important point is to give external workers tools to register and comment their work time. This helps the individual worker to keep overview, as well as the the project manager to do the cost controlling of the project.

Furthermore, if such information is stored in the project management system, the outcomes can be extremely helpful for new project planning ("post-mortem analysis"), to reduce the financial risk in future projects.

10.9. Multi-Channel Publishing

10.9.1. Web-Access, Applications and Print Publishing

Following the suggestions described in the chapters above, all preparations have been done to allow processing data and information in various ways. Part 3 and the Appendix will give examples for accessing data and information using a webbrowser, desktop client application and using print publishing.

The use of open standards, metadata and database systems in combination with standardized interfaces like JDBC and ODBC allow to create different types of applications. Moreover, using the XML formats also allows to export the data for data-exchange as well as for print publishing. This has beed demonstrated in the German literature and language projects as well as in the *Open Science Workplace* project. In the latter one, project and user data can be exported to XML, then generating project reports for offline viewing and printing.

10.9.2. Mobile Client

The most recent "new publishing/access channel" are mobile clients, and they are very heplful for dislocated projects with users of high mobility. Using mobile clients to access information systems usually poses the following problems:

1. How to access the information system (technical aspects).
2. User interface/application problems, as a lot of different devices are available (from mobile phones to various PDAs)
3. Mobile Clients are usually not continuously online.

For the first problem, two basic solutions can be seen: The first one is, to connect the mobile client to a desktop system to synchronize data with the server. This is a good solution in many cases, but in fact not really a mobile one. The second possibility is, to use mobile phones or other wireless technologies like WLAN combined with Internet protocols to connect to the server and exchange data.

It should be mentioned here, that there are activities to standardize communication between mobile clients and servers as well as between different mobile clients for synchronizing data. As an example can be seen the SyncML [116] standard (XML based). Unfortunately by the time writing, it is not clear whether one of the proposed standards will suceed.

The *Open Science Workplace* mobile client prototype uses Internet protocols to connect to a webservice on the server for data exchange. In fact, this is an interesting solution, as implementing a webservice interface can be useful by many reasons and thus the usage by the mobile client needs no additional implementation effort.

The second problem is also a serious one: A lot of different mobile clients are available on the market: cell-phones, Palm computer, Windows CE systems and many more. It is easy to understand, that it is hardly possible for a small development team to support all of those devices by generic progamming. A good solution is the usage of the Java 2 Micro Edition (J2ME): this virtual machine is available for most of the mentioned systems, and is even often pre-installed on mobile phones. The J2ME offers different *profiles* to address devices with big variations in terms of display size, processor speed and available memory. So ideally, one needs to develop only one application and can distribute it to "all" or at least "nearly all" mobile target systems. First prototypes to access the OSWP data were promising, using different emulations of mobile clients.

The third mentioned problem offers three possible solutions: The first and simplest is, to require, to stay online, when accessing the system via mobile client. Then the "normal" application server techniques can be used, e.g., serving WAP pages. This is not always a desired solution: The second possibility is, to download the data from the server, but for *read-only* access, then terminating the connection. The last solution allows offline work *and* modification of data. This is obviously the most interesting implementation, but also the most difficult one. If it is allowed to modify the data offline, one has to consider the possibility, that different users modify the same records offline. Finally, when the mobile clients are synchronized, conflicts occur, that have to be detected by the synchronization mechanism and users have to be asked to solve these conflicts. In the first OSWP mobile client prototype we do not allow data modification by that reason, but basically the application is prepared to do this in future releases.

10.9.3. Unified Information Access

The last aspect is unified information access and information integration. Especially for mobile workers, it is very helpful to aggregate informations from various sources to a collaboration portal. This is one of the ideas of *Open Science Workplace*.

A second strategy becoming more and more important is to unify the access interfaces to different information systems. This is crucial as a simple aggregation of information may lead to the problem of information overload. For the mobile as well as for the resident or unskilled worker. A concept in combination with knowledge management is outlined in chapter 11.

10.10. Acquiring Meta-Information

At the end of the day, the project is over. And unfortunately many results of such projects are lost with the finished project. This is unfortunately true, particularly in the university context, where the knowledge and experience is very tightly coupled with persons who change very fast by system reasons. Following the strategies suggested here, should help to keep project results alive, reuse it and maybe even more important, post-mortem analysis are possible.

This means: new colleagues can learn from mistakes as well as from good strategies of ancestors, when it is possible to easily access former project information and knowledge.

Also a special "lessons learned" report should be required for every project written by the project manager after having finished the project. This is done, e.g., at IBM. When new projects are started, the new project managers can access the *lessons learned* by other colleagues in similar problem situations. Managers should encourage the implementation of such a system.

11. Unified Information Access

> "[...] 'computer science' is in fact not a science but a *synthetic*, an engineering discipline. [...] the computer scientist is a *toolsmith* — no more, but no less. It is a honorable calling. [...] If we perceive our role aright, we then see more clearly the proper criterion for success: a toolmaker succeeds as, and only as, the *users* of his tool succeed with his aid."
> *Frederick P. Brooks, Jr. [19]*

11.1. Introduction

In this last chapter, the consequences of the ideas developed above should be drawn. Using well planned information management strategy combined with open formats and standards is still no guarantee for a system, that has a high usability. Moreover a significant problem today is not the lack of information system, but much more the inflation of *different* information systems. This chapter will collect the open threads again and will try to put them together to a homogeneous easy to use framework of different information and knowledge systems.

> "The concerns related to information were primarily associated with a desire to avoid overloading already taxed users with yet more information. The concern was as much about the new information that would now be available as it was about eliminating 'old/wrong data' or knowledge that was no longer valid. This supports Courtney et al's (1997) assertion that 'omitting the unimportant may be as important as concentrating on the important' in determining what knowledge to include in KMS."
> *Alavi et.al. [6]*

In chapter 9 a new concept to manage knowledge following a demand/question based mechanism was introduced. This strategy is especially very well suited to be used as an entry-point to different information systems from the logical point of view.

Thus unified information access should be seen as a collective effort based on many strategies mentioned in the chapters above. So finally a brief inspect in mobile clients is given as well as a complete framework of an IT/KM based problem solving process will be outlined, continuing the idea started in the KM chapter, particularly in section 9.5.3 on page 135.

11.2. Information Integration: Basic ideas

One of the major goals for the question/answering system as described earlier, is to offer a user with even little computer knowledge a desktop where he or she can find her way to information that might come from different heterogeneous sources. Every user with little IT knowledge has his or her own methods to access information needed. Some scan the whole hard disk or the network for a file, while others make copies of every file they need. Even by choosing the right web search engine different preferences can be detected. And many users not even know how to find all relevant information sources! Hence to receive the information needed one usually has to contact different sources.

In a working environment often quite similar problems occur over and over again in such a way that a single coherent system with only one user interface might enable to solve these problems more easily and moreover simplify the search process. Considering such an "information portal", that centralizes the information retrieval activities of all users, there is an important "side effect": As questions are posed through one central system and answers are collected by this system, those question/answer activities can be analysed and processed by this application. As a result those activities can be used to build up knowledge that will be saved in a way that it can be accessed again. The proposed system does not only delegate queries to other subsystems and collect the answers, but in case that this procedure does not lead to a success (in terms of solving the users problem), also stores the open questions in the system and encourages staff members to answer/solve open issues. To increase the workers' motivation and to ensure the quality of the knowledge base, a scoring system is suggested. This complex mechanism will be described in detail below.

As mentioned earlier, a meaningful knowledge management software must be embedded in a worker's everyday practice. To attain a tool that will be used as the "standard information finder" a user must find her daily needs of information with "one click". It should be the top priority to make it comprehensible to the user, that using this system is the most efficient way to obtain information from different sources.

11.3. Concrete Example

11.3.1. Introduction

In this section, the concrete scenario of a user query will be described in detail as a combination of the inclusion process of different internal and external information sources and on the other hand as the building of a knowledge repository following the users interaction with the system. The users view is simple, should contain a "Google" like user interface and a step-by-step "wizard" interaction with the system to evaluate the quality of the answers and possibly proceed from Step 1 up to Step 5 with increasing complexity until the problem is solved.

In a way this “wizard” like system could also be regarded as a knowledge proxy, that guides the users through various levels of interaction with different systems, starting from the simplest and cheapest approach (internal knowledge repository) and ending with the most expensive (query of other users, or external assistance).

The following sections will briefly explain the steps toward receiving a solution to a problem a user posts to the system. Figure 11.4 gives a first overview over a complete problem solving step, that is supported by IT and KM mechanisms. This diagram is a more detailed “follow” up of figure 9.2 on page 138. This sequence diagram illustrates the sequence of the following steps:

11.3.2. Step 1: Answer given directly by Knowledge Database

The diagram 11.1 shows a typical case scenario at which a user’s query is being answered by the knowledge base (1.). After the initial installation and setup of the system, the knowledge base is empty. Over the time the knowledge base is filled with a set of questions and answers. Evidently, the larger the database the better the hit ratio for an answer to a question is expected. The user’s query will be processed by the question management engine (no other resources are involved in so far). This software module decomposes the question to a common form, so that every declension of a word has one common stem. This process is known as stemming [17].

After decomposing the query the question management engine searches in its knowledge database for an appropriate answer (2.). The system evaluates matching answers by comparing the questions, which already have been answered, with the current question. If a certain similarity is given the database delivers the relevant data back to the engine which passes it to the user (3. and 4.).

Finally the user evaluates the answers delivered by the system. If the user appraises the answer to be a solution to his or her problem, the question (as long as it differs from other questions linked to the answer) will be added as a further possible question to this particular answer. For that reason the knowledge base grows although no direct knowledge is added. The users might also come to the conclusion, that the given answers were not sufficient. This would lead the system to step two, to refine the information retrieval process.

