COMENIUS UNIVERSITY
FACULTY OF MATHEMATICS, PHYSICS AND INFORMATICS
DEPARTMENT OF COMPUTER SCIENCE
BRATISLAVA, SLOVAKIA

Computer Supported Cooperative Work

Master's Thesis

Marek Mrázik
author

RNDr. Richard Ostertág
advisor

BRATISLAVA MAY 2007

Computer Supported Cooperative Work

Master's Thesis

Marek Mrázik

COMENIUS UNIVERSITY
FACULTY OF MATHEMATICS, PHYSICS AND INFORMATICS
DEPARTMENT OF COMPUTER SCIENCE
BRATISLAVA, SLOVAKIA

Study Programme: Informatics

RNDr. Richard Ostertág

BRATISLAVA, MAY 2007

Declaration

Hereby I declare that this thesis is my own work, only with the help of the referenced literature and under the careful supervision of my thesis advisor.

Acknowledgements

I would like to thank my diploma thesis advisor RNDr. Richard Ostertág for his valuable advices and professional help during my study and the writing of this work. I also want to thank my father Ing. arch. Augustín Mrázik who encouraged me to elaborate this topic. Furthermore, I thank Prof. RNDr. Branislav Rovan, PhD. and my student colleagues for their helpful remarks and suggestions for improvement during diploma seminars.

Marek Mrázik

Abstract

In this thesis we will introduce the reader into the research area of Computer Supported Cooperative Work (CSCW) and groupware, their definitions, history and surrounding terms. Additionally we will list most important groupware classification factors according which groupware can be classified and functionality supported by groupware systems. We will discuss the importance of each factor and choose a representative subset of factors which can be considered as a basis for groupware classification. We will also discuss some known problems in groupware development. Last chapter offers information about present groupware standards and about the failures of standardization initiatives in the past.

Keywords: *Groupware, CSCW, groupware definitions, groupware history, groupware classification, groupware development problems, groupware standards*

Preface

The scope of this work is to provide an overview of the broad research area of CSCW, associated terms and problems. We saw a challenge for this work since a complex and comprehensive overview of groupware and CSCW is not available. This overview includes a groupware characteristics table which classifies existing groupware systems and which can be extended by new ones.

The aim of this thesis can be divided into two directions. Firstly, it is targeted for groupware developers. Diverse classification factors, description of the scope and the style of groupware application use, groupware standards as well as list of possible groupware problems can be used as a source of information and a checklist before starting the developing a groupware application. Finally the developed application can be inserted into our classification system and it can be compared to other groupware applications.

On the other hand, the content of this work is also suitable for managers and groupware users who are planning to deploy a groupware system. It can help them to identify possible social and management risks of the deployment of the particular groupware application. They can use the characteristics table (appendix A) for selection of a system according to their needs and compare it to other potential alternatives. This thesis can be veiwed also as an introduction into the research area of CSCW and groupware. From this point of view it provides a view at distributed computer systems from a different perspective.

Bratislava, May 2007 Marek Mrázik

Table Of Contents

1 Introduction

For solving of complex and extensive tasks (e.g. in larger research or design projects in different fields of industry and science) a high level of communication and cooperation between the team members as well as an efficient coordination and management of the whole team is necessary. In the advent of computer systems, the word „groupware“ started to be used for this purpose. Tom Brick in [2] defines groupware as „*technology designed to facilitate the work of groups. This technology may be used to communicate, cooperate, coordinate, solve problems, compete or negotiate*.“

The scope of this thesis is to explore the state of the art, as well as new opportunities and challenges for the Computer Supported Cooperative Work (CSCW), which were brought by the rapid spread of the Internet and invention of new technologies. At the beginning, in Chapter 2, the reader will be introduced to the history and term definitions. We will also list and briefly explain other terms that are often used jointly with the term CSCW – e.g. communication, collaboration, cooperation, workgroup computing, workflow management or WYSIWIS (What You See Is What I See), which are necessary to understand the basis of this broad research area. In Chapter 3 we will analyze the major groupware classification factors, according to which groupware can be classified to categories. In addiction, we will list also all major groupware functionalities and some most important non-groupware specific factors, which are important when deciding between various groupware applications. This list has two possible goals. Firstly, it can be seen as a „groupware designer’s checklist“ of what is to be thought of before starting groupware development – which parameters the particular application have to fit and which aspects can be omitted. The second, but far not less important, goal of this list is that it forms the basis for the classification of groupware applications, which is also a part of this thesis.

An important part of this work is a table of groupware applications where most important groupware features are listed. This table is designed to facilitate users the decision of which groupware application they should choose, because it contains relevant information about the particular groupware applications. At the end of Chapter 3 we analyze the decisions of choosing the concrete subset of all features for the table and the reasons for not including the omitted ones.

In Chapter 4, we form categories for the classification of groupware according to the functionalities described in chapter 3.3. Known groupware systems are classified according to these categories in appendix B.

The main purpose of the classification itself as well as of the classification table is to support selection of the best suitable groupware application for a particular purpose. This table can be useful also for groupware developers. - they can easily categorize their system according to its functionality.

Chapter 5 describes groupware design problems from three different points of view: technical, social and management. This chapter is targeted primarily for designers and it includes the most frequent problems which cause failures of groupware development projects – not only technical and architectural, but also social which are mostly unknown by software developers. This chapter suits also for managers who are planning to deploy groupware in their team. They find useful information about what they should consider before this process and how to prepare prevention methods for possible rejection of the groupware by the team.

Finally, in Chapter 6 we describe commonly used groupware standards, as well as less successful standardization initiatives and the reasons of their failure.

The contribution of this work is that it offers an introduction into groupware and CSCW. From various aspects of this interdisciplinary research field. Finally it points the reader to additional sources of information in the particular area of his/her interest.

The aim of this thesis is to find answers and solutions to the following questions and problems:

- Are the terms groupware and CSCW in literature correctly and unambiguously defined? Is there a definition that defines groupware according to present view of groupware? If not, we will try to do so.
- What are the main requirements on groupware systems by users? It is possible to create a classification system for groupware systems? Are there any existing classifications and are they relevant in present time? How can these categories be defined? Are these categories disjoint?
- Are there any other (groupware independent) factors that influence the choice of a concrete groupware system?
- What specific problems can be identified in groupware development process? Which of these problems are seen as most critical in respect to groupware history?
- Are there any standards which should be satisfied by groupware systems?

2 History, facts and definitions

2.1 Introduction into the world of groupware

As far as history goes, from diverse reasons people form groups. Everybody would agree that almost every kind of work was and is done in larger or smaller groups. Later, in the first decades of the 20th century, men started to invent approaches to increase the efficiency and quality of work with lower effort. So management as science was born. Many important concepts into management were brought by Frederick Winslow Taylor, Henri Fayol or Alexander Church. Although management defined various models of increasing work efficiency, there is no doubt that the most important step ahead was the invention of computers. The computer was seen as an ideal work facilitator. In the mid 1960's tasks such as filling seats on airplane flights or printing payroll checks were already mastered by huge mainframe systems. By this time, computers were too expensive for personal use.

Computers implicitly supported the work of single users, but the majority of work is done in groups. The single-user concept was seen as an insufficiency of computers. The situation changed in the 1970's when minicomputers started to be built. Minicomputers shifted computers from the high-financed research laboratories into daily business life. *Minicomputers promised to support groups and organizations in more sophisticated ways: Office automation was born* [3]. Single-user applications such as word processors and spreadsheets were implemented and gained enormous success. But office automation promised more: it tried to extend these successes to support groups and departments.

2.2 A brief history

While single user application development was on its rise, group support systems were facing several problems. Gathering precise requirements for such systems was a real challenge. Building technology was not enough. A need for a better understanding of *how people work in groups and organizations and how technology affects that* was necessary. [3] This was the major impulse for defining Computer Supported Cooperative Work (CSCW). CSCW was an effort by technologists to learn from social, economic, organizational and anthropologic researchers or anyone who could *shed light on group activity.* [3] The term itself was coined by Paul Cashnam and Irene Grief for a workshop in 1984. It is though important to mention that it was Douglas Engelbart who started research in the area of technology that supports cooperative work 20 years ago. After the 1984 CSCW workshop, as in all other areas of science,

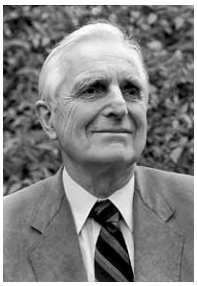
publications and conferences are the grounds for advancement. The first conference on CSCW was held in Austin, Texas in 1986. *It brought together around 300 people from a variety of backgrounds, artificial intelligence, human-computer interaction, office information systems, computer science, psychologists and anthropologists, and was by all accounts a great success* [20]. After this first conference, biannual scientific conferences on CSCW followed in the USA. With slight difference, Europe also has its approaches to CSCW which where discussed on the ECSCW conferences (European Conference on Computer Supported Cooperative Work). ECSCW conferences started in 1989 and were held every odd year.

CSCW seen as a multidisciplinary field had to overcome inevitable obstacles. The new potential audience was frustrated when the other group was ignorant to work that was to be basic shared knowledge. As described in [3] it was the „Tower of Babel“ merging different terms from different areas. An example explains that the term „user“ in information system refers to the person who uses the output of the system – he might not ever touch the keyboard. On the other hand, „user“ in human-computer interaction refers to the person sitting at the display and interacting directly with the system. After more than 20 years of CSCW research, many problems were solved and likewise many new appeared. The CSCW community grew to a stable and accepted part of the computer science spectrum.

A more detailed history of CSCW and groupware applications, as well as a list of people involved in this research area is to be found in [44].

2.3 The term groupware

2.3.1 Groupware history



Douglas Engelbart

The first computer mouse held by Engelbart showing the wheels which directly contact the working surface.

In 1968 on the Fall Joint Computer Conference in San Francisco Douglas Engelbart *flabbergasted his audience in a special session* [6] with the demonstration of NLS (oNLine System) – a prototype of a system later commercially known as Augment. NLS featured on-screen video teleconferencing, shared-screen collaboration and telepointioning between Engelbart and his college from the SRI lab in Menlo Park. Engelbart was a visionary and his work has never been easy. Through the years he has been misunderstood, told he was "dead wrong", ridiculed, or simply ignored, which many say is to be expected when one is "20 years ahead of his time". Apart from groupware, pioneered many innovations such as the mouse, 2-

dimensional display editing, in-file object addressing, linking, multiple windows, document version control, shared-screen teleconferencing, remote procedure call protocols, distributed client-server architecture and hypermedia. These ideas found their way to millions of computer users relatively quickly. Although it took longer to promote the idea of groupware to a broad mass of users, now there are no doubts about the power of Engelbart's vision and importance of this idea.

In spring of 1967, it was announced that all the ARPA-sponsored computer research labs, including Engelbart's ones, would be networked to promote resource sharing. He saw the ARPANET as an excellent vehicle for extending NLS provisions for wide-area distributed collaboration. Because of this early active role in the formation of the ARPANET community, his site was the second host on the network.

Although according to Henri Ter Hofte in [6], Engelbart is considered to be the „father of groupware", the term Groupware was brought to life by Johnson-Lenz in 1977.

2.3.2 Groupware definition

From the beginning, there was a dispute what is contained under the broad „umbrella" of groupware and where is the borderline. Seeing groupware as a tool for supporting human communication (but also collaboration, coordination and also competition), some considered a regular phone to be groupware. According to [3] Crowley *felt that network file servers and related software was of central importance and should be considered groupware.* Besides these disputes, we have to realize that groupware is not only software; it can also contain hardware parts to form the whole groupware system (e.g. a video camera for a online multimedia conference).

In 1988, Engelbart pointed groupware as: „A co-evolving human-tool system". Since then, various definitions have emerged. With changing technology and requirements, the definitions varied. A list of these definitions is to be found in the end of this chapter. This list shows how the view of this term changed with time and can help the reader to understand more accurately the development of the view on groupware systems. However, the best definition that suites for groupware in these days was coined by Tom Brick in 1998:

Groupware is technology designed to facilitate the work of groups. This technology may be used to communicate, cooperate, coordinate, solve problems, compete or negotiate. **[2]**

While traditional technologies as the regular telephone (by this definition) classify as groupware, the term is ordinarily used with respect to modern computer technologies such as email, videophone and other.