11.3.3. Step 2: Answer given by Local Information Resource

The scenario is almost the same as in step one (again see fig. 11.1). By now the knowledge base offers no appropriate answer to the question. Either because no fitting answer was found in the knowledge base or the user rejected the suggestions. The procedure of rejecting an answer is not explicitly shown in the diagram as it would unnecessarily complicate the whole graphic. In our approach another software module,

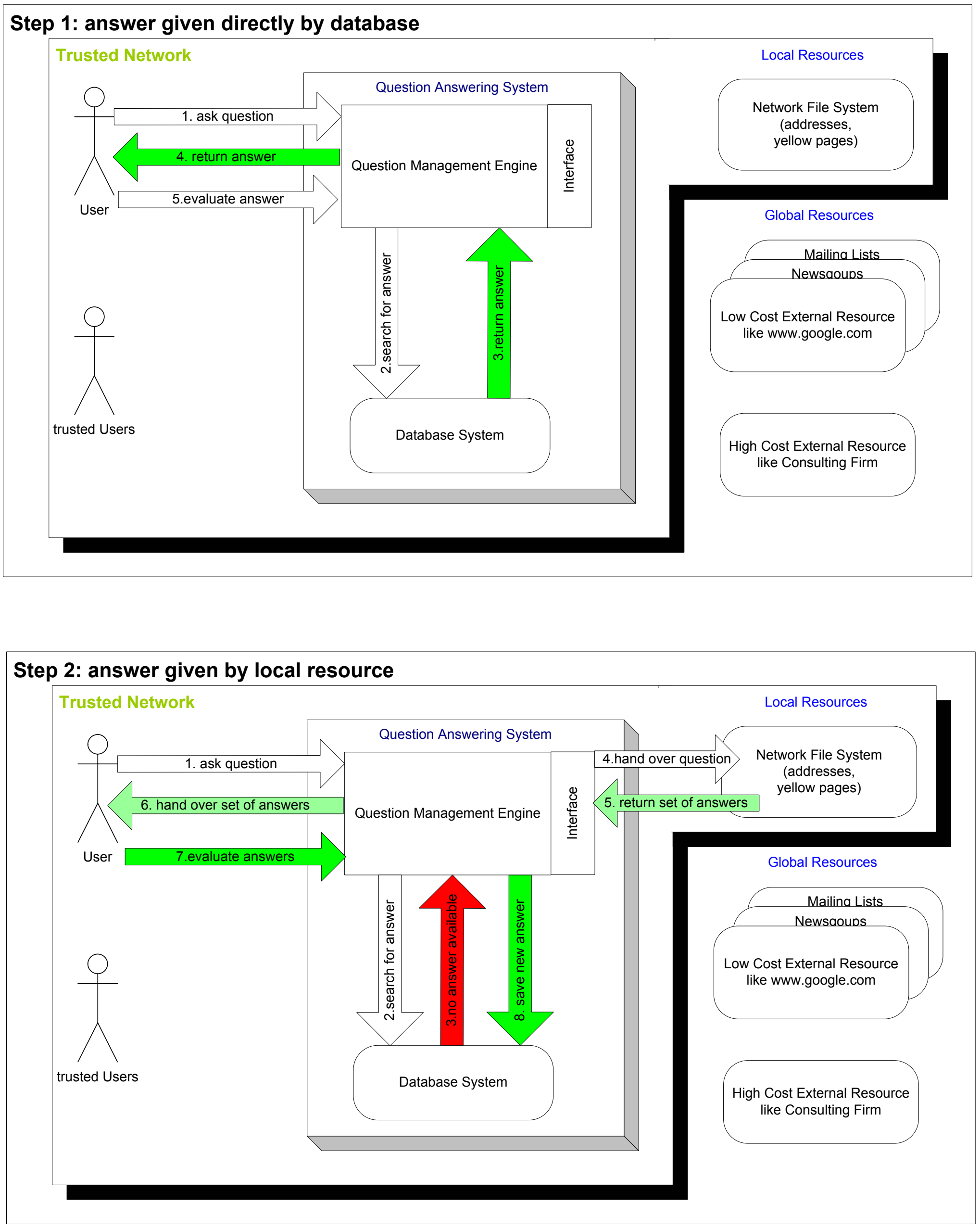

Step 1: answer given directly by database
Trusted Network
Local Resources
Question Answering System
1. ask question
4. return answer
5.evaluate answer
User
Question Management Engine
Interface
Network File System (addresses, yellow pages)
Global Resources
Mailing Lists
Newsgoups
Low Cost External Resource like www.google.com
2.search for answer
3.return answer
trusted Users
Database System
High Cost External Resource like Consulting Firm
Step 2: answer given by local resource
Trusted Network
Local Resources
Question Answering System
1. ask question
6. hand over set of answers
7.evaluate answers
User
Question Management Engine
Interface
4.hand over question
5. return set of answers
Network File System (addresses, yellow pages)
Global Resources
Mailing Lists
Newsgoups
Low Cost External Resource like www.google.com
2.search for answer
3.no answer available
8. save new answer
trusted Users
Database System
High Cost External Resource like Consulting Firm

Figure 11.1.: First and second Step in the solving of user problems

establishes a connection to other resources to find a good answer to the question (4.).

(The function of this interface and its structure will be described later on.)

A local resource can be imagined as an extension of the knowledge-database system. Local resources can be arbitrary in type, e.g., databases, file systems, XML repositories and so on. From that point of view sequence 4. and 5. in the diagram just show the access to the local file system whereas the keywords for detecting the local resource are stored in the database system itself. But still the procedure of how to access local resources, which also could be another database, is a question of software design. The design should be planed to be relatively open so that every programmer can implement a new module for the interface based on his own requirements.

Finally a set of answers will be returned to the question management engine (5.) which again passes it over to the user. The user evaluates the returned answer(s) by giving the system a positive or negative feedback. If an answer is satisfying the set of questions and the links to the relevant resources will be saved in the knowledge database. If the user sends back a negative feedback or if no answer could be found the question management engine has to proceed with step three.

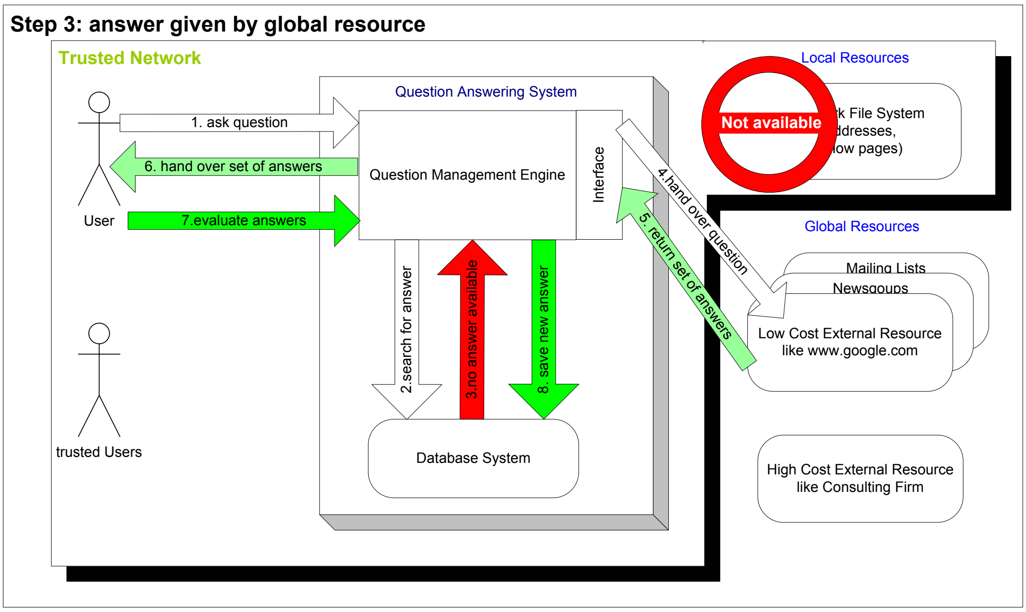
11.3.4. Step 3: Answer given by a Global (External) Resource

The third step illustrated in diagram 11.2 is very similar to the second step. Again the user formulates a query (1.) and the knowledge database has no appropriate answer (2. and 3.) or the answer is rejected by the user as described above. Now the interface-module gets into the game again. The scenario represented in step three shows the local resource as not available which could be caused by several reasons. The simplest reason is, that the interface-module has not been implemented yet. Another could be that the user, asking the question, has no right or no access to the local resources being offered.

Anyhow, since the question answering engine cannot find an answer in its database the query is passed to the interface. The interface then “consults” the module which handles the global resources like search engines, newsgroups or mailing lists (4.). It is important to mention here, that new resources can be easily added: An interface has to be written (as described below) and this interface/resource has to be registered to the KM system.

As those global resources may return a huge amount of answers (5.), it is the task of the interface-module to screen the best answers offered and to forward them to the question management engine. The engine sends the best answers back to the user (6.), who possibly makes an evaluation for one answer, which meets his expectations best (7.). In this positive case the question, the ranking and the link to the resource will be saved in the knowledge database. It is up to the interface-module if just the link to the resource or the whole resource will be saved for later use in the knowledge database. Usually the link should sufficient at the moment.

If the user responds to the system that none of the answers being offered are solving



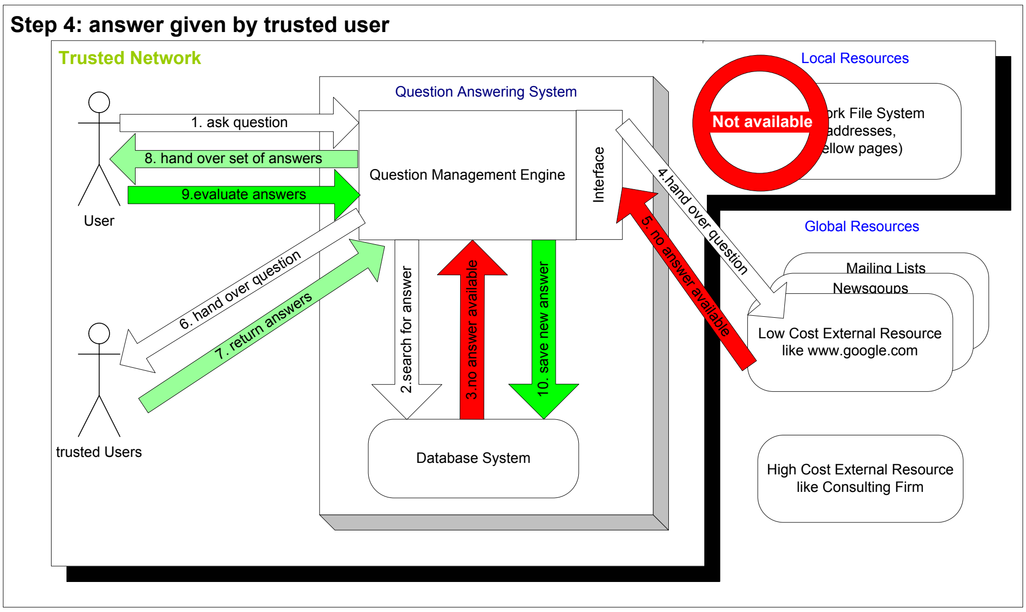


Figure 11.2.: Unified Information Access Steps 3 and 4

his problems the question management engine goes on with step four.

11.3.5. Step 4: Answer given by "Trusted" User(s)

Step four (see figure 11.2) in the question answering process might be of most interest of since for the first time another person or user will be involved in the answering process. First of all the user again formulates a query (1.), which is neither registered in the knowledge database (2. and 3.) nor be answerable by the use of a global resource (4. and 5.). Obviously (as described in step two) the answer can not found in the local resources either. So far all "cheap" resources have been exploited which forces the question management engine to transfer the question to human actors, e.g., a group of users being part of the project, the organisational unit or the company. Every authorised participant of the system has now the possibility to access these open questions.

As soon another user adds a comment to a open question, provides an answer or puts in any kind of feedback, the question answering engine will return that feedback to the user(s), who had asked the original query (8.). Right now the user can enter into a dialog with the adequate person(s) by asking more details about the needed information or she is just satisfied and evaluates the answer as positive. It is also possible that multiple staff members take part in a discussion about an open question (even a notification that other users are interested in this problem is an important fact, as will be noted later). As soon as more comments are being added to the open question every user taking part in the discussion will be notified that a new comment was added to the open question until the problem is solved and the question will be closed.

When a question is forwarded to other staff members the user asking the question has to attach points from his score account to it. Depending on the difficulty and his personal interest he can add high or low scores. The score system will be described in details later. If after a certain time the question will not be answered by any staff member and is still marked as open in the system, the question management engine proceeds with step five.

11.3.6. Step 5: Management Activities

If all the steps above did not lead to a solution of the open query, this particular problem becomes a management issue (see figure 11.3). Now the (project) manager or team leader has the responsibility to evaluate the severity of the problem. This process is aided by the points attached to a problem as well as by the number of users that marked this problem as "interesting". Different solution strategies can be imagined, starting from explicit order to internal staff to solve this very problem, up to the usage of external consultants. Those further steps have to be decided by management and

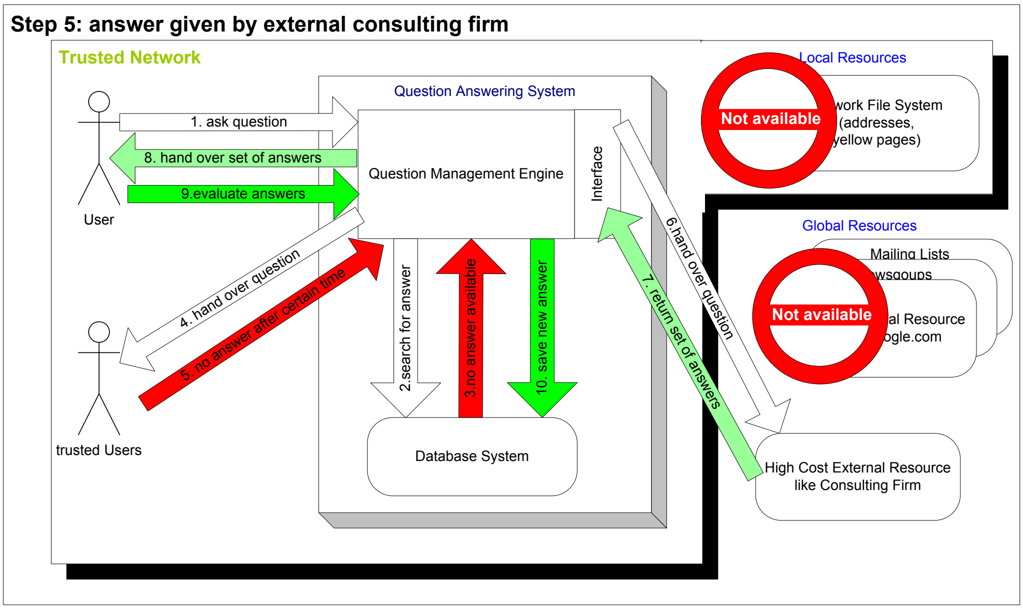
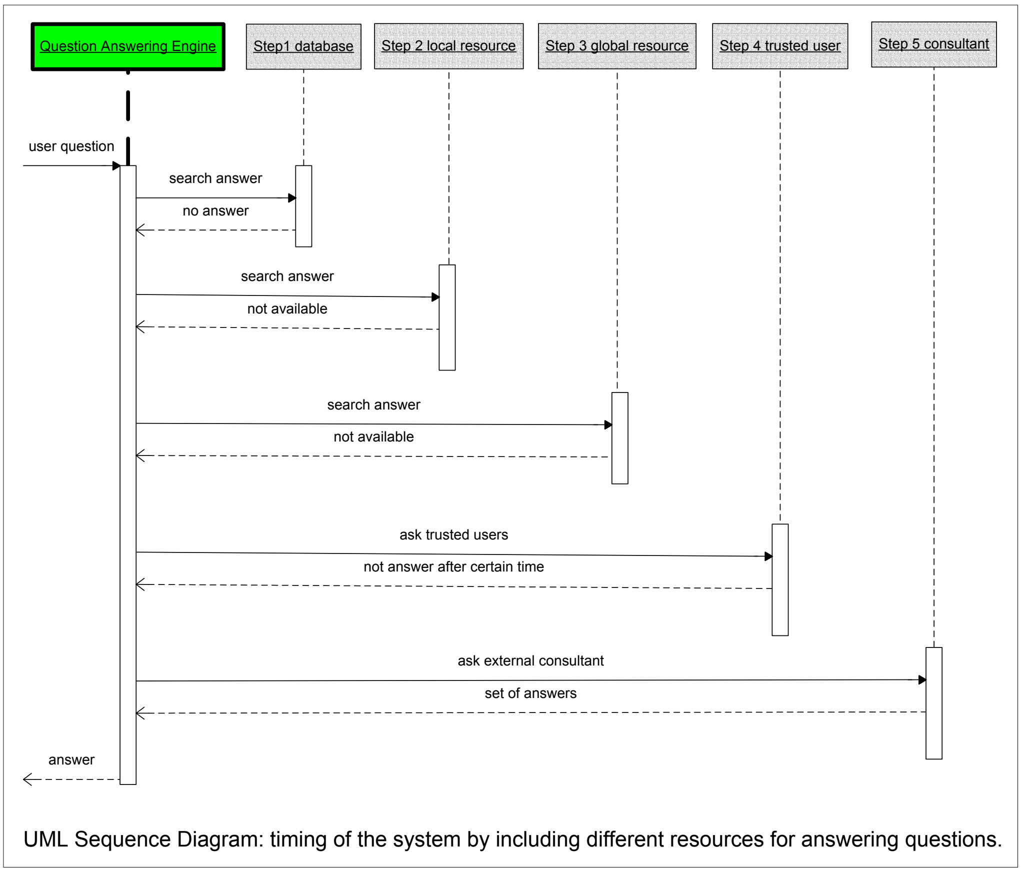


Figure 11.3.: Unified Information Access Step 5 (Management)

cannot be automated by obvious reasons.

The answer to the problem probably has to be entered manually afterwards, to save the question and answer set in the knowledge base. If the problem is too complex to be described textually it might be a help for the questioners to find another staff member who lastly solved the problem. The questioner can then contact this person to get some help from him or her. This leads to better networking and efficient communication. This process of the last step goes in detail:

The user starts by asking a question to the question management system (1.). After searching in the knowledge database for an adequate answer the engine passes the query to the trusted users (4.). These users try now to help the questioner (5.). If after a certain time period no fitting solution could be found the question management engine finally contacts the (project) manager to eventually consult a high cost external resource like a consulting company (6.) which is specialized on the problem. Ideally the “external help” now solved this particular problem (7.). The solution is handed over to the user (8.), who evaluates it (9.). After that the solution and the question are stored in the knowledge database (10.)



UML Sequence Diagram: timing of the system by including different resources for answering questions.

Figure 11.4.: Sequence Diagram of Unified Information Management System

11.4. Scoring System

The scoring system should be an additional motivation factor to keep the “trading” of questions and answers alive. Every new registered member of the question answering community could receive a certain amount of points periodically (which possibly can be regarded as “virtual money”). These points can be used to rank questions. The more important or difficult a question is the more points from the own account can be added. But not only own questions can be donated. If a user detects a problem posed by others, he or she may add points from the own account to demonstrate the importance of the problem.

If another user helps the questioner to find a solution to his problem, she should be obliged to give him the points. If more users took part in the answering process the questioner can also decide to split her points up to more users or even increase them if he appreciates their helpful work. As soon as her account of points approaches to zero, she should be motivated to answer questions from other users as well. The fact that questions have to be scored and that points are used are economically good causes for the users to deliver cogitated questions and reasonable contributions.

Using statistical reports, managers can trace out which staff members have special skills, take part in the system and where there is need for more training for individual workers. Moreover people can be detected and brought together who have the skills that are needed in a special project. Also the already mentioned "post-mortem" analysis of projects may be supported by the data collected here.

To give the score system a positional value the acquired points should be changeable in a kind of swap market. For instance one hundred points could be used for a bonus or a day off. The ideas of how to use points in another way than spending them for questions should not be given any limitations. In addition there should be a periodical list of the members with the highest score. To increase the amount of points managers or superiors can contribute additional points for employees as an incentive.

11.5. Technical Aspects

The interface module in the question answering system is just a software module upon which individual modules can be hooked-up to expand the functionality of the whole system. One of this modules could be the interface to the local resources. Another one could be the interface to the global resources, etc. The graphic shows the interface with its modules and the particular physical appearance of the information sources. The solid interface sets up the guidelines for the individual modules being implemented. It is important to find the right policy mix of routines which have to be implemented and those given by the solid interface. On the one hand the system should be relatively open and on the other hand the solid interface should represent a stable basis on which other freely selectable modules can be linked without a big effort. Some modules like local resources need to pre-scan their resources frequently while others like web search engines cannot do that (as it makes no sense to scan the whole web). Only after locating a web resource and saving the link in the knowledge database one can verify the resource for its availability. These two simple examples show clearly that the complexity, of the different requirements, needs to be carefully planed and will always need some compromises.

So at least two types on interface specifications will be defined: One for resources that need to be indexed (like file systems, XML documents), and one for resources, that can (or should) be queried directly (like WWW search engines, relational database systems). Essentially the interface specification describes methods that forward the query to the specific subsystem, and a method to return the result in a unified way.

If a new resource needs to be added to the system, a programmer has to implement (in the easiest case) those two methods for the specific information subsystem, add some meta-information about the system and register it to the KM main-application. With the next user queries, this new registered service will be included into the information pool.

11.6. Motivation and Cost Saving Factors

Participation of knowledge owners and future users is an important factor for the success of knowledge management systems. Many knowledge management systems failed, because of their lack of participation. Knowledge owners did not have the time or the intention to write down their skills. In many departments one person is the specialist of a particular domain. But that person is most probably the busiest and therefore the bottleneck of information flow. If she leaves the firm all her skills will accompany her. Especially these staff members often do not have time to do proper documentation of processes. While they keep on developing software for instance, the amount of undocumented software grows.

Finally one does not know where to start as the time for the whole documentation is not available. By choice people might not do any documentation at all, since there is no direct benefit of doing so. Exactly at this point a question answering system can be the solution. The direct benefit is given as someone helps solving a problem. It is a motivating process to see that one's skills are needed and it usually does not take more than five minutes to answer a question. The problem of documenting a huge amount of processes is split up into little pieces which makes it more convenient for knowledge owners to participate. And even more importantly, the process is problem driven, that means, in this approach solely the kind of knowledge is documented that is really needed! This is an important time saving factor as traditionally documents have been written that will never be used again. A software developer can now decide if it is necessary to describe a routine or an operation more precisely if other team members permanently press him for help concerning this routine. In that case he writes an tutorial on how to make use of an operation and sends the link as an answer to the users asking for help.

In many companies the problem exists, that staff members are dissatisfied with business procedures. They complain that procedures are old fashioned, long winded and untouchable in their execution. So it often happens that a question for improvement does not reach the responsible person and will be forgotten or lost a frustrating and little motivating experience. Moreover, it is often overseen, that many users suffer under the same problems! These effects could be leveraged by using a question answering system based on a knowledge base. Questions or suggestions for improvements can be watched by the managers and heads of departments in a "democratic" way. After figuring out a problem or innovation suggestion, suitable countermeasures have to be launched. The systems should be able to act like an early detection system and should discover trends and grievances in a company.