Author name (Year)	*Definition*
P. & T. Johnson-Lenz 1978	"An intentional group process plus software to support them."
D. Engelbart 1988	"A co-evolving human-tool system"
Johansen 1988	"... a generic term for specialized computer aids that are designed for the use of collaborative work groups. Typically, these groups are small, project-oriented teams that have important tasks and tight deadlines. Groupware can involve software, hardware, services, and/or group process support."
Greenberg 1991	"Groupware is software that supports and augments group work. It is a technically-oriented label meant to differentiate "group-oriented" products, explicitly designed to assist groups of people working together, from "single-use" products that help people pursue only their isolated tasks."
Opper/Fersko-Weiss 1991	"Groupware is any information system designed to enable groups to work together electronically."
Wilson 1991	"Groupware is a generic term for specialized computer aids that are designed for the use of collaborative work groups."
Lewe/Krcmar 1991	„CSCW is referred as the research field, which is focused on the role of information- and communications-technology in the group work, whereas groupware describes the concrete researched technology“ *Originally in German: "Mit CSCW wird das Forschungsgebiet bezeichnet, das sich ganz allgemein mit der Rolle von Informations- und Kommunikationstechnologien bei der Gruppenarbeit beschäftigt, während Groupware die beforschte Technologie selbst bezeichnet."*
Oberquelle 1991	„Groupware is multi-user software, which is designed and used for support of cooperative work, and it allows information and (other) materials to be exchanged electronically between the participants of the group or corporate materials to be maintained on a common space.“ *Originally in German: "Groupware ist Mehrbenutzer-Software, die zur Unterstützung von kooperativer Arbeit entworfen und genutzt wird und die es erlaubt, Information und (sonstige) Materialien auf elektronischem Wege zwischen den Mitgliedern einer Gruppe koordiniert auszutauschen n oder gemeinsame Materialien in gemeinsamen Speicher zu kooridinieren."*
Ellis et al. 1991	“computer-based systems that support groups of people engaged in a common task (or goal) and that provide an interface to a shared environment”

Finke 1992	„… groupware are software products, which allows workgroups to cooperate efficient and effective on a corporate work assignment and simultaneously contribute to the constitution and use of information in the area of the concrete working process.“ *Originally in German: "... handelt es sich bei Groupware system um Softwareprodukte, die es Arbeitsgruppen ermöglichen, effizient und effektiv im Rahmen gemeinsamer Aufgabenstellungen zusammenzuarbeiten und die gleichzeitig dazu beitragen, Informationen im Rahmen von Arbeitsprotz essen besser zu erschließen und verwerten."*
David Coleman 1992	"Computer-mediated collaboration that increases the productivity or functionality of person-to-person processes."
Petrovic 1993	„Groupware is a community usable computer-based environment which allows workgroup computing.“ *Originally in German: "Groupware ist eine gemeinschaftlich nutzbare computerbasierte Umgebung, die Workgroup Computing ermöglicht."*
Nastansky 1993	„Groupware offers computer-supported concepts for team work. It should in particular support workflow and the process-management within the communication and work interaction between coworkers in office-environment or a workgroup.“ *Originally in German: "Groupware stehlt computerunterstützte Konzepte für die Teamarbeit bereit. Insbesondere müssen dabei, natürlich, der Arbeitsfuß und das Vorgangsmanagement in den vielfältigen Kommunikations- und Abarbeitungsinteraktionen zwischen Mitarbeiterinnen und Mitarbeiter im Office-Bereich bzw. in Projektteams unterstützt werden."*
Diminik Stein 1996	Groupware is described as the practical use of the information, offered by CSCW research, in the information and communication systems, which support teamwork. Groupware is influenced by the following factors: human, task, organization and technique. *Originally in German: Die praktische Umsetzung der im CSCW - Forschungsgebiet gewonnen Erkenntnisse in ein Informations- und Kommunikationssystem, das die Teamarbeit unterstützt, wird als Groupware bezeichnet. Einflussfaktoren von Groupware sind Mensch, Aufgabe, Organisation und Technik*
Fouss & Chang 2000	„Any computer application (software or hardware) that in some way supports group activities.“
Ross 2001	Groupware is software that provides a virtual space where group interactions can take place, often in a unique way that could not happen in “real” face-to-face group situations.

2.3.3 Groupware's potential and growth in the last decades

The significant growth of interest on CSCW can be demonstrated by the increasing number of conference participants. There had been only 34 people invited to the first workshop organized by Cashmen and Grief. A year later, for the conference in Austin 300 people applied. The 3rd North American CSCW Conference held in 1990 in Los Angeles attracted around 560 delegates, with over 40 from Japan - a massive increase compared to the previous Portland Conference.

Although the interest in groupware rapidly grew, the majority of conference participants in the 80's were academic. This changed by the spread of Internet and the growth of the interconnection possibilities. However, the picture below shows the conference attendance at seven different conferences dedicated to groupware related issues. You can see a big difference between the commercial/academic ratio on the US and European conferences. Furthermore, Japan played in the groupware design its autonomous role. The reason was because Japan's management model is different US and European models. A concrete example of the difference is that Japanese contributors focused their articles on small-group applications. For more information about different approaches to CSCW in US, Europe and Japan we refer to the paper of Jonathan Grudin [3].

	USA		Europe	
	CHI '90	CSCW '86 - CSCW'90	ECSCW '89 Crete '90	ICIS '90
Academic	40 %	30 %	70 % *	85 %
Product development	30 %	40 %	10 % **	1 %
Telecommunications	10 %	7 %	5 %	0 %
Other	20 %	23 %	15 %	14 %
** includes government research laboratories* *** 2/3 from U.S. computer companies*				
ECSCW – European Conference on Computer Supported Cooperative Work *CSCW - Conference on Computer Supported Cooperative Work* *CHI – Conference on Computer-Human Interaction* *ICIS – International Conference on Information Systems*				

Image 1: Table of percentages of conference predicaments divided according continents (USA and Europe) and industry. This table shows that the majority of participants of European conferences were academic, whereas in the USA the ratio was rather equal.

In these days, groupware is gaining great popularity. The spread of Internet and lowered costs of the technology allowed the use of groupware not only by huge corporations, but also by small companies. Here we list some of the most relevant reasons (according to [2]) why groupware is being deployed:

- To facilitate communication: make it faster, clearer, more persuasive
- To enable communication where it otherwise would not be possible
- To enable telecommuting
- To cut down on travel costs
- To bring together multiple perspectives and expertise
- To form groups with common interests where it wouldn't be possible to gather a sufficient number of people face-to-face
- To save time and cost in coordinating group work
- To facilitate group problem-solving
- To enable new modes of communication, such as anonymous interchanges or structured interactions

2.3.4 The term CSCW

The acronym CSCW stays for Computer Supported Cooperative Work. Some argued that four words are too many and used a shorter form mainly CSC – computer supported collaboration. Some opened the question whether cooperation is broad enough and covers also collaboration, coordination and more. There were also those who thought that „cooperation“ is often more a goal then reality. Further details about the dispute of the CSCW acronym are in paper [29].

In spite of serious criticism the acronym, CSCW survived unchanged since 1984.

Similarly to groupware, also CSCW’s definition causes inconsistent opinions. Some authors view CSCW as a paradigm; some even ask whether it is really necessary to define CSCW exactly. Almost ten years after the CSCW was mentioned for the first time, Wilson stated that *there is still no commonly accepted definition of CSCW.* The majority saw CSCW as an „umbrella term“ that allowed people from different disciplines come together and discuss issues without any common ground. *Indeed, whether CSCW can be viewed as a new field of research in its own right has been questioned by some. … Rob Kling has spoken of CSCW as an "arena"*

where different groups vie for the attention of participants, rather than a coherent focused field. [20]

Though there are newer definitions then the first definition of CSCW be Irene Grief, we believe that this correctly states what should be under CSCW understood. We list the all CSCW definitions in the table below.

Name (Year)	*Definition*
Greif 1988	CSCW is an "identifiable research field focused on the role of the computer in group work."
Bannon et al. 1988	"We believe that for the moment the name CSCW simply serves as a useful forum for a variety of researchers with different backgrounds and techniques to discuss their work, and allows for the cross-fertilization of ideas, for the fostering of multidisciplinary perspectives on the field that is essential if we are to produce applications that really are useful."
Suchman (1989)	"…the design of computer-based technologies with explicit concern for the socially organized practices of their intended users."
Greenberg 1991	"CSCW is the specific discipline that motivates and validates groupware design. It is the study and theory of how people work together, and how the computer and related technologies affect group behavior."
Grudin 1994	Computer Supported Cooperative Work (CSCW) is the study of how people use technology, with relation to hardware and software, to work together in shared time and space. CSCW began as an effort by technologists to learn from anyone whom could help them better understand group activity and how one could use technology to support people in their work. These specialists spanned many areas of research, including economists, social psychologists, anthropologists, organizational theorists and educators.
Hasenkamp 1994	We can identify three tightly-coupled CSCW research fields, namely: 1. The development of mind for team work and coordination 2. The development of concepts and tools for support of work processes 3. The evaluation of these concepts and tools *Originally in German: Es können drei eng zusammenhängende CSCW - Forschungsbereiche unterschieden werden:* *1. die Entwicklung eines Verständnisses der Zusammenarbeit und Koordination* *2. Entwicklung von Konzepten und Werkzeugen für die Unterstützung arbeitsteiliger Prozesse* *3. Bewertung dieser Konzepte und Werkzeuge.*

Bornschein-Grass 1995	„CSCW is the research field which is focused in general on the role of information and communication technology within the boundaries of cooperative work. “ *Originally in German: " allgemein mit der Rolle der Informations- und kommunikations- -technologie im Rahmen kooperativer Arbeit beschäftigt, indes Groupware die beforschte Technik bezeichnet."*

We see that although the definitions vary all agree on CSCW being a mixture of computer science, psychology, sociology, economy, management and more research fields. This is illustrated in the picture below.

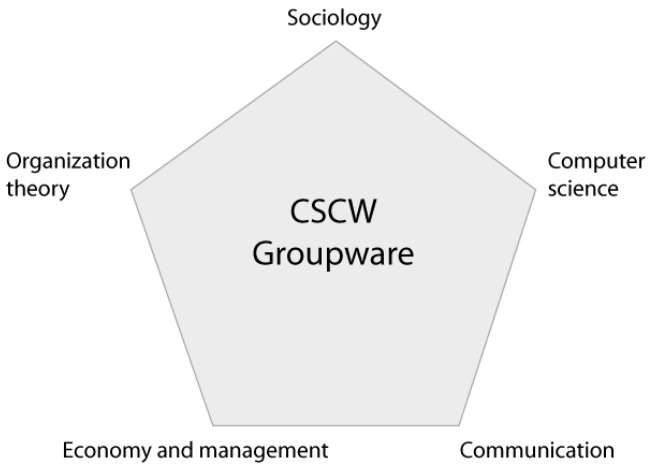


Image 2: The placement of CSCW within other research disciplines.

2.3.5 Other terms around CSCW

As already stated, there are several terms that are frequently used in the CSCW surrounding. We introduce the most important ones in this chapter.

2.3.5.1 Workgroup computing vs. workflow management

According to [16] *there are several reasons to differentiate between these two* types of groupware. We listed the most relevant of these below:

- *The most significant difference is the focus. Workgroup computing focuses on the information being processed, enhancing the user’s ability to share information within workgroups. Workflow emphasizes the importance of the process, which acts as a container for information.*

- *Workflow systems need a set of rules to define the steps for the problem solving. Workgroup computing is more flexible and spontaneous.*
- *The user controls "workgroup computing" tools. The user initiates the interaction. Workflow is defined at the beginning of the process and then the Workflow system initiates the necessary actions to finish the task (computer-mediated communication).*
- *The basic idea of Workflow is to divide the problem into several smaller sub-problems, which can be solved by different people. "Workgroup Computing" focuses on people working together at the same time to solve one big problem.*
- *The number of participants in Workflow systems can be large, but in workgroup systems, the number of people involved in the solution of a problem is still limited because of the difficulties in concurrency control and coordination.*

The interactions between workflow management, workgroup computing, groupware and CSCW is according to [1], illustrated on image 3. In this image, we see that workflow management is simply a subset of groupware. Workgroup computing certainly interrogates with groupware but we believe that is not a subset of groupware – which is also to be seen on the picture. Therefore there is no reason to dedicate more space to it in this thesis. More information about this approach and differences described above can be found in [16].