Particularly the motivation of the employees and the idea that their help, innovations, proposals and complaints are being heard is a main reason to justify the use of a question answering system like this. When thought further on, such a system leads to better quality management and faster innovations. By now problems are documented and especially managers can take their time to have a look at it. Very often an employee meets her superior on the floor and confronts him with a problem. The superior on the other hand is on his way to an important meeting and has absolutely no understanding for the employee?s needs, as he is too busy with his own matters at this moment. The need for a mediation architecture is obvious and the suggested system can be seen as such.

And finally, as mentioned already, the proposed system works like a information/knowledge proxy. And when designed and implemented properly, using the system will increase productivity and lower costs, as:

1. Employees should find solutions for problems faster

2. They will use available (expensive) information resources only when really needed.

3. Transparency in the problem/solving process is added, hence traceability is increased dramatically.

4. Communication between employees is encouraged.

Especially also the last point (not yet analysed in detail here) can increase the productivity dramatically as Abecker et. al. point out [1].

Part III.

Examples, Proof of Concepts

Preliminaries

> “3.3421 Eine besondere Bezeichnungsweise mag unwichtig sein, aber wichtig ist es immer, dass diese eine *mögliche* Bezeichnungsweise ist. [...] Das Einzelne erweist sich immer wieder als unwichtig, aber die Möglichkeit jedes Einzelnen gibt uns einen Aufschluss über das Wesen der Welt.”
> *Ludwig Wittgenstein [130]*

Part III of this thesis gives a brief overview over the practical work that partly lead to the conclusions of part II and was inspired by the theoretical findings described above. The projects and applications mentioned here where developed from summer 1999 to 2003. As the detailed results have been published and presented on international conferences, as well as in project summaries, at this place only a rather brief inspect into the ideas and concepts will be given. More concrete details about the projects can be found in the articles in the Appendix, and details about the theories behind the projects have been elucidated in the previous sections. Of course we tried to follow the suggestions made in the previous part when developing the projects. However, it is clear, that parts of the findings in Part 1 and 2 are a result of “lessons learned” by doing those projects.

For a detailed information, a list of conference papers can be found in the Appendix (page 203 *ff*). Additionally there are websites available of all projects online, as well as offline on the CD-ROM added to this thesis. As there is a lot of information to be found there, it was impossible to add it in printed form, and it seemed inappropriate to rewrite it here.

12. German Literature and Language Science Projects

12.1. Introduction

From summer 1999 to 2002 a cooperation between the *Institute of Software Technology and Interactive Systems (Vienna University of Technology)* and German literature and language scientists from the *University of Salzburg* was established to create two open distance learning websites. The content of the projects was: "Literatur in der Wiener Moderne"[1] and "Exilliteratur"[2]. The two projects differed mainly in the number of scientists involved. Whereas in the first project about 4 german literature and language scientists were working on the content, in the second project more then 10 were part of the team.

12.2. Preliminaries

Thus the initial idea was to support the German literature and language scientists (from a technical standpoint) by providing an infrastructure with the following characteristics:

- Generate a *Content Management System* (CMS), that allows to manage the following types of content
    - Lexical Information.
    - Biographical data from authors.
    - Articles organized as a set of lectures.
    - Multimedia content like: images, scanned (OCR) Text in PDF format, audio and maybe video snippets, Internet resources — URLs
    - Interactive Material (to give the possibility to interact with the material provided).
    - Special (manually) generated sections (like an interactive "walk" through Vienna)

[1] Literature round 1900

[2] Literature in Exile

- Content should be published web-based, but as these topics are (by nature) very text-based, give the user the possibility to print lectures.
- Exchange of graphical design should be possible easily. (In fact both projects do have a different graphical design).
- Content generation is a lot of work and the results are very valuable and should be preserved for *future re-use.*
- Content is managed by non-technical colleagues, namely by the German literature and language scientists themselves. A *user friendly interface* is required.

12.3. Content Management System Development

From the beginning there was the idea to keep the project content in XML and use XML tools for publishing purpose. Unfortunately, when the first project started in 1999, XML technologies, particularly those for publishing (XSLT, FO) and editing were not found to be mature enough, that the risk would be too high using XML directly for content management.

Additionally there was the problem, that the content has to be managed by non-technical collaborators. Hence it was not reasonable to let them work with XML directly, especially as stable XML editors were not available then. As a consequence, we decided to develop a CMS based on a relational database with an easy to use form-based GUI to access the content.

In the first project, a desktop database system was used, where both the database engine and the user interface components were both on the client. This concept was useful, as very quickly prototypes needed to be written and the number of editors was low.

In the second project, this *complete system was rewritten* to a client server based mechanism. This had several advantages: First of all a clean separation between user interface and data was guaranteed. This allowed better backup strategies for the database (on the server) and a better publishing mechanism. Additionally the system was (in contrary to the first) multi-user capable.

12.4. Publishing

From the beginning, multi-channel publishing was planned. Targets platforms were essentially HTML for web-publishing and PDF for print publishing. But additionally I tested to generate e-book versions to transfer (parts) of the content to electronic books. This was tested practically with the Rocket E-Book and basically worked fine.

However, as E-Books never really were a success on the market we did not include the E-Book support into the production version.

Web-publishing was done using XML and templating mechanism to generate HTML using design templates from the graphical designer. PDF publishing was initially performed using an intermediate format that could be described like a XML-based LaTeX, then a processor that generated LaTeX out of this format and the pdflatex processor to generate PDF. This rather complex strategy was selected, because in 1999 there were no real functional versions for, e.g., *formatting objects processing* available. And still today, the open source FOP processors lack a lot of functionality. Nevertheless the advantage was, that the results from the LaTeX publishing was very high in typesetting quality.

In the second project the system was changed to an XML based formatting objects processing. FO processors, as mentioned, still were and are not very mature, but quality and functionality was sufficient for our purpose.

12.5. Project Communication and Information Exchange

In the first project no particular communication support like a discussion-forum or chat was required, as nearly all problems could be solved in a 1:1 communication effort, and some more difficult topics were discussed in personal meetings either in Salzburg or in Vienna. The second project was more difficult in that respect. Hence we tried to setup a NNTP based discussion infrastructure. Unfortunately this measures was not as successful as expected. This was one of the "lessons learned" mentioned above, and details about installation, setup and introduction of such systems have been written in Part 2 of the thesis.

Additionally project todo lists and documentation was provided at an internal manually created intranet. This proved to be useful for all sides, particularly in the first project, but it also turned out, that it is a lot of work to keep such an intranet site up-to-date manually. The "lessons learned" from these activities were also analyzed in the previous chapters and moreover they led to the *Open Science Workplace* project described below..

12.6. Longevity of Digital Information

It was clear fact from the beginning, that the *value of the content* is far higher then the value of the (fast changing) IT infrastructure. So the idea was, to export all project content to XML. These XML documents then could be archived as project content (including the binary multimedia elements). Of course, these two parts: XML and multimedia binaries do not result the complete open distance learning project, but the essential content is saved that way for future re-use..

13. Open Science Workplace Project

> "Hitching our research to someone else's driving problems, and solving those problems on the owner's terms, leads us to richer computer science research." *Frederick P. Brooks, Jr. [19]*

13.1. Introduction

The *Open Science Workplace* (OSWP) project was started as a joint effort of the *Institute of Software Technology and Interactive Systems (Vienna University of Technology)* and the *University of Kerman (Iran).* The basic ideas that lead to this project were lessons learned from the previous projects mentioned above. It seemed clear, that practically every project, especially when project partners are dislocated, needs support in various areas like communication, resource exchange, project monitoring and others. Additionally it seemed reasonable to integrate those ICT tools into one portal allowing the project member to access all necessary information from one central place.

This chapter will give a brief description of the OSWP ideas and implementations, but will not go into too great details. The reason is, that most theoretical aspects are discussed in all details in Part 1 and Part 2 of this thesis and should not be repeated here. Additionally there were several articles about implementation and concept of OSWP published at scientific conferences and are attached to this thesis in the Appendix. And finally there is the project website http://www.oswp.info were additional details on implementation information as well as project source code can be found.

13.2. Various Prototypes and Concept-Tests

> "Plan to throw one away; you will, anyhow." *Fred Brooks [18]*

Various *prototypes* have been done in the effort to generate an open source platform for scientific cooperation. Partly concepts were tested by manual generation of intranet sites (see projects above), partly infrastructure was rapid prototyped for use in very specific situations, like a Cocoon application to support a project preparation phase.

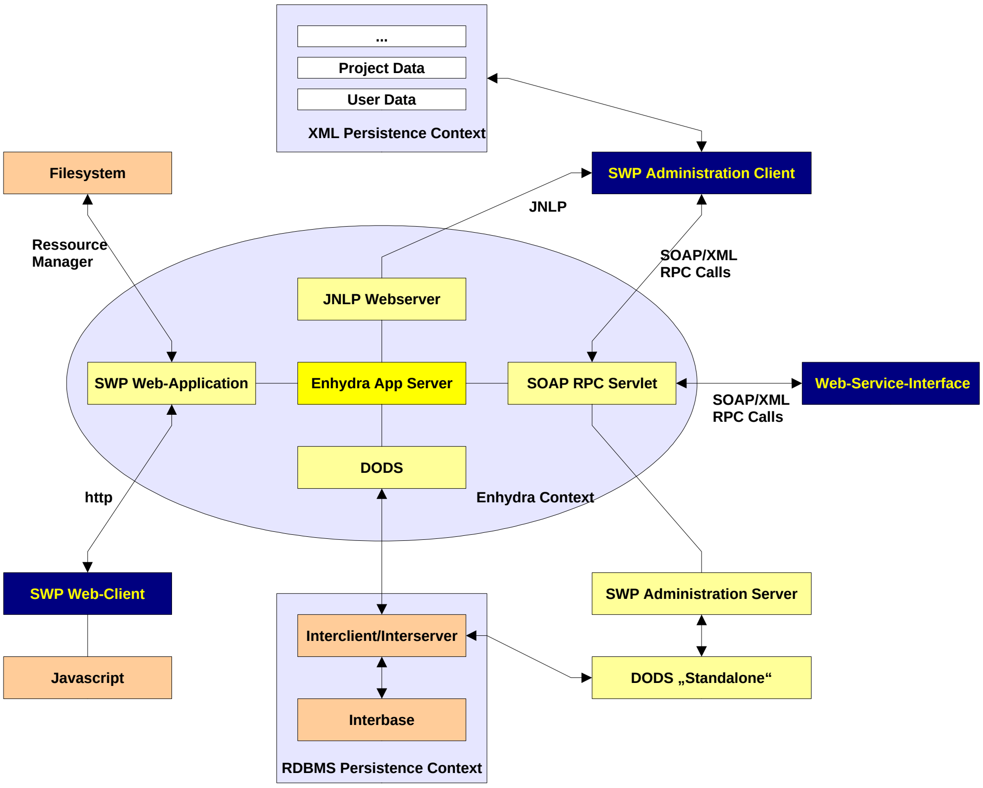


Figure 13.1.: The first SWP project was based on Enhydra application server using Webservices with a stand-alone Java client (delivered by Java Webstart) for administration of the system.

Also the first part of the OSWP project can be seen as an initial phase, as a *complete re-design* of the application programmed was decided.