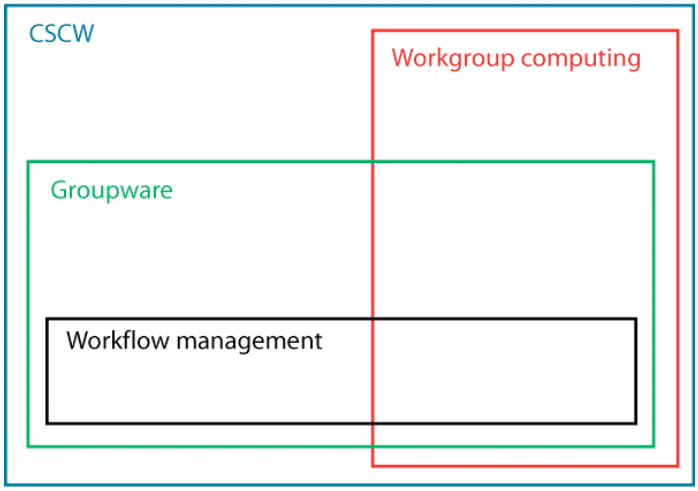


Image 3: The diagram of relationship between CSCW, groupware, workgroup computing and workflow management.

2.3.5.2 Communication vs. collaboration vs. cooperation

It is very important to state the difference between these three frequently used terms. In the dispute of the correctness of the CSCW acronym in [29], they believe that the C for *Cooperative*

is used to refer to all group interactions such as competition, conflict, and cooperation. In this respect we think that the C in „CSCW“ covers also communication and collaboration. Furthermore, similarly to [31] we believe that these three terms form the extremes of a triangle where groupware systems can be put in on a certain coordinates. For more information and an example picture of such triangle, see chapter 3.2.17.

2.3.5.3 What-You-See-Is-What-I-See (WYSIWIS)

Synchronous groupware applications provide data sharing and seeing the changes made by other participants as they arise. This is called the What-You-See-Is-What-I-See (WYSIWIS) capability. There are different levels of WYSIWIS:

- *Strict level of WYSIWIS* - Applications that support strict level of WYSIWIS allow all participants seeing the same screen simultaneously. That includes all cursor movements by other users, new characters or graphics symbols added, and synchronization of scrolling.
- *Relaxed level of WYSIWIS* - This level of WYSIWIS allow all users to work on different parts of a shared object (document or design) and all copies of the document are kept up-to-date in real-time. Should users work on the same segment of the document, their changes would appear on each other's screens in real-time.
- *Time-relaxed WYSIWIS* - Lafon and Karsenty in [39] introduced another level: time-relaxed WYSIWIS. It defines a situation where the users can switch to a ‘off-line’ mode and their modifications do not appear on the other participants screens. After they finish their work, they can decide whether to cancel their actions or to commit them to all users and synchronize their views. In this mode all other users actions are buffered and executed when the user synchronizes. The most advantage of this approach is that the users, which are not familiar with the application, can train without disturbing other participants. More about time-relaxed WYSIWIS can be found in [39].

2.3.6 Interaction of CSCW with other fields of science

As stated above, CSCW interrogates with many other fields of science. From the broad range of interaction aspects, we chose some important ones, which are closer to CSCW than others or are frequently discussed in literature:

- *Distributed systems* – groupware users are often present on different places and in different time. So standard distributed system problems like data consistency, concurrency control and other have to be solved. Some of these problems are discussed in more detail in chapter 5.
- *Multimedia* – using diverse types media for displaying different information and enabling better user interaction. Because of the rapid spread of broadband Internet connections, multimedia's role in groupware rose. An example of using multimedia in groupware is audio and video-conferencing or shared whiteboard drawing.
- *Communication* – The information exchange between users needs suitable communication protocols, efficient usage of the bandwidth, the choice of concrete transport format.
- *Human-computer interaction* – the knowledge from HCI has to be reconsidered for enabling group communication. In CSCW, this implicates human-computer-human (HCHI) interaction. The difference between HCHI and HCI is that while in HCI the sides of communication are a human and a computer, in HCHI both sides are formed by humans and the computer is only an intermediate channel, which enables this communication.
- *Artificial intelligence (AI)* – Automation of group activities using heuristic of learning agents for simulating of different group behaviors. One example of using Artificial intelligence in groupware applications is learning group behavior and in this way e.g. establishing or adopting automatically the workflow process according this gained knowledge. Another example of AI in CSCW can be agents, which can simulate thousands of users during the testing phase. Such a testing might have a positive impact on the development, especially because testing in groupware is a known problem (see chapter 5.1.8).
- *Sociology* – Group work involves people. CSCW needs to understand how people in a group interrogate, how roles are distributed in groups.
- *Organization theory* – Group work is in most cases done within one or more organizations. By deploying groupware to an organization, all structures of the organization have to be prepared and aware of the change.
- *Management theory* – Management defines work distribution, controlling and reporting. Groupware in many cases is a suitable tool for supporting these goals.

- *Graph drawing* – Many features of groupware like workflow in an organization, project tracking or the style of communication between participants can (or must) be visualized by graphs. Correct planar graph drawing though plays an important role in these systems.

3 Factors for groupware classification

This chapter gives an overview of major factors, according to which groupware can be classified. In chapter 3.1 we define and describe the *groupware characteristics table* which is shown in Appendix A of this thesis.

Later, in each subchapter of chapter 3.2, we discuss the meaning of each factor, possible ambiguity of this factor, if differences were found in literature. These factors will be the base for the classification later in chapter 4 of this work.

Another way to look at these factors (especially useful for groupware developers) is to consider them as a checklist, which has to be read before developing a new groupware application. In groupware development it is necessary not just knowing all the functional and non-functional requirements and be aware of possible development problems, but even more having the sense of all other (e.g. social) factors that will effect the final deployed application. This list should suite as a basis for gaining knowledge of the environment of the future application.

3.1 The groupware characteristics table

The groupware characteristics table is shown in appendix A. It lists some of current groupware systems and their classification according chosen factors. The columns of the table are divided into three categories: groupware specific factors, groupware independent factors and all possible groupware functionalities. Each of these subcategories is described in more detail below.

Firstly, we want to mention that groupware independent factors mentioned in chapter 3.4 are all included in this table. They were chosen from a larger group of software characteristics with the emphasis on what regular groupware users as well as managers look for when choosing a concrete groupware application.

Another group of parameters (columns of the table) are the groupware functionalities listed in chapter 3.3. For every groupware application we decide whether it is capable of this functionality or not – e.g. users can communicate using synchronous messages or not.

We would like to remark that most classification factors are relevant for the whole system, but there are some of them which are feature specific. For more complex systems included in the table, it was necessary to provide information about each of the groupware subsystem (functional entity – e.g. the messaging subsystem). The reason is that e.g. while a concrete system's messaging subsystem is only asynchronous, its shared drawing functionality is

synchronous. The most important subsystems of concrete groupware systems are listed indented under the system's row.

The last group of parameters are groupware specific factors, which were chosen as a subset of all factors listed in chapter 3.2. In the past paragraph of each subchapter of 3.2, we discuss why we included or not included the particular factor into the groupware classification table.

3.2 Groupware specific factors

3.2.1 The time factor

The most often mentioned characteristic of groupware in literature is indisputably the time factor. In majority of cases groupware is being divided into two time categories:

- **Same-time**, also referred as synchronous or real-time (when it is able to communicate only if the particular receiver is present at the same time - e.g. chat, audio/video communication).
- **Different-time**, also referred as asynchronous (when it is able to communicate even if the receiver is not present and it is expected that the message is received by the receiver when he emerges online e.g. e-mail, newsgroups or forum)

In other literature there are more categories to describe the time factor. In [3] we can find a three-category time classification, which divides asynchronous time to „different time but predictable" (meaning the e.g. message will be read in during the day it was sent) and „different time and unpredictable (posting a message on a message board, where it even is not guaranteed that the message will be read). Finally in [21] time is divided into four categories:

- ***Synchronized*** *collaboration must happen in a structured manner at the same time. Synchronized Groupware will handle locking and collision detection in real-time. (e.g. chat systems).*
- ***Unsynchronized*** *collaboration can happen entirely unsynchronized. Unsynchronized Groupware supports people working together, completely separate from each other. Collaboration only kicks in when requested from a user, otherwise all work performed does not affect other collaborating users. (e.g. forums).*
- ***Mixed (Synchronized & Unsynchronized)*** *collaboration can be either synchronized or unsynchronized. (e.g. ICQ which allows sending of synchronized messages, but also*

allows sending messages to offline users – they will be shown to the user after he logs into the system).

- ***Serial** collaboration are unsynchronized with the exception that one user must perform a specific task before another user can continue with another task. Email is a classical example of serial collaboration. [21]*

We decided to use the two factors time classification, which is commonly used and satisfactory for this thesis. Other categories for time, as described above, would not make positive differences to the results of the classification, on the contrary they would make it less comprehensible.

The only category for time (not used in the classification) we would like to give in consideration to is the *Serial* category, which covers more then only a time information. It also tells something about the method of the system to coordinate users activity. This classification factor is covered by the 3.2.6 „User activity coordination“ therefore using this factor in the time classification would be redundant.

As the time factor is one of the most often used factors in literature we decided to ***include*** it in the „groupware characteristics table“.

3.2.2 The place factor (location of group members)

The place characteristic describes the location of group members in relation to each other. This characteristic divides groupware into:

- **Same place**
- **Different place**

Although there is no need for more explanation on this characteristic, there were some different place categories to find in [3]. As well as in the time characteristic, also different place was divided into „different but predictable“ and „different and unpredictable“. Examples of predictable and unpredictable features can be found on image 5 in chapter 3.3.

As the place factor describes one of the most important characteristics of groupware use and it is also frequently used in literature, it is obviously ***included*** in the table.

3.2.3 Time – place matrix

Combining time and place dichotomies gives as a result a two dimensional matrix. This *time-place matrix* [6] is probably the best known groupware systems classification. Groupware systems can be classified by placing them into one of the matrix's quadrants.

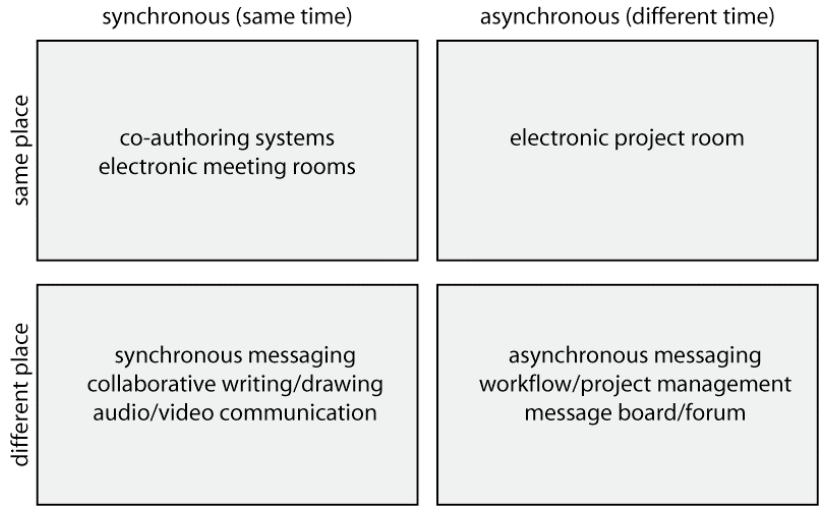


Image 4: The time – place 2x2 matrix

However this most-cited 2x2 classification matrix has been subject to considerable criticism. One reason was the strict division between the quadrants (no space left for intermediate stages). This problem can be solved by classifying groupware by placing them into more than one quadrant. Another problem is the fact, that several groupware systems support both (synchronous and asynchronous) type of communication. Though, this is a considerable fact to all classifications and the solution is to classify the concrete groupware functionalities separately. Finally [6] states another problem: *Whether the classification should be based on the functionality provided by the groupware system, or on the way this functionality is used.* Meaning that asynchronous massaging systems (e-mail), when used with high frequency, can be considered as synchronous messaging and on the contrary synchronous messaging can be used asynchronously.

By considering the third time and place category in [3] gives us a 3x3 matrix which divides groupware into 9 parts (instead of 4, showed in the 2x2 matrix). It is important to remind again, that many groupware applications may actually overlap and fit into multiple locations in this grid.

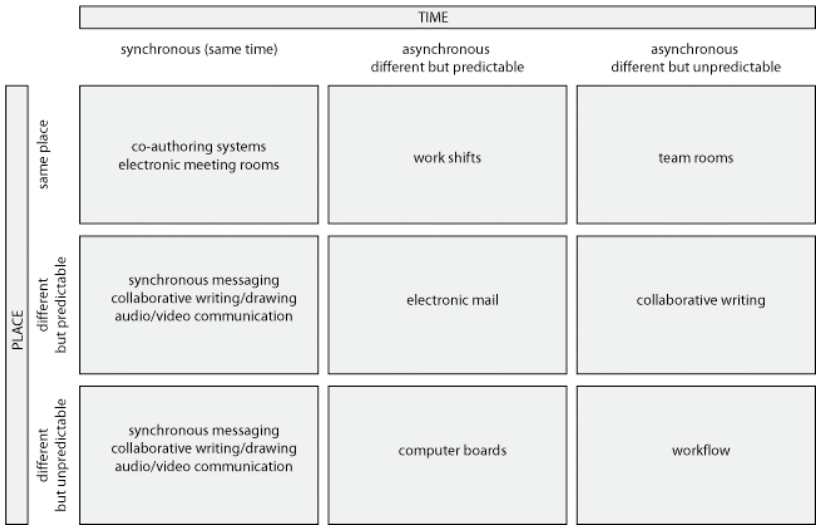


Image 5: The time – place 3x3 matrix

Although the time-place matrix shows visually and intuitively the main functionality of the particular groupware system, it is only a mixture of the time factor (chapter 3.2.1) and place factor (chapter 3.2.2). For this reason we ***do not include*** it in the characteristics table.

3.2.4 The architectural criterion

This criterion is a significant feature of groupware system. It defines which parts of the system run on a central server, which ones run on decentral parts of the system and the logical linking between these parts. The choice of the system's architecture is one of the most important decisions in groupware development. *The debate between centralized versus replicated architecture for multi-user applications is an old one. The two primary issues are performance and consistency* [28] which have affect on some of the most important attributes of groupware systems (for developers as well as for users) e.g. application's response time, scalability, storing state of communication, and others. However, often in literature you find these types of distributed architecture:

- **Central** – The collaboration is managed at a central server. These architectures are often called single-site execution and relates directly to client-server architecture.
- **Replicated** – The application runs on each user's site and the collaboration is managed by all the peers in the network. This architecture relates to pure Peer-to-Peer architectures and can be referred to as multi-site execution. An example could be a peer-to-peer audio communication application.

- **Hybrid** – In hybrid architecture, some components of the application run centralized, some run replicated. An example of such a hybrid architecture is a chat system which needs a central sever to initialize itself and to get information about which users are online. After this initialization phase all communication between the users runs replicated.

Every architecture have its pros and cons. In centralized architectures managing consistency is far easier than in any replicated architecture. On the contrary replicated architectures provide better feedback to the user since local inputs are handled locally.

According to [6] most of the newer applications would be all classified as hybrid architectures. Another problem with classifying whole systems is that for some groupware applications there are some services centralized (storing history of a chat communication) and some services replicated (direct voice communication between the users). In our opinion when talking about architecture a finer granularity should be used, and the architecture of concrete subsystems has to be considered, because calling such a system *hybrid* would be blurring of the details.

In [22], two more categories expanding the *centralized-replicated-hybrid* classification can be found: **Asymmetrical structures** (with no central component, but distribution among the peers is not symmetrical) and **Multiple servers** (with more than one central server). The paper also presents twelve distributed architectures with detailed focus on them. For this work it is sufficient to use the *centralized-replicated-hybrid* classification with a slight difference. When a groupware is classified as hybrid, we will choose finer granularity and classify the distribution architecture for every systems service.

The architecture of the system is interesting information not only for groupware developers. The architecture implies several other factors such as deployment complexity, storing data, storing collaboration state (e.g. easier in centralized architectures, more difficult in replicated architectures), hardware costs and others. This is one of the reasons why we decided to ***include*** this factor into the characteristics table.

3.2.5 The functional criterion

This criterion specifies the functional expectations of the user on the system. We found several different classifications according to functionality. As this is in our opinion the most important classification, we describe in this chapter one of these classifications in more detail. At the end we discuss the usefulness of this classification and state our solution to functional classification.

In [21], there are following categories for the functional criterion to be found: Messaging, Conferencing and Electronic Meeting Systems (EMS), Group decision support systems (GDSS), Document management, Document collaboration and Compound Document Management.

However in [29] an almost similar classification could be found. This classification divides groupware to message-based systems, conferencing systems, GDSS, multi-user editors, collaborative programming and coordination systems. By realizing that

a) *multi-user editors* are equal to *document collaboration,*

b) *collaborative programming* (as it is defined in [29] is equal to *document management),*

we see, that the taxonomy from [29] not only covers the classification from [21], it extends it with the *coordination systems* category so, that it covers also workflow systems which was a insufficiency of the classification in [21].

But also [29]'s classification has its weakness because the names for categories are better chosen in [21] (*document management* is more suitable description then *collaborative programming*). Let us describe the classification from [29] in more detail:

- **Message-based systems** – Message–based systems are the most common used systems in real world, mainly because of email, the oldest widely used internet form of communication.
- These systems provide synchronous or asynchronous messages functionality. In past, there were several attempts to enhance email by e.g. adding additional fields to a standard email message. This would make email more structured, and more suitable for collaboration. This idea was implemented in COSMOS (released in year 2000) and others products, but these systems were not accepted by users because they did not see the benefit of the additional work. For more about enhancement of messaging systems We refer to [29].
- **Coordination systems -** Coordination systems are systems dedicated to help group members coordinate their activities. Applications that fit into this category are group calendars, project management systems and workflow systems.
- **Conferencing Systems**– These systems, on the contrary to Messaging systems, where the focus lies on asynchronous communication, provide users with functionality of

a shared communication channel, which allows multiple users to collaborate on a problem simultaneously (synchronously).

- **Group decision support systems (GDSS)** – GDS systems are designed to help in the decision making process in either to speed up this process or to improve the quality of the decision. It assures that all group members have access to the same data source on a given subject.
- **Multi-user editors** – Systems in this category allow multiple users to edit a single document including text, graphics or a combination of them.
- **Collaborative programming** – There are two types of systems that are covered by this category. First there are tools designed to help groups collaborate on a programming project including CASE tools and version control. Second type of systems are development kits. These provide three main services: a framework for supporting groupware, pre-built groupware applications and necessary libraries to create new applications. Examples of such systems are DistView of GroupKit.

There is a more extensible classification of functional categories to be found in [6]. However this classification factor is far not detailed enough for users to provide them with satisfactory information to choose a concrete groupware system. We think that the classification according functionality should be far more extensive.

This reason for we decided to list all functions that can be provided by groupware systems in chapter 3.3. and include them all in our classification table (appendix A). In the final classification (in chapter 4) we will for every category choose a minimal subset of the functions listed in 3.3. As this subset is the real functionality provided to the user by the system, this functional criterion will be the most important criterion in the classification.

Reflecting to the unambiguity which is described above and the fact that the table includes all possible groupware functions, it is not necessary to include this classification factor. This factor is therefore ***not included*** in the table.

3.2.6 User activity coordination

User activity can be divided into:

- **sequential** – Sequential method is defined so, that two different users can not work on a project (problem, item …) simultaneously. After a user finishes his work with the item, the item is passed to the next user waiting to work with it.

- **parallel** – Parallel method of coordination allows users to work independently on different parts of the project and their work can be done simultaneously.
- **reciprocal** – In this method multiple users work together on the same part of the project.

The closeness of collaboration factor (described in chapter 3.2.7.) covers some of the information provided by the user activity coordination factor. Further it is more useful for the user as it gives more social information about the collaboration. It does not only technically describe the type of collaboration (if it is sequential, parallel or reciprocal), it gives you the sense what is the groupware build for: division of labor or sharing mind. These were the main reasons why we ***do not include*** this factor into the „groupware characteristics table".

3.2.7 Closeness of collaboration

The **closeness of collaboration** classifies groupware by defining a spectrum to measure how closely together team members work. This characteristics is quite similar to the coordination characteristics described above.

- **division of labor** – means dividing work to individual group members. Each of them works on their assigned task (parts of the project), which are combined together. The collaboration takes place in combining the parts into a whole.
- **sharing mind** – here, instead of dividing to parts, the project is being completed by all group members together simultaneously. An example of such groupware application is a same-time (synchronous) collaborative writing application.

At the first sight, there are no major differences between the 3.2.6 and 3.2.7 classification factors: division of labor is similar to a merge of parallel and sequential methods, and sharing mind is similar to the reciprocal method. Looking further on 3.2.7 there is a social dimension to be found (which is not included in 3.2.6). The factor in 3.2.6 does not provide information when the collaboration takes place. Division of labor means the work is divided into parts and every worker is responsible for his part. The collaboration takes place when these parts are combined. This is different to sequential coordination, where group members can work on the same part of the project.

As described in the end of chapter 3.2.6 this factor provides the user with useful information (technical and social) and is therefore in the table ***included***.

3.2.8 The focus criterion

The Focus criterion defines where the focus of the collaboration is.

- **user centered** collaboration poses the importance on the user. A communication channel is created for the users; with no limitations what the users do with the channel. The goal is to give the users the possibility to communicate (not depending on what are they communicating about). A simple chat system would fit into this category.
- **artifact centered** groupware focuses on the artifact – on the outcome of the users collaboration. For example in collaborative programming, the focus lies on the result of the collaboration – the final source code. Another example is a resulted image drawn by participants of a shared whiteboard session.
- **workspace centered** Groupware can be considered as an enhancement of user centered groupware. The difference is that a workspace is not dependent on the users. The workspace can store the state of the collaboration even if no user is present. An example is a message board, where the workspace remains after all users leave the communication and new (possibly different) users can join the communication afterwards. The stored message board cannot be considered an artifact, because there must not necessarily be an outcome of the communication.

The focus criterion is another social factor which helps the user to have a better understanding of the system. Usually users are aware of what they are expecting from the system. Either it is a communication channel between two or more users (*user centered*), or a shared environment where users can communicate, but there is an outcome of their communication which is not dependent on participating users (e.g. stored forum state – *workspace centered*). Finally *artifact centered* implies that the users will work together to create an object (e.g. shared drawing – where the outcome is a drown picture). In this respect we ***include*** this factor into the „groupware characteristics table“.

3.2.9 The user involvement criterion

This criterion defines how involved the user has to be to take advantages of the groupware system.

- **High** user involvement means that the user, to use the collaboration functionality, has to work with a different user interfaces that he is used to. For example a user is working with a database system. When he needs new information that has to be inserted into the

database, he has to switch to a mail client (which is a different application) and using copy/paste to insert them into a database system.

- In **Medium** user involvement, the users work with their default user interfaces. The collaboration takes place by execution special collaborative commands of the system. (e.g. the database system has an embedded email client).
- **Low** user involvement implies the user only sets up the collaboration once and then continues working normally. The collaboration features of the system are fully integrated in his default user interface and automated so that he does not have to even realize, that there is a collaboration taking part. (e.g. the system automatically filters the information from the email and the user only need to approve them).

Considering that „of the shelf" groupware systems (which is the case for most of the listed systems) can not be interconnected with all other systems automatically, the majority of systems will be classified as high user involvement. However this criterion can also cause ambiguity. An example could be a concrete system user by a secretary can be classified as low involvement – the secretary reads e-mail messages and the system automatically extracts the contact information from the message to their address-book. Her manager (using the same system) classifies the system as high user involvement system – as he has to copy/paste the email message into another project-management tool. This implies that user involvement would have to be defined for every systems role separately which would be non trivial to do for any groupware system. This lead us ***not include*** this factor into the table.

3.2.10 Restrictive vs. permissive groupware

The purpose of the system is another classifying criterion. Groupware can constrain or restrict the behavior of the users. An example would be a workflow system which prescribes the user what to do in diverse situations or states in his work. We call this **restrictive groupware**. Contrary to such a system is **permissive groupware** which gives the user the possibility to take any action at any time. These systems does not direct the users behavior, they leave the coordination on the user decision. The example of such a system is a shared whiteboard application, allowing drawing any user anything at any time.

Not all groupware systems can be considered strictly restrictive or permissive. Taking in consideration a audio conferencing system (permissive) with a user as the moderator who

decides who is the next to speak (restrictive). It is a restrictive behavior in a permissive system. We can go more in detail and consider restrictiveness for every single system's function.

Another alternative is to consider functions restrictive only within the system. This situation arises when one user is allowed to do more actions than another – then the second user is restricted. Example (in a shared whiteboard): User nr.1 has the possibility to draw in any color. User nr.2 is restricted only to black color. A different situation is when the system itself only allows black color – then all users have the same possibilities – so this is not considered restrictive.

We are sure there are even more types of view on restrictiveness and its granularity. Therefore we agree with [6] that *Restrictive and permissive groupware are proposed as extremes on a continuous scale.* In this generalization of this factor it is easier to define restrictiveness for any system and more intuitive the user to understand.

This classification factor gives again the user social information about the way the system can be used. Therefore we believe that it is important to ***include*** it into the characteristics table.