This happened by different reasons: Essentially the first OSWP application is functional, but when starting with the first project, there were not too many open-source Java application servers available. And as the complete project should be open source itself (see also chapter 6) it was required to use only basic infrastructure like database system and application server that are open source themselves.

The decision turned out to be not optimal, and in beginning of the second project the Iranian partners suggested a complete re-design to generate a Java 2 Enterprise Edition compatible system. (Meanwhile there are also open source J2EE application server available). This gave the additional chance to re-design the project from bottom up, and re-implement it with the knowledge of the first development phase. Figure 13.1 illustrates the “old” design of the first SWP project, whereas figure 13.2 gives an

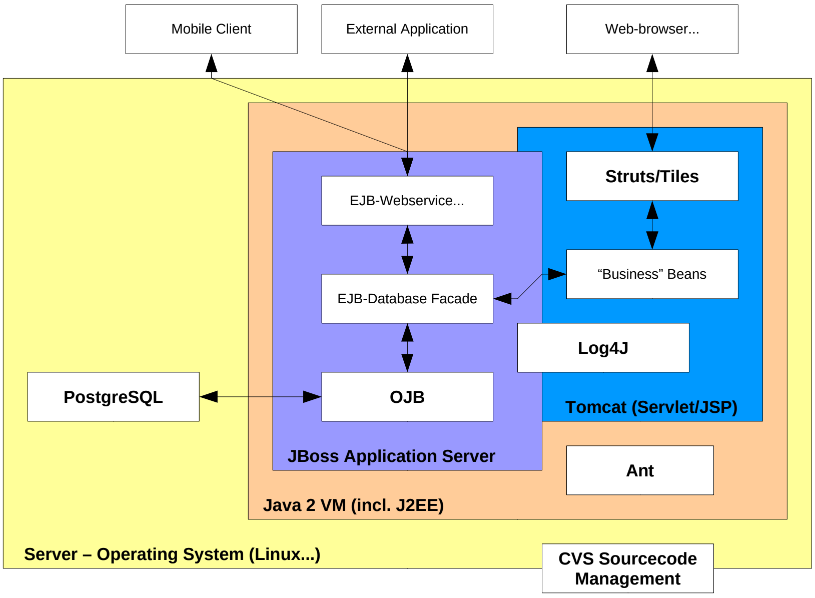


Figure 13.2.: The OSWP project is a complete re-implementation and re-design. This illustration shows the preliminary design of OSWP, building on top of a J2EE server, using CVS and ANT for code management, the OJB object relational bridge and the Struts/Tiles framework for building the JSP frontend. Webservices should integrate other systems like mobile clients.

overview over the preliminary design of the "new" OSWP project.

Unfortunately the project cooperation with the Iranian partners became difficult in the second project, so that I decided to do the second important step, to implement the new version of OSPW, at least the essential core part completely in Austria.

13.3. User Management Support

User management is a first and obvious necessity in every project. It starts with the fact, that *authentication* is required, hence a user administration has to be available. But moreover every user in a project should keep his or her personal information (like contact data) up-to-date, so that it is easily possible for every project partner to contact all others when needed. Even this rather simple functionality is already very helpful in many (dislocated) projects, particularly when projects patterns change often, e.g., when students are involved for programming specific parts of the project.

User-management means also supporting project managers in selecting users for specific tasks, as well as helping each individual in getting an overview over the capabilities of other colleagues. Hence a basic skill management is included in the user management support.

Moreover, as authentication was mentioned, *access control* is always a difficult task. In OSWP we decided to two paradigms:

1. No access control lists (like in Notes/Domino), as this creates an additional layer of complexity for the user and the administrator, but implicit access control as explained below.

2. Friendly policy.

This means, that we assume, that people that work together in a project should cooperate, not hide information. Hence a user has *write-access* to all data in tasks assigned to her, and may *read* all information from tasks that belong to projects she is working at. Only the project manager and the system administrator has write-access to the complete project.

I believe, that a more complex access control logic makes no sense in project cooperation; as if people urgently want to hide information, there exists a problem in the project already *outside* the ICT infrastructure, that should be solved first of all.

13.4. Project Management Support

Details about project management and monitoring have been described in chapter 3 and chapter 10. The ideas introduced there were mainly implemented in the OSWP system:

In each OSWP server instance an arbitrary number of projects with parameters like start date, end date, project manager, and the like can be defined. A *new project* is added by the administrator of the OSWP system, and a project manager is assigned.

From now on, the project manager can start to *structure the project* by defining project tasks, subtasks and by assigning colleagues to tasks. As soon as the project is running, the project manager is supported by a *visualization* system to follow the project progress of the project/task tree. This is also an important factor for other "observers" like a "top manager" who has access to all projects in the OSWP server and gets a *concise graphical overview* over the status of all projects. In case, she can go into detail and browse the parameters of a specific project or task.

13.5. Implicit Organization by Project Tasks

This idea of organizing each projects in a tree-like task structure is the *back-bone of the OSWP organization.* Not only are task core informations (like start-, end-dates,

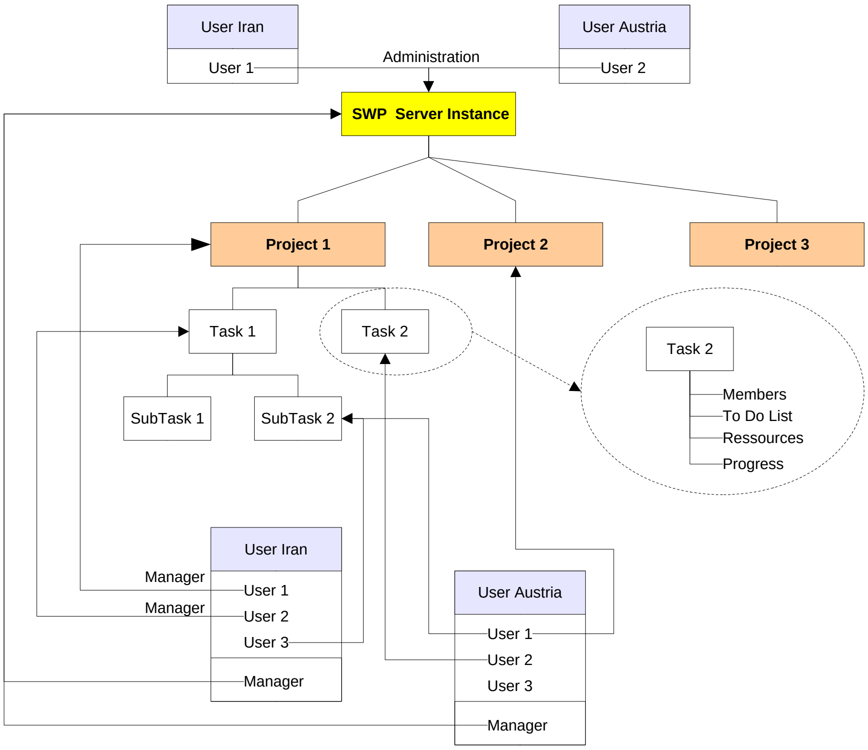


Figure 13.3.: This figure illustrates an example (O)SWP instance with projects, users and tasks. Projects and tasks are organized hierarchically, for each project and task there is precisely one manager defined. Administration can be done by multiple users. Each task has members, stores a todo list, resources, progress and other information.

responsible persons, progress tatus, ...) organized that way, also all other modules in the OSWP contact use this organizational mode. That means, that, e.g., resources are attached to tasks, discussion forums are organized following the project structure. An example in figure 13.3 should illustrate this concept.

13.6. Communication Support

Communication support is an extremely important issue as analyzed in chapter 3 and 10, as well as in sections 1.2.3 and 6.4.2.

OSWP does not implement an own mailing mechanism, which seemed unreasonable with everyone using email; but it supports mailing by specific features like group mailing to members of specific projects. Additionally there will be a news/discussion system included, that is again structured on the base of the project/task tree(s). This means, that every project has automatically its discussion forum, and every user sees immediately the discussions in all projects he is involved. Furthermore also synchronous communication should be supported including a chat system. Here again: chat rooms are organized following the project structure.

And finally there is a notification mechanism available, that automatically notifies users when interesting events occur, like task progress changes, new resources available or new articles posted.

13.7. Resource Management

Resource management is one of the core problems in any (dislocated) project. OSWP allows to manage *different types of resources* (like files, databases, XML) server based, and organize them by assigning resources to tasks. So no "new" structure is required, and resources are located where they belong to. (Access control to resources is described in section 13.3.)

It is unnecessary to mention, that the analysis made in chapters 7, 8 and 10 are a result of dealing with the problems mentioned here, and the suggestions made there are taken into consideration when implementing OSWP.

13.8. Multi-Channel "Publication"

The term "publishing" is not really appropriate here; essentially we try to implement the ideas described in chapter 8, 10, 11 in OSWP.

This means, that a clear separation of data, logic and design is provided. This should guarantee, that project data is accessible from the web-portal, as well as from other applications like mobile clients using web-services interfaces. Additionally XML export and import will guarantee that project data can be *archived in a platform-neutral way*, may be exchanged between systems and used for publishing: That means, that for example stylesheets may auto-create project reports.

13.9. Other Modules

Various other modules are planned. E.g., the ideas expressed in the knowledge management chapter (see chapter 9 on page 131) will be implemented for testing. Group scheduling, calender functionality and web-based email support are planned for the final release.

We could summarize, the basic idea as, to give *every project collaborator* the possibility to enter *one* portal page, and get a clear overview about *all relevant project* *information*, starting with task status, discussion news, resources, e-mails, important dates and so on.

Part IV.

Conclusions

Part 1

In part 1 the main problems of project related (scientific) cooperation were introduced including "definition" of all important terms. A first (problem driven) view toward information and project cooperation is given, as well as a first introduction into the experiences and practical works that are the basis for this thesis. The basic concepts of *knowledge building* versus *information degradation* are introduced here.

Part 2

Part 2 analyzes the problems from a more theoretical point of view, but also gives clear suggestions for practical work. Hence the awareness of essential topics like (management of) information, data, knowledge, cooperation, policies and others should be the consequence.

In brief, Part 2 of the thesis should help the reader in doing the following steps:

1. *System decision*: Chapter 6 discusses the effects of various system decisions (open source, open protocol, closed source, proprietary systems) with regard to effects on politics and society. The bottom line is, that many important arguments for selecting open systems (open source as well as open protocol) are highlighted.

2. In chapter 7 the importance of steps toward *longevity of digital information* were recollected. It is outlined, that besides all advantages in digital information processing, the risk of losing essential data and information is higher today then ever in history. This is particularly true for project-related work. This chapter analyzes the situation and gives some recommendations; but also the following chapters are part of a strategy to avoid information degradation.

3. Chapter 8 is the foundation for building high quality ICT systems particularly for supporting project related work. Attributes of *data, information and knowledge* are outlined, as well as structural aspects of data. Conditions for future oriented architectures are analyzed, on the basis of ontologies (like the semantic web) and open standards (like XML). These activities should leverage interoperability as well as re-use of data and information in various domains. Also the aspects of user interaction with data according to clean separation of UI issues (application based as well as web-based) and data management are accentuated.