3.2.11 Information sharing vs. information exchange

Information sharing is a functionality which gives the users the possibility to view and change a shared object. The object is virtually located on a shared workspace. All actions done on the object by the users are interpreted by the system and resulting in an update of the state of the workspace. Other systems, such as messaging systems, are based on **information exchange**. This means that the information is not kept on a shared place. The information is directly exchanged between the participants of the communication without interpretation by the system.

The information provided by this factor can help user to decide if he wants to create an information base (information sharing) or if he wants just exchange information with other users (information exchange). This causes its helpfulness in deciding for a concrete groupware application and is ***included*** in the „groupware characteristics table".

3.2.12 Level of computer use in collaboration

Another classification is the **degree to which the technology supports group members**. It divides groupware applications into:

- **f-groupware** – In this type of groupware, there is a single workstation controlled by one person, which is organizing the meeting (the ***f***acilitator). Afterwards he inputs the

gathered information and results of the meeting into the computer. The software is used only as a tool for interpreting and organizing group's brainstorming.

- **k-groupware** – Similarly to f-groupware, also here, only one coordinating workstation is used. The difference is that each member inputs information directly into the system (through a ***k***eypad).
- **w-groupware** – In a w-groupware system, each group member has its own workstation, which leads this category being the most flexible and allows greater efficiency in capturing data.

For more details about this classification we refer to [29].

First of all we think that this classification is made-up and synthetic. Moreover in these days all users working with groupware are equipped with their own workstation. Therefore we think that all current systems will fall in the w-groupware category. This fact places this classification factor out of date and that is the reason ***not including*** it into the table.

3.2.13 Common task dimension

Common task dimension is another classification factor. It measures how tightly coupled the users of a groupware are. On the one side of the spectrum, there are **loosely coupled** users, for example users working on a mainframe. All users work on different tasks independently. The combining factor is the mainframe which all are working on – this makes them a group. On the other side of the spectrum there are **tightly coupled** users. An example of these can be a collaborative writing system which allows users to work on the same piece of the project at the same time. The combining factor in this case is not only the same machine they are working on, it is the work which is being done jointly.

We decided ***not to include*** this characteristics table factor because, as well as the user involvement criterion, this factor can also be interpreted in many ways – some users would classify a concrete system as loosely coupled and another one would have the opposite point of view. Moreover this factor is dependent on the size of the group (supported group size – chapter 3.2.18.) which is in the table included.

3.2.14 Shared environment dimension

Shared environment dimension indicates the level of information the system provides about the environment and/or the users. From e.g. email, which provides only a small amount of

information as well about the environment as about the user. On the other hand there can be a company workflow system, which has to provide far more information about the user (name, position, department, role in project) and about the environment (project goal, project status, kind of company, workflow model, etc.).

As the previous factor, also this one is dependent on a concrete user's consideration and the concrete use of groupware. In some systems (and for some users) the user ID is more information than in another system the users full address and personal information. Another reason ***not including*** this factor into the table is that in case of a system which allows providing much information about a user, it might not be necessary to fill out all this information. Then the system provides different levels of information for different users.

3.2.15 Type of system output

Type of output supported by the groupware system is also a classifying factor. It escalates from **share opinions** (lowest level of output – assuring that group members are aware of all other members opinions on the problem), through **shared opinions** (middle level of output – assuring a consensus about the project's goals and priorities done by all members) to **shared mental model**, which indicates that all group members do exactly equally understand what is the project's goal and the steps to achieve this goal. More in reference [29].

Seeing that the majority of groupware systems which are commercially used does not support shared opinions nor shared mental model type of output only one possibility remains – mainly share opinions. Therefore this factor will not provide useful information so it is ***not in need to include*** it into the table.

3.2.16 Strategy of concurrency conflict management

There are many schemes how to manage concurrency in traditional distributed architectures. Groupware, as a distributed system, faces the question what strategy is used for treating conflicts. As We will show in chapter 5.1.4 groupware has to be treated differently. Concurrency management strategies can be divided as follows:

- **serialization** – consistency is managed by algorithms that synchronize actions, so that atomic transactions are executed serially on all replicas, or repair the effects of actions that were received out of order.
    - **non-optimistic serialization** – allows only submitting new actions when it is sure that no other action will be processed.

  - **optimistic serialization** – as in praxis conflicting actions are rarely received, it is more efficient to execute the waiting action, and when a conflict rises, running mechanisms to repair the state.

- **locking** – locking gives privileged access to an object for a certain time.
  - **non-optimistic locking** – user must wait until the lock is released and only then is granted access to the concrete object.
  - **semi-optimistic locking** – while users are waiting for the lock, they perform the action. If the lock is not granted, the action must be reverted. It is not allowed to proceed to another action until the lock to the performed action is not cleared.
  - **fully-optimistic locking** – extends the semi-optimistic locking with the possibility of proceeding to next actions. However if any lock is denied, all actions have to be undone. This implies that a full history of all actions and states has to be stored until all tentative locks are approved.

The strategy of concurrency conflict management is a very important factor of groupware. In our opinion a system that does not have any concurrency conflict policy is not usable, because as groupware systems are build to cooperate, in all groupware systems it is expected that more than one user will interoperate with the system at the same time. However it would be very difficult to acquire information of all systems included in the table for all subcategories for this factor. Therefore we only state if there is a concurrency conflict management solution in the system or not. This factor is ***included*** in the „groupware characteristics table".

3.2.17 3C classification of supported functions

This classification tells about the groupware system what is its main goal. Weather is it

- **communication**,
- **coordination** or
- **collaboration**.

Each groupware can be placed into a triangle with the corners (extremes) described by each of the above mentioned functions.

The 3C factor is defined as the ratio of distance between the placement of the groupware and the corners of the triangle. This definition implies that for example conferencing systems can also be used to coordination and cooperation, but communication is more important than the others.

The placement of the systems is also affected by the group size - see chapter 3.2.18 for further information.

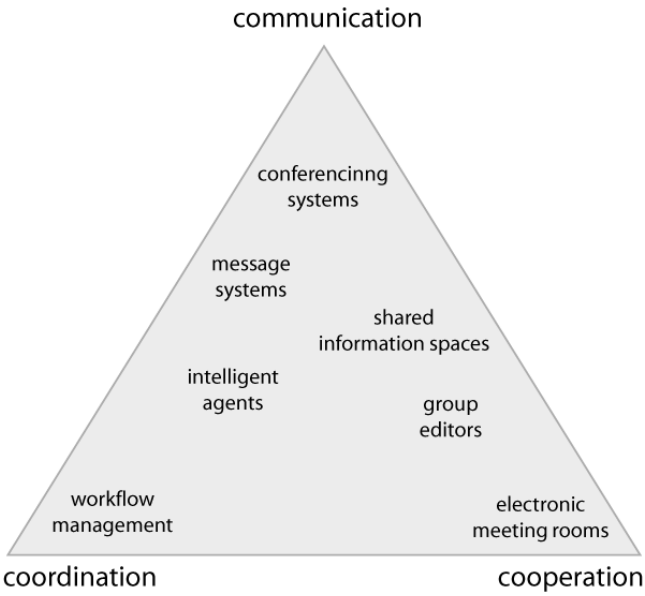


Image 6: The 3C triangle

We decided to ***include*** the 3C classification factor because it is an easy to decide classification factor which provides the user with additional information about in which part of the 3C triangle a concrete system would lays. As it is not possible to give exact information for this factor in a way that the reader can clearly understand (we would have to use a percentage ratio e.g. 20%/70%/10%), we will fill in the table the nearest corner of the triangle – e.g. for a messaging system it would be communication.

3.2.18 Supported group size

The size of the group which is using the system can be a continuous scale between a **team** (group) and a **community**. The group size has a significant influence on several groupware factors such as degree of interaction or focus criterion. The dependencies of some of these are shown on Image 7.

	Group*		Community
Size	Small	⇔	Big
Degree of interaction	Tight	⇔	Loose
Motivation / Orientation	Common goal	⇔	Shared Interest
Objectives of Work	Defined and Shared Objectives	⇔	Occasional Information Exchange
Personal Relationship		⇔	Individuals don't know each other
* Groups have usually defined inner structure and administrative regulations			

Image 7: group and community relationships

With the group size and the degree of interaction also the main goal of the group changes from **Information** (where the sender and receiver mostly even don't know each other) continuously through **Coordination** and **Collaboration** to **Cooperation** (where all users know each other, and have the same goal trying to achieve).

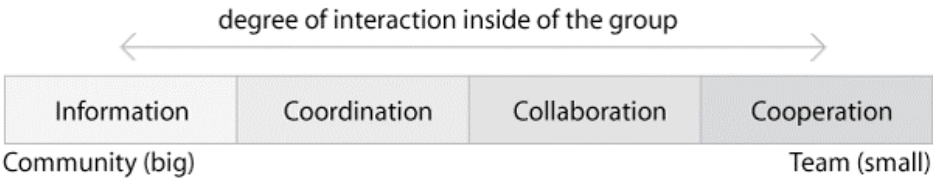


Image 8: the influence of size of the group on the degree of interaction

The supported group size is an interesting information for the user we therefore ***include*** it into the characteristics table. Beyond some exceptions, systems built for small groups are often not usable for larger ones. This also holds the other way around. This is a good reason to place this factor into the classification table. As there is a continuous scale between a team and a community, which is not possible to describe in the table, we decided (similar to the 3C factor) to place only the nearest extreme of the scale: e.g. an internal project management tool which does not support more than 10 users is considered as team groupware.

3.2.19 Guidance dependency

The Guidance dependency factor describes how involved the user has to be to take advantages of using of groupware for the group. The factor is a continuous scale from **guidance dependent** to the opposite **guidance independent** side. A good example of guidance dependent would be using „copy-paste" to input the information from a email client into the groupware system. On

the other side the mail client could be fully integrated in the groupware and filter these information automatically – so the user wouldn't even notice that he is working with two different systems. Such a system would be guidance independent.

The guidance dependency factor is almost similar to the user involvement criterion factor. Even more in some cases (e.g. ICQ) there is no need to speak about guidance dependency because the user cannot be less involved in using the system – he is always aware that he is using a messaging system to communicate. This also does not affect the advantages brought by the use of the system. As this is not possible to make a decision for every system and even if it would be, that information would be covered by the user involvement criterion, we decided ***not to include*** this classification factor.

3.3 A list of functions provided by groupware

As described in part 3.2.5, instead of defining functional groups (e.g. messaging systems) we list here all the functionality that can be included in groupware. By this decision we can give more information to the user about each of the groupware categories and define more accurately the boundaries between them.

- **synchronous messaging**
  Chat systems (IRC), instant messaging (ICQ, MSN messenger).
- **asynchronous messaging**
  The most known electronic type of communication – email, mailing lists, newsgroups.
- **message boards and forums**
  Allows posting of messages in more than one possible threads.
- **workflow management**
  Tools for maintaining workflow. A good example is routing documents through an organization. This functionality assures that a concrete document (a change request for a software project) is written by person A and is passed to person B, who takes care of filling out the request.
- **project management**
  This functionality includes creating projects, tasks, workers, assigning tasks to workers, viewing project status, quality, completeness, todo lists and more.
- **collaborative writing**
  The most known form of collaborative writing is certainly „wiki". However, there are

more such systems. Some of them support even synchronous collaborative writing e.g. SubethaEdit for MAC.

- **collaborative drawing - shared whiteboards**
  Allows multiple users view and draw on a shared drawing surface. Shared whiteboards can indicate each users activity by coloring his/her objects by his/her associated color.
- **audio communication**
  Allows two-users or multi-user voice communication (as regular telephone).
- **video communication**
  Enhancing the audio communication with the possibility of viewing each other.
- **decision support**
  This functionality provides voting, brainstorming, putting weight and probabilities on ideas and events.
- **document/file management (maintenance)**
  Allows sharing of documents, managing their accessibility to different classes of users, archiving and versioning of documents.
- **resource management**
  Allows booking and managing resources like cars, rooms, technical equipment or even employee's time.
- **address-book / contacts management**
  Maintaining contacts of people including functions like duplicating a contact, removing a contact or sending a contact to other user.
- **calendar**
  Maintaining events, sharing and synchronizing of calendar events.
- **todo list**
  Maintaining todo lists of group members.

3.4 Other (non groupware specific) factors

These factors are the few of most important factors for the choice of a groupware system, groupware deployment and its usage. Even if these are not groupware specific factors, not using these factors in the classification would cause its uselessness for the user.