4. On the foundation of the findings in the previous chapters, chapter 9 and chapter 11 suggest a *question-based knowledge management* strategy, that is coinstantaneous the footing for a *unified information access* concept toward various different information systems.

5. Chapter 10 finally delivers on the one hand a conclusion of the previous chapters with particular focus to *dislocated project work*, and on the other hand stresses the importance of *multi-channel publishing*, a well designed *communication structure* as well as strategies toward unification of access to information sources (information portals, KM systems)

Part 3 and Appendix

The last part of this thesis are an introduction and review of the projects performed by the author at the *Institute of Software Technology and Interactive Systems (Vienna University of Technology)* that were the basis for the findings in this thesis, namely the

- German literature and language projects: “Literatur in der Wiener Moderne” and
- “Exilliteratur”
- The concept and development of *Open Science Workplace*

In all projects the ideas of openness, interoperability and durability of data, information and knowledge were a main concerns, as expressed throughout this thesis.

Particularly the work on OSWP gave the insight into various aspects of information management, knowledge management and cooperation/communication issues.

Finally the Appendix contains a detailed list of the essential *scientific articles* presented on various peer-reviewed confereces and *references to important literature.*

Part V.

Appendix

A. Citations: Original Language

A.1. Helmut Willke: Dystopia

A.1.1. Introduction

The original book [129] is in written in german; as no "official" english translation is available, the citations are translated by the author.

The pages mentioned here refer to the first edition in *Suhrkamp Taschenbuch Wissenschaft* paperback.

A.1.2. Page 7

"Die gegenwärtige Moderne ist so furchtbar fortgeschritten, dass Erfolg und Krisen gleichzeitig auftreten. Ob jemand die Erfolge feiert oder die Krisen zelebriert, hängt vom kontingenten Standpunkt und der Wahl des Kanals ab.

[. . .]

Eine Krisis des Wissens ensteht deshalb im Kern dadurch, das sich niemand auf das vorhandene Wissen verlassen kann, solange das komplementäre Nichtwissen nicht in gleicher Weise zur Kenntnis genommen und handhabbar gemacht ist wie das Wissen selbst. Die Krisis des Wissens bezeichnet die Unfähigkeit, mit Nichtwissen kompetent umzugehen."

"The current modern spirit is so terrible progressive, that success as well as crisis emerge at the same time. If someone celebrates a success or a crisis, depends on the viewpoint and the selection of the channel.

[. . .]

A crisis of knowledge results basically because no one can count on the available knowledge. This is true until the complementary nescience is noted and made usable the same way as the knowledge itself. The crisis of knowledge means the inability to deal with nescience in a competent way."

A.1.3. Page 38

"Die Krisis des Wissens wird, wie gesagt, kognitiv getrieben von der neuen Relevanz des Nichtwissens, und sie wird operativ davon getrieben, da es nun darum geht, die richtigen Fehler schneller zu machen als die Wettbewerber, um Lernprozesse zu intensivieren, die im Kern darin bestehen, Expertise im Umgang mit Nichtwissen zu entwickeln.”

“The crisis of knowledge ist congnitively driven by the new relevance of nescience. Operationally it is driven by the necessity to make the right mistakes faster than the competitors to intensivate learning processes, what means developing expertise in handling nescience”

A.1.4. Page 56

“Mehr Wissen produziert, wie bereits im Wissenschaftssystem, so nun auch im Finanzsystem, nicht mehr Wahrheit und mehr Sicherheit, sondern paradoxerweise mehr Optionen, mehr Ungewissenheit, und damit mehr spezifisches Nichtwissen.”

“As already seen in science, also in the financial sector, more knowledge does not produce more truth and more security, but in a paradox way more options, more unsecurity, hence more specific nescience.”

A.2. Open Source

A.2.1. Pressemitteilung deutscher Bundestag

> ”‘Die Ankündigung der Firma Microsoft, ihr Betriebssystem Windows NT 4.0 ab dem Jahr 2003 nicht mehr zu unterstützen, zwingt den Deutschen Bundestag, für seine ca. 5000 Arbeitsplatz- und Serversysteme eine Migration zu einem Nachfolgesystem vorzunehmen.
>
> Mit der im Quellcode offenliegenden und lizenzgebührenfreien Open Source Software existieren heute Alternativen zu den Microsoftprodukten. Damit besteht auch die Möglichkeit, *die Abhängigkeit von einem Unternehmen zu reduzieren.*
>
> [. . .]
>
> Eine solche Entscheidung bedarf umfangreicher Untersuchungen.
>
> [. . .]
>
> [die] Kommission [ist] zu folgenden Empfehlungen gelangt:
>
> - Die Server werden auf das Betriebssystem Linux umgestellt, als Verzeichnisdienst wird OpenLdap eingesetzt.
>
> - Auf den Arbeitsplatzrechnern wird Microsoft XP und das entsprechende Office Paket eingesetzt.
>
> - Als Standard für Browser und eMail Client wird weiterhin Netscape genutzt.

[. . .]

In der Studie war die Variante, bei der nur einige Server auf Linux, der groe Rest der Server aber auf Windows umgestellt worden wäre, bei der Nutzwertanalyse auf dem ersten Platz gelandet. Entsprechend wäre als Verzeichnisdienst Active Directory eingesetzt worden.

Die Variante, der die Kommission jetzt zugestimmt hat, hatte bei der Nutzwertanalyse den zweiten Platz erreicht.

Mit dieser Entscheidung weicht die IuK- Kommission bewusst vom Ergebnis der Studie ab, *weil sie mit dieser Entscheidung die strategische überlegung verbindet, die zur Zeit bestehende Abhängigkeit von den Produkten eines Anbieters zu lockern. Sie erhofft sich durch diesen Schritt einen gröeren Freiraum bei zukünftigen Entscheidungen.*"' *Pressemitteilung Deutscher Bundestag (Hervorhebung vom Autor) [21]*

A.2.2. Pressemitteilung Schwäbisch Hall

"'Die Stadtverwaltung Schwäbisch Hall hat sich für den Aufbau einer vollständig linux-basierten IT-Infrastruktur entschieden. Durch den Einsatz von SuSE Linux auf Servern und Desktop-PCs wird eine vorhandene Windows-Infrastruktur abgelöst. Das Einsparpotential durch diese Lösung liegt im sechsstelligen Euro-Bereich und trägt mageblich zur Entlastung des Haushalts der 36.000 Einwohner zählenden Kommune bei.

[. . .]

[Der] Oberbürgermeister [. . .] erläutert die Entscheidung der Stadtverwaltung: "Für mich gibt es drei Gründe, um auf Linux zu setzen. Erstens erwarte ich eine deutliche Kostenreduktion unserer Ausgaben im IT-Bereich durch die Senkung der Software-Lizenz-Gebühren. [. . .] Zweitens soll unsere IT-Struktur sicherer werden; die Fachleute stellen Linux in dieser Hinsicht hervorragende Noten aus. *Drittens setzen wir auf offene Standards, die eine Wahlfreiheit bei den technischen Angeboten sicher stellen.*"

[. . .]

Eingebunden in das Projekt ist auch ein umfangreiches Servicekonzept, das die Eröffnung eines Linux-Kompetenz-Centers in der Stadtverwaltung vorsieht. Dieses dient in der ersten Ausbaustufe zur Schulung und Information der Mitarbeiter, in weiteren Schritten auch zur Nutzung durch örtliche Schulen, Krankenhäuser und die öffentlichkeit, um die Potentiale von Linux-Lösungen möglichst weiten Teilen zugänglich zu machen. Darüber hinaus sind auch der Aufbau einer Hotline für die Mitarbeiter, zielgruppengenaue Schulungen zu Linux und Beratungsangebote für die Auswahl und Umsetzung von weiteren linux-basierten Anwendungen vorgesehen."' *Pressemitteilung Schwäbisch Hall (Hervorhebung vom Autor) [104]*

A.3. Knowledge Management

"Durch die Welt 3 werden unsere Träume dauernd korrigiert, bis sie dann schliesslich konkretisiert werden können." *Karl Raimund Popper [82]*

B. Publications

This part of the Appendix contains a list of peer-reviews english (conference) publications, as well as german publications and finally important web-references.

B.1. English Publications

- Conference Paper: Euromicro 2003 (Antalya): “Closing the Gap-From Nescience to Knowledge Management”
- Conference Paper: iiWAS (International Conference on Information Integration and Web Applications and Services) 2002 (presented by Prof. Tjoa): “The Application of Software Agent Technology to Project Management Infrastructure”
- Conference Paper: ICWL (1st International Conference on Web-based Learning), August 2002, Hongkong: “Refactoring the Application Infrastructure for Building Open Distance Learning Websites — Lessons Learned in a Case Study in a German Literature and Language e-Learning Application” in: Web-based Learning: Men and Machines, 2002, World Scientific Publishing
- Invited Speak: iiWAS (International Conference on Information Integration and Web Applications and Services) 2001 Linz,“Building an Web-Based Open Source Tool to Enhance Project Management, Monitoring and Collaboration in Scientific Projects”
- Conference Short Paper: ED-Media 2000, Montreal: “Austrian Literature Moving to Cyberspace ? A Framework for Building an Open Distance Learning Website using Platform Independent Standards Like XML”[1]
- Conference Paper: ED-ICT Conference 2001 (Vienna): “Persistent Knowledge Acquisition for Educational Purpose: An Open Distance Learning Website for German Literature and Language Scientists”
- Conference Paper: Yogya 2000 (presented by Prof. Tjoa): “Developing a Framework for Building Open Distance Learning Websites in the Literature and Culture Domain”

[1] In the following appendix the long paper will be reprinted; simply because the german literature and language project is presented there in more details, that could not be found in the short paper.

B.2. German Publications

- iX (Verlag Heinz Heise) 5/2003: “XML nativ speichern: XML-DBMS Xindice von der Apache Software Foundation”
- iX (Verlag Heinz Heise) 11/2002: “Immer gröšer (S. 18)” —Bericht über die ”Very Large Databases”Konferenz in Hongkong 2002
- iX (Verlag Heinz Heise) 6/2002: “Web-Projekte - Apache.org: mehr als ein Webserver (S. 98)”

B.3. Web-Resources

The results of the mentioned projects can be seen online, and on the added CD-ROM respectively:

- German Literature and Language Projects: http://www.literaturepochen.at
- Open Science Workplace: http://www.oswp.info

C. Policies of this Document and the Author's Work

In this thesis I suggest the usage of open tools and protocols as far as possible. As a logical consequence, all projects I have done during my time at the *Institute of Software Technology and Interactive Systems (Vienna University of Technology)* were dedicated to the open source/protocol idea. I used nearly exclusively open source products, where available.

Also this thesis is written using the LaTeX typesetting system [52], Open Office tools, the GNU/Linux operating system with various GNU tools and the CVS (concurrent versioning system) for versioning and backup.

Bibliography

[1] A. Abecker, A. Bernardi, and M. Sintek. Developing a knowledge management technology: An encompassing view on know more. In *Proceedings of the 8th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises*, pages 216–222, 1999.

[2] Andreas Abecker, Ansgar Bernardi, Knut Hinkelmann, Otto Kuhn, and Michael Sintek. Toward a technology for organizational memories. *IEEE Intelligent Systems*, 13(3):40–48, 1998.

[3] Serge Abiteboul. Querying semi-structured data. In *ICDT*, pages 1–18, 1997.

[4] Serge Abiteboul. On views and xml. In *Proceedings of the eighteenth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pages 1–9. ACM Press, 1999.

[5] G. Aichholzer and R. Schmutzer. Organizational challenges to the development of electronic government. In A Min Tjoa, Roland Wagner, and Ala Al-Zobaidie, editors, *Proceedings of the 11th International Workshop on Database and Expert Systems Applications*, pages 379–383. ACM Press, 2000.

[6] Maryam Alavi and Dorothy E. Leidner. Knowledge management systems: issues, challenges, and benefits. *Communications of the AIS*, 1(2es):1, 1999.

[7] David P. Anderson and John Kubiatowicz. The worldwide computer. *Scientific American*, March 2002.

[8] Kenneth M. Anderson. Supporting industrial hyperwebs: lessons in scalability. In *Proceedings of the 21st International Conference on Software Engineering*, pages 573–582. IEEE Computer Society Press, 1999.

[9] Amin Andjomshoa, Alexander Schatten, A Min Tjoa, and Hassan Shafazand. The application of software agent technology to project management infrastructure. In *Proceedings of the International Conference on Information Integration and Web-based Applications and Services*, 2002.

[10] Apache project. http://www.apache.org (last accessed July 2003).

[11] Bruce R. Barkstrom, Melinda Finch, Michelle Ferebee, and Calvin Mackey. Adapting digital libraries to continual evolution. In *Proceeding of the second ACM/IEEE-CS Joint Conference on Digital Libraries*, pages 242–243. ACM Press, 2002.

[12] Kent Beck. *Extreme Programming Explained. Embrace Change.* Addison-Wesley Professional, October 1999.

[13] Tim Berners-Lee, James Hendler, and Ora Lassila. The semantic web. *Scientific American*, may 2001.

[14] Abraham Bernstein. How can cooperative work tools support dynamic group process? bridging the specificity frontier. In *Proceeding of the ACM 2000 Conference on Computer supported cooperative work*, pages 279–288. ACM Press, 2000.

[15] Suresh K. Bhavnani, Frederick Reif, and Bonnie E. John. Beyond command knowledge: identifying and teaching strategic knowledge for using complex computer applications. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 229–236. ACM Press, 2001.

[16] Grady Booch, James Rumbaugh, and Ivar Jacobson. *Unified Modelling Language User Guide.* Addison-Wesley Professional, Januar 1999.

[17] Martin Braschler and Bärbel Ripplinger. Stemming and decompounding for german text retrieval. In *Proceedings of the European Colloquium on Information Retrieval and Research*, pages 177–192. Springer, 2003.

[18] Frederick P. Brooks. *The Mythical Man Month. Essays on Software Engineering.* Addison Wesley, August 1995.

[19] Frederick P. Brooks. The computer scientist as toolsmith. *Communications of the ACM*, 39(3):61–68, March 1996.

[20] Basic support for collaborative work groupware system. http://www.bscw.com (last accessed December 2003).

[21] Bundestag.de pressemitteilung: Empfehlung für künftige it ausstattung. http://www.bundestag.de/aktuell/presse/2002/pz_0202285.html (as of June 2003), feb 2002.

[22] Peter Buneman. Semistructured data. In *Proceedings of the sixteenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems*, pages 117–121. ACM Press, 1997.

[23] Sheryl Burgstahler. Distance learning: Universal design, universal access. *Educational Technoloy Review; International Forum on Educational Technology Issues and Applications*, 10(1), 2002. AACE Online Publication.

[24] Donald D. Chamberlin. Relational data-base management systems. *ACM Computing Surveys (CSUR)*, 8(1):43–66, 1976.

[25] Cocoon XML publishing framework. http://cocoon.apache.org (last accessed July 2003).

[26] C. J. Date and Hugh Darwen. *A Guide to the SQL Standard. A user's guide to the standard database language SQL.* Addison Wesley Longman, fourth edition, 1997.

[27] Chad Davis and Coskun Bayrak. Open source development and the world wide web: a certain tension. *ACM SIGSOFT Software Engineering Notes*, 27(5):93–97, 2002.

[28] Alin Deutsch, Mary Fernandez, and Dan Suciu. Storing semistructured data with stored. In *Proceedings of the 1999 ACM SIGMOD International Conference on Management of Data*, pages 431–442. ACM Press, 1999.

[29] Keith Devlin. A mathematical framework for the study of information. *Complexity*, 4(2):30–32, 1998.

[30] Rose Dieng, Olivier Corby, Alain Giboin, and Myriam Ribiere. Methods and tools for corporate knowledge management. Technical Report RR-3485, INRIA Sophia-Antipolis, 1998.

[31] Klaus R. Dittrich, Dimitrios Tombros, and Andreas Geppert. Databases in software engineering: a roadmap. In *Proceedings of the Conference on the Future of Software Engineering*, pages 293–302. ACM Press, 2000.

[32] The docbook specification. http://www.docbook.org (last accessed May 2003), 2003.

[33] Christoph Drösser. Ewig lockt die Tantieme. *Die Zeit*, 10:33, Feb 2003.

[34] Dublin core metadata initiative. http://dublincore.org (as of January 2003), 2003.

[35] Manfred Eigen and Ruthild Winkler. *Das Spiel, Naturgesetze steuern den Zufall.* Piper, München, vierte edition, 1996.

[36] Douglas C. Engelbart. *XML Topic Maps: Creating and Using Topic Maps for the Web.* Addison-Wesley, first edition, July 2002.

[37] Shelly Farnham, Harry R. Chesley, Debbie E. McGhee, Reena Kawal, and Jennifer Landau. Structured online interactions: improving the decision-making of small discussion groups. In *Proceeding of the ACM 2000 Conference on Computer Supported Cooperative Work*, pages 299–308. ACM Press, 2000.

[38] Alan Foley and Bob Regan. Web design for accessibility: Policies and practice. *Educational Technoloy Review; International Forum on Educational Technology Issues and Applications*, 10(1), 2002. AACE Online Publication.

[39] Manfred Füllsack. *Leben ohne zu arbeiten? Zur Sozialtheorie des Grundeinkommens.* Avinus Verlag, Berlin, 2002.

[40] Les Gasser. The integration of computing and routine work. *ACM Transactions on Information Systems (TOIS)*, 4(3):205–225, 1986.

[41] Volker Grassmuck. *Freie Software zwischen Privat- und Gemeineigentum.* Bundeszentrale für politische Bildung, Bonn, 2001.

[42] Jonathan Grudin. Why cscw applications fail: problems in the design and evaluation of organization of organizational interfaces. In *Proceedings of the Conference on Computer-Supported Cooperative Work*, pages 85–93. ACM Press, 1988.

[43] Nicola Guarino and Christopher Welty. Corporate knowledge management. In *Proceedings of ECAI-2000, The European Conference on Artificial Intelligence*, Amsterdam, 2000. IOS Press.

[44] Wolfgang Hagen. Bill Luhan und Marshall McGates, Die Extension des Menschen als Extension der USA. In Alexander Roesler and Bernd Stiegler, editors, *Microsoft: Medien, Macht, Monopol*, pages 24–47. Edition Suhrkamp, Frankfurt am Main, erste edition, 2002.

[45] Diane Hillman. Using dublin core. http://dublincore.org (as of January 2003), april 2001.

[46] Ursula Holtgrewe and Raymund Werle. (re-)de-commodifying software? open source software between business strategy and social movement. *Science Studies*, 14(2):43–65, 2001.

[47] The human genome project. http://www.ornl.gov/hgmis/ (last accessed January 2003).

[48] Renato Ianella. An idiot's guide to the resource description framework. *The New Review of Information Networking*, 4, 1998.

[49] Iso standards. http://www.iso.org (last accessed May 2003).

[50] Stuart Kauffman. *At Home in the Universe, The Search for the Laws of Self-Organisation and Complexity.* Oxford University Press, Oxford, New York, 1995.

[51] Ralph Kimball and Margy Ross. *The Data Warehouse Toolkit.* John Wiley, Chichester, second edition, April 2002.

[52] Helmut Kopka. *LaTeX Einführung.* Pearson Studium, dritte edition, 2002.

[53] Ken Krechmer. Cathedrals, libraries and bazaars. In *Proceedings of the 17th Symposium on Proceedings of the 2002 ACM Symposium on Applied Computing*, pages 1053–1057. ACM Press, 2002.

[54] Stefan Krempl. Microsoft als Wirtschaftsmacht. Eine Softwarefirma tritt an, die digitale Welt zu erobern. In Alexander Roesler and Bernd Stiegler, editors, *Microsoft: Medien, Macht, Monopol*, pages 73–102. Edition Suhrkamp, Frankfurt am Main, erste edition, 2002.

[55] Thomas S. Kuhn. *The Structure of Scientific Revolutions.* University of Chicago Press, third edition, December 1996.

[56] Vipin Kumar and Mohammed Zaki. High performance data mining (tutorial pm-3). In *Tutorial notes of the sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 309–425. ACM Press, 2000.

[57] Preserving digital information: Report of the task force on archiving of digital information. http://www.rlg.org/ArchTF/index.html, may, 1996.

[58] David M. Levy. Heroic measures: reflections on the possibility and purpose of digital preservation. In *Proceedings of the third ACM conference on Digital libraries*, pages 152–161. ACM Press, 1998.

[59] Apache software license. http://www.apache.org/LICENSE.txt (as of March 2003), 2000.

[60] Gnu general public license. http://www.gnu.org/copyleft/gpl.html (as of March 2003), June 1991.

[61] Raymond A. Lorie. Long term preservation of digital information. In *Proceedings of the first ACM/IEEE-CS Joint Conference on Digital Libraries*, pages 346–352. ACM Press, 2001.

[62] Lotus/IBM Website. http://www.ibm.com (last accessed July 2003).

[63] Bertram Ludäscher, Richard Marciano, and Reagan Moore. Preservation of digital data with self-validating, self-instantiating knowledge-based archives. *ACM SIGMOD Record*, 30(3):54–63, 2001.

[64] Miki Magyar. Automating software documentation: a case study. In *Proceedings of IEEE Professional Communication Society International Professional Communication Conference and ACM Special Interest Group on Documentation Conference on Technology and Teamwork*, pages 549–558. IEEE Educational Activities Department, 2000.

[65] Thomas W. Malone, Kenneth R. Grant, Kum-Yew Lai, Ramana Rao, and David Rosenblitt. Semistructured messages are surprisingly useful for computer-supported coordination. *ACM Transactions on Information Systems (TOIS)*, 5(2):115–131, 1987.

[66] David Mattox, Len Seligman, and Ken Smith. Rapper: a wrapper generator with linguistic knowledge. In *Proceedings of the second International Workshop on Web Information and Data Management*, pages 6–11. ACM Press, 1999.

[67] Microsoft website. http://www.microsoft.com (last accessed July 2003).

[68] Audris Mockus, Roy T Fielding, and James D Herbsleb. Two case studies of open source software development: Apache and mozilla. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 11(3):309–346, 2002.