3.4.1 The platform criterion

- **Operating system based**
  - **Dependent**: The groupware system is build for a single OS, which does not allow users with other operating systems to take part on the collaboration.
  - **Independent (Multi-platform)**: The collaboration occurs on multiple platforms. Either there are binaries for more than just one OS, or a multiplatform supported language (e.g. JAVA) is chosen.
- **Browser based platform**

  Here, the system is implemented as a web browser application, which does broadens the range of compatibility, as almost any operating system is capable of running a web browser. Although, not all web applications are compatible with all web browsers, which causes similar problems as in the above mentioned category. Therefore we distinguish between:
  - **Browser dependent** – The application designed to run in a concrete browser.
  - **Browser independent** – Applications don't use browser specific functions or are capable to run in the majority of browsers.
- **Small displays enabled (mobile groupware)** In [21] another category can be found, namely *Mobile Platforms*. In our opinion, in these days, all mobile platforms can be seen as other operating systems (or different web browsers). However it is important to know if there is a special build for such small screen devices. Therefore we'll divide systems also to these categories: **small screen capable** and to **small screen incapable**.

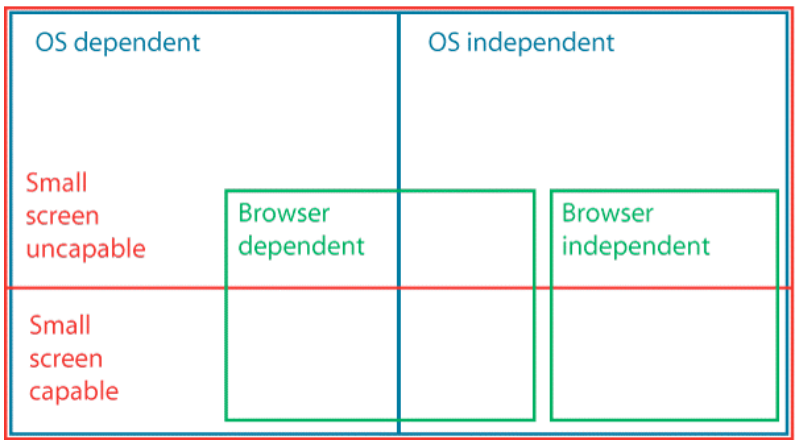


Image 9: Platform dependency diagram

3.4.2 Interoperability

It describes the level of interoperability of the system. This factor will be assigned intuitively to one of the following levels:

a) **Maximum level** – all functions of the groupware are implemented according to an accepted standard or they are compatible with the majority of similar groupware functions in other systems.

b) **Medium level** – majority of the functions are implemented according to an accepted standard or are compatible with the majority of similar groupware functions in other systems.

c) **Low level** – the interoperability is considered only in few or none of the systems functionality.

3.4.3 Client or server platform

This criterion describes whether the platform of the client and/or server is

a) **free**, and

b) **commercial**.

We would like to remark that free groupware is different to „open source“ groupware. Open source groupware will be discussed in Extendibility (chapter 3.4.5).

3.4.4 Backup

By this criterion we assure that the system has a backup functionality. We consider three cases:

a) **automatic backup** functionality with archiving (versioning),

b) **manual backup**,

c) **no backup** functionality.

3.4.5 Extendibility

The extendibility of the system is another important criterion. We divide this criterion to these cases:

a) **free extensions** – there are free extensions available for the groupware application.

b) **open source** – it is possible (and by license permitted) to implement extensions on your own.

c) **commercial extensions** – all extensions must be made by a certain licensed authority.

d) **no extension possibility** – this type of extendibility is the case in large systems, where the developer or distributor does not offer the possibility of extensions and/or the implementation of extensions is technically not possible.

4 Classification of groupware

Groupware is a very broad class of applications, which is linked together in the scope and purpose: facilitate group work.

In the last chapter we have discussed factors according which groupware can be classified in some categories. The goal of this chapter is to form concrete categories of groupware according to the classification factors that were listed and described in chapter 3. Including all the factors that were chosen to be included in chapter 3 into the classification table is necessary to give the user an overall view of the concrete application.

Though facing the question, which of these factors should form the boundaries of the groupware categories we have come to an assumption that, the major factor for the choice of a groupware is its functionality. Users first look what the software offers, and only after an application matches this criterion they look further on other, far not less important aspects, like interoperability, OS dependency or awareness. This implies that for the ease of use of these categories, they should be formed only from functional criteria. If then someone needs a concrete system he looks at all systems in a particular category and filters out those, which also fit other requirements, e.g. are OS independent.

On the contrary the categories of functionality that can be found in [6] or [29] are not concrete enough for deciding what are the software capabilities. For example *messaging systems* does not tell whether is it synchronous or asynchronous messaging which is for the user the main question. We therefore decided to form alike categories which though are more concrete and give the user the sense what to expect from the application.

Every category's definition will have the following structure: category name (in bold font), category description (normal font) and category definition (italic font). The category definitions will be boolean functions of predicates in the form described below. For every functionality listed in chapter 3.3 we choose whether:

- the system has to support this functionality *definition = (FUNCTIONALLITY: yes),*
- this functionality is forbidden - *definition = (FUNCTIONALLITY: no),*
- it is recommended, but not necessary, to support this functionality *definition = (FUNCTIONALLITY: recommended),*

- or it is independent according to this functionality *definition = (FUNCTIONALLITY: independent).*

We would like to remark that the difference between *recommended* and *independent* is that when we use *recommended* we would like to emphasize on this functionality because it is a useful functionality for systems especially in the particular category. We would like to mention also that for a clear structure of the definitions, instead of listing all independent functionalities e.g. (… *(forum: independent) AND (calendar: independent)* …) we rather group them and substitute with the word *other* e.g. (*AND (other: independent)*).

We formed these categories:

- **asynchronous messaging systems**

Groupware systems which fit in this category has to be able to send asynchronous messages (not depending on whether they send synchronous messages as well or not).

definition = (asynch.messaging: yes) AND (other: independent)

- **synchronous messaging systems**

The systems in this category must be designed to send synchronous messages between users (not depending on whether they can or not send asynchronous messages as well). Systems that are able to simulate synchronous messages through frequent sending of asynchronous messages does not fit in this category.

definition = (synch.messaging: yes) AND (other: independent)

- **audio communication systems**

Audio communication systems have to have the possibility of synchronous audio communication within two or more users. Systems which does not allow just audio communication without video are not to be classified in this category.

definition = (audio communication: yes) AND (other: independent)

- **video communication systems**

Systems in this category offer video communication features, but can in addition support also pure audio communication.

definition = (video communication: yes) AND (other: independent)

- **productivity tools** (calendar and todo list)

The productivity tools category includes systems with the functionality of managing his/her personal calendar and todo list and sharing it to others. The application however could provide more sophisticated sharing/exchange of calendar events, searching for free/busy times of other users or other types of queries.

For the definition of productivity tools we extend the definition pattern with one of the groupware independent classification factors. The predicate itself will define the level of this factor.

definition = (calendar: yes) AND (ToDo list: yes) AND (contacts: recommended) AND (interoperability: medium or higher)

- **conference systems**

Conference systems must support one or more of the basic communication features (audio communication, video communication or sych/asynch messaging) for the minimum of three users. In addiction they have to support at least one *other conference enhancement* such as list of participants, conference moderator function or communication report.

definition = [(audio comm: yes) OR (video comm: yes) OR (synch. messaging: yes) OR (asynch. messaging: yes)] AND (collaborative writing: recommended) AND (collaborative drawing: recommended) AND (document management: recommended) AND (other conference enhancement: one or more)

- **workflow systems**

This category includes systems that are designed to support workflow in organizations. They though need to have implemented at least one function which is meant to support workflow e.g. document or message routing through an organization.

definition = (workflow management: yes) AND (other: independent)

- **project management systems**

Project management groupware systems have to give the users the possibility of defining projects, assigning employees to projects, tasks to projects, tasks to employees in order to facilitate and manage the project process. It has to have the function of tracking what projects and tasks are open and which closed. All other features like further division of projects to phases, employee time reports and statistics, management of customers etc. are considered as additional.

definition = (project management: yes) AND (other: independent)

- **resource management**

This category groups systems with the simple functionality of managing resources – allocating free resources for particular actions/persons/projects and viewing the status and calendar of all resources.

definition = (resource management: yes) AND (other: independent)

- **co-authoring systems**

Systems in this category share the scope of allowing participants to collaborate on a document (written or a picture). Typical representatives are shared whiteboard, collaborative writing systems such as WiKi or SubEthaEdit.

definition = [(collaborative writing: yes) OR (collaborative drawing: yes)] AND (other: independent)

5 Groupware design issues

M. Mandviwalla and L. Olfman in 1994 in the article „What Do Groups need?“ [30] state that *Researchers are beginning to question the design of current groupware systems in empirical and conceptual work. An analysis of these limitations shows that groupware systems do not fully match the work life of groups in organizations.* Technology went further but still this situation has not been radically changed. Instead of focusing on satisfying user requirements, in many cases, the effort of developers lies on implementing challenging technologies. In groupware development companies, this is also from the marketing point of view a great message to broadcast. It is easier to promote a concrete feature than to talk about improving implementations of some user (group) requirements – which in reality might be more difficult, and does not show such strong effect in marketing. A basic list of groupware success criteria is pointed out in [35].

However in most articles as the major problem in deploying groupware is the acceptance of the users. The acceptance is influenced by many aspects mainly interoperability and interface design (including awareness of other users).

In this chapter we will list some problems and questions in developing groupware. They are not explained deeply enough to use it as a complete source of information. It should be considered as a checklist of problems, which have not to be omitted before starting any groupware development. The list items are structured according following categories:

- technological aspects
- social aspects (understanding of groups and group’s requirements)
- management aspects (groupware deployment problems)

5.1 Technological aspects

5.1.1 Architecture

The taxonomy coined out by Petterson in [37] was one of the first architectural models for synchronous groupware. It introduced three models: centralized, replicated and hybrid. Later additional models were invented e.g. by Roth and Unger in [22]. Some of these models are described in chapter 3.2.4. In this chapter we will state some issues which has to be considered

before choosing the architecture for future groupware system. Another sources of information about groupware's architecture are [6], [17], [28] and [40].

Even though groupware is a kind of a distributed system, the view on some details might diverse or even be totally opposite. For example while other synchronous systems use point-to-point communication on the data transport layer, groupware needs point-to-multipoint or even multipoint-to-multipoint mode. Although it is possible to simulate multipoint communication through more point-to-point messages, this causes errorness and slowness of the system.

Another fact is that distributed operating systems and middleware platforms are aimed to mask out the presence and location of other users and giving the user the feeling being a single user of the system (referred as transparency in distributed systems). This may cause difficultness of using them for groupware development, because presence, awareness and collaborative consistency might by hard to implement.

However, here are few systems which support multipoint communication and consistency management of replicated data and were even used as basis for groupware platforms. Some of these systems as well as groupware development platforms are discussed in [6] on page 40.

According to Greenberg and Marwood: *There is no real answer to whether a centralized or replicated scheme works best for groupware. Rather, it is a set of trade-offs that revolve around the way they handle latency, ease of program installation and connection, programming complexity, synchronization requirements, processor speed, number of the participants expected, communication capacity and cost, and so on* [40].

5.1.2 Interoperability

Groupware systems need to be interoperable because it is unlikely that a single system will ever be able to satisfy all requirements, whereas multiple self-contained systems will bring too many usage constrains [30]. In addiction Henri Ter Hofte [6] states that *within a single project or cooperative task, people frequently make transitions between various forms of cooperative work* namely transitions between individual and collaborative work or synchronous vs. asynchronous collaboration. So to complete a project or task, different groupware systems are needed.

Bullen and Bennett organized a large study involving 223 people from 25 organizations, where each of them used a subset of eight groupware applications. The outcome was the fact that the productive use of groupware is hindered by the lack of interoperability and that most groupware are unaware of the existence of other systems. Thus users have to switch between different

applications which includes problems like logging into more applications simultaneously to start the collaborative work, copying data between applications and finally move the results between these applications. As described later in the social aspects, unless high interoperability is supported by the system, users reject using more not inter-connected systems and tend to negate the whole groupware deployment.

A great example of failure caused because of interoperability lack is described in [2]. AT&T and MCI introduced videophones commercially. These systems were not able to cooperate. This caused that if somebody wanted to buy a videophone, he had to be sure that everyone has the same system. So the customers waited until a clear standard has been resolved.

It is important to mention that interoperability is not only needed between groupware systems themselves. It is needed as well in-between groupware systems and single-user applications, co that the activity can be coordinated and the data in the systems is consistent.

Although groupware interoperability seems to be a very important issue which must not be obeyed, it is not desirable implementing only functions that are interoperable and bypass all other functions. This would lead into lack of functionality which is surely more important than the systems interoperability.

Here we would like to refer the reader to chapter 6, where actual groupware standards and the history of them is discussed.

5.1.3 Presence and awareness

As one of the most important features of groupware is to support collaboration with different place and different time, it is very important to let the participants know about the actions and state of other users as well as of the state of shared objects. Another reason for considering awareness is that a group can also benefit from implicit communication such as information about the users environment, indirect gestures or other information. Maintaining such awareness should be incorporated in every groupware system. Users should be informed by the groupware even if they are sitting in the same room at the same time. Groupware systems do this *by providing users with feedthrough information* [6] – the information about other user‘s actions. Hofte in [6] identifies these problems in implementing awareness into groupware:

- Estimating the level of presence and awareness – which feedthrough must the system provide on which actions of which users?

- Estimation the level of extent to which the users can control their presence (if they have e.g. control which of their actions cause feedthrough to other users)
- Estimation the level of extent to which the users can control their awareness – which actions of other users cause feedthrough to them?

Awareness information could be for example the date and time when a message was sent. Such an information can avoid conflicts – if you see that the message was sent in the middle of the night you can expect errors in the message and treat the information differently to a message send during standard work time.

5.1.4 Concurrency conflict management

Concurrency conflict management is sometimes also referred as collaborative consistency management. Groupware systems with shared data face often the problem with the concurrent user's activity and different representations of the data on different screens. One of the problems faced by groupware developers is that, due to unpredictable amount of time that is needed to send packets across the network and other factors even serial actions can be received in a different order that expected. This can then cause inconsistency of the data. A concrete example of such actions which cause inconsistency is illustrated in [6] on page 35.

As everybody knows, this is not a problem only in groupware systems. Concurrency solving policies also play an important role also in other networked and database systems. The typical solution for this problem is to allow only one user at a time to modify a concrete object. Such locks though are in groupware systems not exactly applicable since groupware's goal is to support (also synchronous) interaction – which is not allowed in locking. This difference causes that in groupware development, issues involved in managing consistency are different from those in typical database systems. Another reason for treating groupware differently is according to Greensberg and Marwood [40] the fact that *it includes not only computers but people as well*.

Concurrency control involves researchers from distributed systems as well as from the CSCW community. Although it seems to be a pure technical problem there are concerns about other questions to be solved that have different manner. The social part of the problem is the user interface, which must be able of showing possible concurrency conflicts in a way people understand it and in this way avoid them. Such awareness is explained in [39] where e.g. a icon is drown when an object is being modified – it is being referred as ‚graphic-echo'. However in relaxed WYSIWIS, where participants can have a different view, this might not be enough.

Beaudouin and Karsenty solve this using ‚audio-echo' – playing a sound when a object outside of the view is being modified.

Although groupware concurrency control is a frequently discussed issue in literature, system developers often ignore it or *consider it to be an issue to be remedied by some textbook approach* [40]. We therefore remind that synchronous groupware without an appropriate concurrency control policy are unacceptable.

To state an example of a conflict that can rise due to concurrent actions consider two sides of a communication. Firstly we assume that t_i are timestamps and $t_1 < t_2 < t_3 < t_4$. Site 1 transmits action A at time t_1. After that, in time t_2, Site 2 transmits action B. This process is illustrated on image 10.

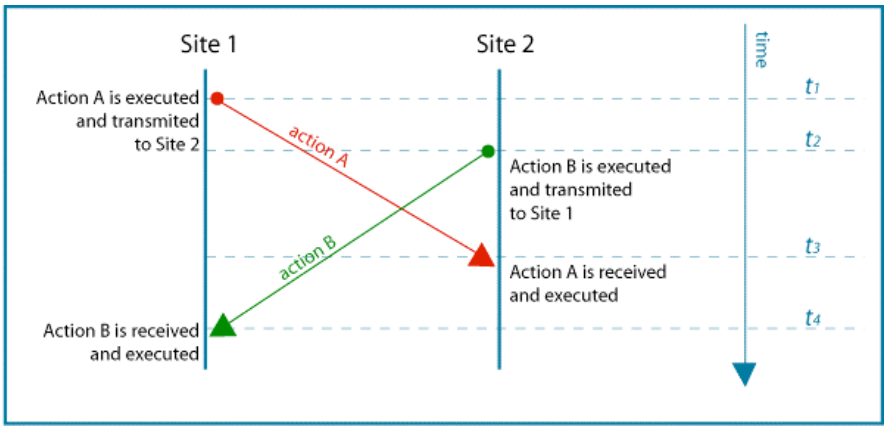


Image 10: example of communication between Site 1 and Site 2 which might cause a conflict

Inconsistency arises when actions A and B are not commutative.

Basic types of concurrency solving policies are introduced in chapter 3.2.16. We will now illustrate all these approaches on the previous example.

- In the case of *non-optimistic serialization*, the transition of action B had to wait until action A arrives and is executed. This causes in real systems latency in response time of the system. This causes frequent user rejection and is therefore for many systems unacceptable.
- *Optimistic serialization* assumes that conflicting actions are not often and therefore allow inconsistency to appear. The following mechanisms of establishing a consistent state can however be tricky. One approach is to roll back all actions that should be executed after the conflicting action and then redo them in the correct order. In our example after receiving A at Site 2, action B is rolled back and executed after the

execution of action A. A different approach is to apply a transformation which transforms the state (in Side 2) so as these actions were executed in correct order. If our example pushes objects in a stack, only the last two objects are swapped. This is certainly more efficient, but might be non-trivial to implement and requires consideration of each problem separately, whereas undo and redo does not require special consideration.

- We assume that the reader is familiar with the *locking* approach and will therefore not discuss it in more detail. The only consideration that we want to point out is that in *fully-optimistic locking* if the users performs more than one action with a tentative lock and the lock is finally denied, it might be difficult to rollback to the state where the first conflicting action was performed.

In general while non-optimistic approaches are acceptable in standard computer systems such as database systems. Groupware is different as it includes not only computers but also people. Therefore groupware systems must often choose other policies because dealing with the latency that is implied by the waiting for one to perform its action might be unacceptable for users.

Another difference is that people can be more or less tolerant to inconsistency. This leads into another policy – ignoring the possible conflicts. This is driven by the idea that allowing inconsistency to rise might be more acceptable than trying to prevent it – and in this ambition restrain important functionality of the system. An example is a bitmap-shared whiteboard where two users concurrently draw a picture. The difference of few pixels (in the intersection of the lines of more users) is of course acceptable especially if it is only a sketch.

An important issue is here the granularity. Imagine a shared whiteboard where users can lock a selection of the image. Here considering granularity is very important especially in the respect to locking. Coarse granularity means less lock requests for the system but also lower the possibility of collaboration (if the user locks the whole image, no collaboration is possible). On the contrary fine granularity gives the users more possibilities for concurrent actions but requires more often locking of smaller objects by the system (choosing small selections for drawing might be restrictive for users). Thus the choice of granularity requires great intuition from the designer.

An encouragement for this policy is that people naturally follow social protocols and if they see what others are working on, they will not naturally perform conflicting actions. This method of reducing conflicts is *awareness* of other user's actions (described in 5.1.3). In this case a object

that is being modified by a user is grayed out on the other users screens. These methods can reduce conflicts and may override the need of conflict management. Sometimes inconsistency can be needed – e.g. when two users what to have a different view on a shared document – they split their actions and then decide which version is better. Awareness is also important in showing what action is being performed by the system. In case of a rollback and redo in an optimistic serialization scheme, this has to be presented to the user clearly to minimize the user's confusion.

Finally in inconsistency prevention developers have more options to choose. They can deny the conflicting action, or they can postpone it until it is safe to process this action without the rise of a conflict. When developers, even though these considerations, choose locking of objects, it might be important to the user, that was denied access to the object, to know who has the lock for that particular object. This might help to communicate between the users and finally free the object by the user with the lock.

A slight alternative view on consistency management can be found in [4].

5.1.5 Late-coming

The joining of a new user to a running collaboration might be a development challenge too. While in some cases there is a trivial solution for this problem: in a centralized architecture, where the state of collaboration is completely stored on the server, this only needs to download it from the server. On the contrary in replicated of hybrid architectures the state is distributed across all users. Therefore for establishing the state for the new user a archive of actions has to be restored, or a copy of a particular user has to be done. This though involves interrupting the other users for example by stopping of the system to establish consistency after the join of the new user.

5.1.6 Distributed file systems

In the majority of distributed file systems only one user has the privilege to access or change a file at a time – the file systems were not designed considering collaborative consistency management. There is another problem in distributed file systems is according to Hofte: *many distributed file systems were not designed with presence and awareness in mind: they typically do not provide update notifications and do not allow for modification of access rights (e.g., from read to write access), without closing and re-opening the file, which hampers the*

construction of groupware on top of these file systems [6]. More details about shortcoming in distributed file systems can be read in [6].

5.1.7 User interface design

Group development and behavior transform groupware interface design into a social issue because, on contrary to standard user interfaces where the second side of the communication is a machine, here people interact with people. The major difference is that every concrete person can behave in a different way (even one person can behave differently in two similar situations) whereas the machine behaves equally (because a machine is programmed in a certain way and can not change its behavior in time unproductively).

Multi-user interfaces face different requirements compared to standard user interfaces. Current user interface systems such as Microsoft Windows or Mac OS X were not designed having multi-user interfaces in mind and therefore does not support presence and awareness – e.g. multi-user scrollbar which let's you know about other user's activity. Another example of lack of support of awareness is when e.g. a user performs an action – choosing a function from the menu; the whole application is paralyzed until the action finishes to avoid inconsistency.

5.1.8 Testing difficulty

In groupware, especially in systems designed for communities, testing is sometimes more difficult than in single-user systems. Some of the reasons are listed below:

- especially in asynchronous groupware many tests take long time to complete
- prototyping is more complex and modifying prototypes can be technically difficult
- organizing tests involving more that few participants requires greater scheduling and might be according to the organization impossible
- group interaction is difficult to predict and finding problems in replicated systems can require some kind of recording of every participant's actions. The analysis of their work can be non-trivial.
- The groups formed for testing may not satisfy every group needs
- groups are dynamic, especially new groups change very quickly their behavior

More details about testing difficulties can be found in [2].

5.2 Social aspects

Social aspects are another, but far not less important, factors that influence the development of groupware systems. We divide them in this thesis into problems caused by insufficiency of understanding of groups, problems in gathering group requirements and management aspects.

5.2.1 Problems caused by insufficient understanding of groups

In this chapter we will list some of the most common groupware problems which are caused by not paying adequate accent on what groups really need. These deficiencies can be overcomed by taking a broader view on CSCW, especially the social view, when designing a concrete groupware system.

5.2.1.1 Only group interaction support

This problem rises when the system supports only the interactive part of the work and dismisses the fact that this is not the only benefit for the group which can be brought by groupware. Groups can also benefit from individual functions (functions which are designed only for single user use) such as note taking or personal drawing. E.g. a participant of the interaction first draws a picture locally and then decides if he wants to publish it or not. This can bring more cleanness and higher effectiveness of the collaboration.

5.2.1.2 Single user perspective

Analytics and designers usually design systems according to the gathered requirements of some users and try to imagine the „average user“ of the system. Usually they communicate with the manager or team leader – which himself can have biased view on the group. It is important to include all group member types into the requirements collecting phase. Even though, the requirements change from person to person even on the same group function. This implies that all potential system users should be involved. This is definitely not possible. The difficult part is hence to set the boundary – whom to include and whom not.

5.2.1.3 Simplified view of groups

Some groupware systems focus only on the positive aspects of group work neglecting that there can be friction, competition or other human factors within the group which can differ from group to group. Other have an egalitarian approach toward the group. They ignore different

roles; ignore the power and interest difference ness of various group members. Another problem caused by simplifying the view is when groupware is designed to *„reduce process losses“* (to reinforce the position and influence of the process - in our terminology a strongly restrictive groupware) and pushes the collaboration into a *„stage based sequential path“*. Some agree with the process importance, some question whether the process loss is not a necessary part of group work which carries to more creativeness. This aspect is discussed in more detail in [30].

5.2.1.4 Only one time-factor support

There are some systems which support only synchronous or only asynchronous communication. As stated in [39] *it does not seem reasonable to expect a group to use separate systems for brainstorming in face-to-face and distributed meetings while receiving related email and documents on another system.* Using more that one groupware system leads to increasing of time spent interrogating between the systems. Such (according to chapter 3 - high user involvement) systems causes frequent user resistance and finally not using one or more systems correctly. This may lead into a complete systems deployment failure.