[69] n/a. Reference model for an open archival information system (oais). Technical Report CCSDS 650.0-B-1, Consultative Committee for Space Data Systems (CCSDS), jan 2002.

[70] Michael Nentwich. De-commodification in academic knowledge distribution. *Science Studies*, 14(2):21–42, 2001.

[71] Ikujiro Nonaka, Hirotaka Takeuchi, and Hiro Takeuchi. *The Knowledge-Creating Company: How Japanese Companies Create the Dynamics of Innovation.* Oxford University Press, May 1995.

[72] H. S. Nwana and D. T. Ndumu. A perspective on software agents research. *The Knowledge Engineering Review*, 14(2):1–18, 1999.

[73] Object management group. http://www.omg.org (last accessed January 2003).

[74] Open cyc. http://www.opencyc.org (last accessed June 2003).

[75] Opensource.org information site. http://www.opensource.org (last accessed January 2003).

[76] Tim O'Reilly. Lessons from open-source software development. *Communications of the ACM*, 42(4):32–37, 1999.

[77] Open science workplace. http://www.oswp.info (as of July 2003), 2003.

[78] John Ousterhout. Free software needs profit. *Communications of the ACM*, 42(4):44–45, 1999.

[79] Jörg Pflüger and Peter Purgathofer. FAQ: Microsoft. In Alexander Roesler and Bernd Stiegler, editors, *Microsoft: Medien, Macht, Monopol*, pages 73–102. Edition Suhrkamp, Frankfurt am Main, erste edition, 2002.

[80] Karl Raimund Popper. *Die offene Gesellschaft und ihre Feinde 1 (Der Zauber Platons).* J.C.B. Mohr (Paul Siebeck), Tübingen, siebente edition, 1992.

[81] Karl Raimund Popper. *Die offene Gesellschaft und ihre Feinde 2 (Falsche Propheten: Hegel, Marx und die Folgen).* J.C.B. Mohr (Paul Siebeck), Tübingen, siebente edition, 1992.

[82] Karl Raimund Popper. *Alles Leben ist Problemlösen. ber Erkenntnis, Geschichte und Politik.* Piper, München, elfte edition, 2002.

[83] Karl Raimund Popper and John C. Eccles. *The Self and its Brain.* Springer, Heidelberg, reprint edition, 1977.

[84] Karl Raimund Popper and John C. Eccles. *Das Ich und sein Gehirn.* Piper, München, elfte edition, 1994.

[85] Rational software website. http://www.rational.com (last accessed January 2003).

[86] Eric S. Raymond. *The Cathedral and the Bazaar.* O'Reilly UK, Feb 2001.

[87] Rdf: The ressource description framework. http://www.w3.org/RDF/ (as of May 2003), 2003.

[88] Rupert Riedl. *Evolution und Erkenntnis.* Piper, München, vierte edition, Mai 1990.

[89] William N. Robinson and Vecheslav Volkov. Supporting the negotiation life cycle. *Communications of the ACM*, 41(5):95–102, 1998.

[90] Jeff Rothenberg. Ensuring the longevity of digital documents. *Scientific American*, 272(1):42–47, 1995.

[91] Ioana Rus and Mikael Lindvall. Knowledge management in software engineering. *IEEE Software*, pages 26–38, May/June 2002.

[92] Gilbert Ryle. *The Concept of Mind.* University of Chicago Press, Chicago, reprint edition, 1984.

[93] Jerome H. Saltzer and Michael D. Schroeder. The protection of information in computer systems. In *Proc. IEEE 63*. IEEE Computer Society Press, september 1975.

[94] Alexander Schatten. Web-Projekte - Apache.org: mehr als ein Webserver. *iX*, pages 98–101, Juni 2002.

[95] Alexander Schatten. XML nativ speichern: XML-DBMS Xindice von der Apache Software Foundation. *iX*, pages 69–71, Mai 2003.

[96] Alexander Schatten, Stefan Biffl, and A Min Tjoa. Closing the gap, from nescience to knowledge management. In *Proceedings of the Euromicro Conference*. IEEE, 2003.

[97] Alexander Schatten, Marian Schedenig, and A Min Tjoa. Refactoring the application infrastructure for building open distance –learning websites – lessons learned in a case study in a german literature and language e-learning application. In *Proceedings of the International Conference on Web-based Learning Conference*, 2002.

[98] Alexander Schatten and A Min Tjoa. Developing a framework for building open distance learning websites in the literature and culture domain. In *Yogya 2000 Conference*, 2000.

[99] Alexander Schatten and A Min Tjoa. Persistent knowledge acquisition for educational purpose: An open distance learning website for german literature and language scientists. In *Proceedings of the International Conference on Information and Communication Technologies for Education*, 2000.

[100] Alexander Schatten, A Min Tjoa, Amin Andjomshoa, and Hassan Shafazand. Building an web-based open source tool to enhance project management, monitoring and collaboration in scientific projects. In *Proceedings of the International Conference on Information Integration and Web-based Applications and Services*, 2001.

[101] Alexander Schatten, Klaus Zelewitz, A Min Tjoa, and Johann Stockinger. Austrian literature moving to cyberspace? a framework for building an open distance learning website using platform independent standards like XML. In *Proceedings of the World Conference on Educational Multimedia, Hypermedia and Telecommunications*, 2000.

[102] Bruce Scheider. About microsofts trustworthy computing announcements. http://www.counterpane.com/crypto-gram-0202.html (as of February 2003).

[103] Jürgen Schmidt. Nicht trustworthy, Internet Explorer gefährdet Rechner und Netze. *ct'*, 25:100–101, Dez 2002.

[104] Pressemitteilung der Stadt Schäbisch Hall: Schwäbisch Hall setzt komplett auf Linux. http://www.schwaebisch-hall.de (as of July 2003), nov 2002.

[105] Leonard J. Seligman, Kenneth Smith, Inderjeet Mani, and Barbara Gates. Databases for semistructures data: How useful are they? (position paper). In *Knowledge Representation Meets Databases*, pages 16.1–16.4, 1998.

[106] Claude Shannon and Warren Weaver. *A Mathematical Theory of Communication*. University of Illinois Press, Urban and Chicago, 1963.

[107] C. P. Snow. *The two cultures: and a second look*. Cambridge University Press, Cambridge, 1986.

[108] Sourceforge open source repository. http://www.sourceforge.net (last accessed July 2003).

[109] Israel Spiegler. Knowledge management: a new idea or a recycled concept? *Communications of the AIS*, 3(4):2, 2000.

[110] S. Staab, H. Schnurr, R. Studer, and Y. Sure. Knowledge processes and ontologies. *Knowledge processes and ontologies. IEEE Intelligent Systems*, 16(1):26–34, 2001.

[111] The standish group report - chaos, 1995.

[112] L. Steels. Corporate knowledge management. In *Proceedings of ISMICK 1993*, pages 9–30. Compiegne France, 1993.

[113] Kathy A. Stewart, Richard Baskerville, Veda C. Storey, James A. Senn, Arjan Raven, and Cherie Long. Confronting the assumptions underlying the management of knowledge: an agenda for understanding and investigating knowledge management. *ACM SIGMIS Database*, 31(4):41–53, 2000.

[114] Nakkiran N Sunassee and David A Sewry. A theoretical framework for knowledge management implementation. In *Proceedings of the 2002 Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists on Enablement through Technology*, pages 235–245. South African Institute for Computer Scientists and Information Technologists, 2002.

[115] Keith D. Swenson. Visual support for reengineering work processes. In *Proceedings of the conference on Organizational computing systems*, pages 130–141. ACM Press, 1993.

[116] Standard for data synchronization: Syncml. http://www.syncml.org (last accessed February 2003).

[117] Palladium and the tcpa. http://www.counterpane.com/crypto-gram-0208.html, August 2002.

[118] Lester Thurow. *Building Wealth: The new rules for individuals, companies and nations in a knowledge-based economy.* Harper Collins, New York, 1999.

[119] Pierre F. Tiako, Tim Lindquist, and Volker Gruhn. Process support for distributed team-based software development workshop. *ACM SIGSOFT Software Engineering Notes*, 26(6):31–33, 2001.

[120] Gunther Tichy. Informationsgesellschaft und flexiblere Arbeitsmärkte. Technical Report ITA-02-03 ISSN 1681-9187, Institut für Technologiefolgen-Abschätzung der Österreichischen Akademie der Wissenschaften, 2002.

[121] Xml topic maps (xtm) 1.0. http://www.topicmaps.org/xtm/1.0/ (as of July 2003), 2001.

[122] Linus Torvalds. *Just for Fun: The Story of an Accidental Revolutionary.* Harper Business, June 2002.

[123] Jon Udell. *Practical Internet Groupware.* O'Reilly, Sebastopol, 1999.

[124] Victor Vianu. A web odyssey: from codd to XML. In *Proceedings of the twentieth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pages 1–15. ACM Press, 2001.

[125] World wide web consortium. http://www.w3c.org (last accessed July 2003).

[126] Andrew Waugh, Ross Wilkinson, Brendan Hills, and Jon Dell'oro. Preserving digital information forever. In *Proceedings of the fifth ACM Conference on Digital Libraries*, pages 175–184. ACM Press, 2000.

[127] Webservices specifications and information. http://www.w3.org/2002/ws (as of February 2003), 2002.

[128] Joseph Weizenbaum. *Computermacht und Gesellschaft.* Suhrkamp Taschenbuch Wissenschaft, Frankfurt am Main, erste edition, 2001.

[129] Helmut Willke. *Dystopia, Studien zur Krisis des Wissens in der modernen Gesellschaft.* Suhrkamp Taschenbuch Wissenschaft, erste edition, 2002.

[130] Ludwig Wittgenstein. *Tractatus logico-philosophicus, Logisch-philosophische Abhandlung.* Suhrkamp, Frankfurt am Main, erste edition, 1963.

[131] Xhtml 1.0: The extensible hypertext markup language. http://www.w3.org/TR/xhtml1/ (as of February 2003), january 2000.

[132] Xml linking language (xlink) version 1.0. http://www.w3.org/TR/xlink/ (as of December 2002), june 2001.

[133] Extensible markup language (xml) 1.0 (second edition). http://www.w3.org/TR/2000/REC-xml-20001006 (as of December 2002), october 2000.

[134] Xml path language (xpath) version 1.0. http://www.w3.org/TR/xpath (as of March 2003), november 1999.

[135] Extensible stylesheet language (xsl) version 1.0. http://www.w3.org/TR/xsl/ (as of March 2003), october 2001.

[136] Xsl transformations (xslt) version 1.0. http://www.w3.org/TR/xslt (as of December 2002), november 1999.

[137] Xul specification. http://www.mozilla.org/projects/xul/xul.html (last accessed March 2003).

[138] Susan E. Yager. Using information technology in a virtual work world: characteristics of collaborative workers. In *Proceedings of the 1999 ACM SIGCPR Conference on Computer Personnel Research*, pages 73–78. ACM Press, 1999.

[139] Yutaka Yamauchi, Makoto Yokozawa, Takeshi Shinohara, and Toru Ishida. Collaboration with lean media: how open-source software succeeds. In *Proceeding of the ACM 2000 Conference on Computer Supported Cooperative Work*, pages 329–338. ACM Press, 2000.

Index