5.2.1.5 Implicit perspective worldview in design

Design work involves the use (implicit as well as explicit) of a guiding design principle, worldview or paradigm. The worldview of current groupware systems is not dependent on the possible system users. E.g. one worldview is when in a conference system the moderator decides what topic will be discussed and allows the meeting to change its flow. A different one would be when the leader consequently chooses the topics and precise flow of the communication. Worldview is less important in single-user systems, because and individual can switch to another system anytime, whereas a group *does not have this luxury because everybody needs to use the same product* [30] and can not change it without the permission of all group members (and possible another higher authority). Mandviwalla in [30] also stats that: *a unified groupware worldview is unlikely*.

5.2.2 Problems in gathering group requirements

It is being strongly emphasized to focus on user needs in the modern software development process. If this rule is applied to groupware development, this leads in groupware development into the creation of an average group that usually reflects the designers personal intuition, experiences and expectations. This might not always be correct as each group is a unique

combination of its constituents, environment and task. Another difficulty in gathering group requirements is that on addiction to standard users, groups are more dynamic and change with time more frequently. Seeing this, groupware development process is surely different to the standard software engineering process.

Every organization has its own environment which leads into different importance of groupware requirements. Some organizations need group development support (defined in chapter 5.2.2.3) but are not in need of sustaining different behavioral characteristics whereas other organizations need the exact opposite.

However to help gathering groupware requirements, Mandviwalla and Olfman in [30] present a *list of generic groupware design requirements.* They were synthesized from a survey of work group literature. These requirements should in no way replace the requirements gathering phase in any project. They should serve as a base for starting of this phase and provide the user and designer with the information what is possible and what isn't. Some of them even don't meet the term requirement. They are *just generic issues that a groupware designer should consider when designing a system.* Moreover it is very important to note that a single product may never be able to meet every requirement.

These generic requirements make the following assumptions:

- *The purpose of groupware is to support the collaborative activities of organizational work groups.*
- *The requirements of specific work groups should be derived as a subset of the generic requirements by applying an appropriate systems development methodology (e.g., user-centered design).*
- *The groupware will focus on supporting the lifeworld of managerial and professional office workers. According to Panko [1992 b], managers and professionals make up three quarters of the office work force and spend about 70 percent of their time communicating.*
- *The groupware will be used by several, if not all, professional work groups in an organization. Therefore, the requirements are oriented toward developing general-purpose organization-wide groupware for professional work groups.*

5.2.2.1 Support multiple group tasks

Everyone has to agree that groups exist to perform tasks. So the most important requirement is to support the tasks that a group have to solve. In this light groupware systems should support multiple tasks because the group's tasks may be influenced by many factors and changed in time. Researchers have developed generalized taxonomies of tasks and processes. An example is McGrath who proposes four *basic task processes: generating, choosing, resolving and executing.* One solution for supporting multiple tasks is implementing different modules – each one for a different task process. A full support of multiple group tasks would need a massive expansion of functionality of the concrete product and interoperation of more products from different vendors.

5.2.2.2 Support multiple work methods

Each task can be divided into smaller sub-tasks and each of them has to be completed. According to the type of the task, different tasks require different methods of task solving. A groupware system should support all these methods. For example a „generating" task may require different task-specific media, communication media, tools and techniques.

A planning group may wish to record their ideas in a shared document. In this case the simple text media can rise interesting questions and design challenges: linear or non-linear text, list / table / unstructured text, does it need different encoding for different users?

Another problem is that these work methods are not completely predictable. An example can be a choosing task e.g. budget selection, where more users has to agree on a specified budget according to facts. Some of the users wish to calculate the budget according to the last (or similar) budget. Others may call their stuff for assistance and other may calculate the budget through social interaction. Studies have found that collaboration is a specific mixture of human interaction and solitary work. This changes the method of task solving because some users prefer work in isolation and then merge their results, while, as others prefer to work totally public without any private part.

All these different work methods have to be supported by the groupware system. As stated above, the problem is that these work methods are by the time of design of the system for every concrete user unpredictable. Thus, this causes a non-trivial problem for the designers.

5.2.2.3 Support the development of the group

In each group, there are tendencies to some change. These changes can be affected by the environment and internal dynamics. This is to be referred as *the Group Development Process.* Some examples of such a change are e.g. the disputes about details disappear as a deadline nears and the pressure rises. Or boundaries change as the organizational relationships are formed.

There are at least two areas in which groupware can support group development, namely:

- **influencing behavioral processes** – This means the use of techniques that affect increase of interaction, redistribution of power, defining roles and increase consensus. An example is a leader of a group, who brings a consensus building tool to help resolving disagreements.
- **managing the mechanical aspects of the development process** – This area includes *group administration* – starting a group, adding, editing and removing group members, schedule meetings, manage rights, and *group memory* – the archive of past group activities. This archive may help to learn from past mistakes, analyze the style of the collaboration and diffuse conflicts.

5.2.2.4 Provide interchangeable interaction methods

Office work is a mixture of same-time and different-time as well as same-place and different-place mixture. In groupware literature this is referred as any-time any-place operation. This means that users may need e.g. a video conferencing system to discuss important issues more precisely but also are in need of a asynchronous messaging system – e.g. email to coordinate the particular video-conference meeting.

Another issue is that collaboration is a mixture of interaction and solitary work (also discussed in 5.2.2.2). Older groups (which have more shared experiences and compatible norms) will use higher level of interaction compared to recently formed groups, where the members have to get used to each other. This reason for should the level of interaction be adjustable with time – to each interaction method. An example could be a messaging system which can be adjustable from asynchronous messaging to synchronous. In the first phases of projects the majority of work is done solitary and the results are coordinated once in a time (asynchronously). As the deadline nears, these parts have to be combined together. This reason for the communication frequency rises and the type changes to synchronous.

Developing groupware that supports multiple interacting methods, which can change with time can be viewed as a interesting technical challenge. A conceptual challenge is the notation of such interchangeable interaction methods in a standard understandable way.

5.2.2.5 Sustain multiple behavioral characteristics

Groups manifest in many behavioral characteristics *while completing the tasks* according to 5.2.2.1 and 5.2.2.2, *during their development* 5.2.2.3 and *as a part of interaction* according to 5.2.2.4. Some of these characteristics are stress, relationship with the organization, commitment and cohesion. These characteristics can influence the use of groupware. For example, whether or not busy managers use to report all their tasks every day (which costs their time) depend on the commitment to the particular task. Groupware systems should resist as many characteristics as possible to sustain the social dimensions of the collaborative environment. These requirements are hard to satisfy because there are many behavioral characteristics and it is hard to evaluate their relative importance.

5.2.2.6 Accommodate permeable group boundaries

Group boundaries are the physical, spatial and temporal divisions that differentiate groups. The most important attribute of the boundary is its permeability. Under permeability a degree of throughput should be understood – the degree which defines how much and how easy it is to enter and exit the group. Permeability is affect by social and economic factors such as relationship with other groups, the presence of an authority and the group's environment. Groups interact within their environment so groupware should be able to set and manage each group's permeability and its boundaries.

The group boundary is defined by group's identity, which is defined by group memory and group development – see chapter 5.2.2.3. Groupware designers must be able to control communication and data exchange with other systems to implement boundary management features. This implies that (in inter-organizational settings and different groupware systems) without solving the interoperability of the system it is impossible to implement the boundary management.

5.2.2.7 Adjustable to the group's context

The context in which the group works influences many of the above listed requirements. Computer experience influences work methods, composition influences behavior and culture

influences preferred interaction methods. Greenberg proposes that groupware should be personalizable to individual needs as well as the whole group needs.

Groups are tied to objects, place and time. If these are changed, great changes to the group's needs can rise. For example the system identifies these roles: editor and rewriter. But in time these can convert to new hybrid roles. This only emphasizes that *we can never fully predict who will, in what time, in what way, and for which tasks, use a particular groupware feature.* Such a groupware is referred as context-aware groupware. In most cases groups have ideas what they need. Using this knowledge can help to customize a particular groupware correctly. So if this context is not seen as a hindrance but as an opportunity, flexible and adjustable groupware is easier to design.

Another difficulty is to implement such a customization that gives each user the possibility to customize. Many systems offer the possibility of changing background color or font size. There are though other features that are a way more difficult to implement than these. Attributes that affect the whole group like the interaction method, or the group permeability can be only adjusted by developers of the system – not by end-users. But high customization has also their disadvantages. Group members are not always able to identify exactly their needs. This may cause that the user selects everything that is possible to choose.

As different users have different points of view, options that affect the whole group can cause disagreements within the group. Mandviwalla and Olfman think that this *conflict is not necessarily a negative consequence – from the socio-technical perspective this „conflict" may be viewed as a process of jointly optimizing the technology and the social goals of the group.* The last problem identified in this manner is that people not always want to take this advantage and customize the groupware. It is caused not only by the fear to damage something but also by their passiveness.

All these groupware problems listed under 5.2.1 and 5.2.2 certainly interrelate. This makes it difficult to consider one without having an eye on the others. In addition importance of all these problems can vary with the group. As example, consider a company with a strong organizational culture. The worldview is definitely different to a small company with egalitarian employees policy (e.g. an extreme programming team). Mandviwalla thinks that these problems are caused by the fact that groupware systems do not fully reflect the work life of groups in organizations.

To overcome problems listed above, groupware research must focus more on work life in organizational groups. Such a deep and comprehensive research would be very helpful for any groupware designer. To learn more about these social aspects we refer to the article „what groups really need“ [30] by Mandviwalla and Olfman.

5.2.3 Privacy and abuse problems

In some groupware systems anonymity is needed to fulfill the main functionality - e.g. voting systems where anonymity can be crucial to encourage users to discuss and protect them from harassment. However anonymity can be tricky and can cause violation of social protocol such as violating privacy, sabotaging the group collaboration and other inappropriate advantages of anonymity.

There are more such problems to be mentioned, for example in systems which provide private information about other users – these can be misused or stolen. Another problems of abuse include also spamming.

5.2.4 Communication structure

Every communication has its predefined or often used structure – e.g. after receiving an email from the client, the responsible worker sends a acceptance response and sends a request to the particular department to satisfy the clients needs. If this structure of the communication is supported by the system, it can speed up and improve the efficiency of the communication.

There are more types of communication structure. *Technologically-mediated communication* is called when the system exactly determines the structure of the communication and does not allow different use. The opposite is *socially-mediated communication* where such a structure is defined automatically by the communication itself – e.g. a person sends an (blank) email message to another person. The choice is left on the receiver whether to respond and start a communication or not. This type of communication is more time-consuming and causes more errors than a communication with predefined structure. Thus can not be used for some types of organizations such as the military or finance sector. On the other hand technically-mediated communication – with its restrains - may lead in creative organizations (advertising or arts) into the rejection of the particular groupware.

5.2.5 Groupware flexibility and tailor-ability

Groupware system is considered flexible when only few or no assumptions are made about the environment in which it is going to be used. Flexible groupware is applicable to a broader target group – thus can produce better return of investments. However many groupware systems lack such characteristic. Most of inflexible groupware systems are restrictive, because they are designed for a concrete type of group and its environment and any stepping outside the patterns that were implemented is unavailable.

Flexible groupware has also its constrains. Using flexible groupware is surely less comfortable than a groupware system exactly designed for a concrete group. Though flexible groupware can be tailored to fit a particular group work situation. Tailoring is done after deploying groupware in its final environment. Though it is needed to include the possibility of tailoring in the early phases of groupware design. Tailoring can be performed by the users themselves, a higher authority (manager) or the system itself by learning the way it is being used and changing its behavior for a better support of them. Tailoring can be implemented in many levels from setting up each users personal keyboard shortcuts to altering the functionality of the system. Further tailoring of component based systems has various ways in which tailoring can be achieved: tailoring by configuring components, tailoring by changing (or extending) the implementation of a component and tailoring by changing (or extending) the composition of components. Detailed description of the above listed types is written in [6] on page 50.

5.3 Management aspects

5.3.1 Acceptance of groupware systems

All software face in this light the same problem. Unless a critical mass of users uses the concrete software it can not be successful. However, groupware acceptance is harder to achieve than in other systems because even one single person can discard the whole adoption process: Consider a group calendar where everybody is forced to manage his free/busy time and meetings are scheduled automatically. If one member of the group lefts its calendar clear, every meeting is accepted - this might thus be not true since he has meetings but he has not submitted them to the calendar. Even if all users are forced to use the system, problems with acceptance caused by natural resistance to change, rejection of bureaucracy and personal opinions can appear. Personal opinions can be negative because of difficult user interface and general use of

the system, which is not foreign to groupware considering that groupware is harder to design – discussed in earlier in this chapter.

Another problem with acceptance is guidance dependency (defined in chapter 3.2.19). According to Cockburn, guidance dependency is one of the main causes of groupware failure. Bullen and Bannett, in a study, found out that users often ignored the possibility of assigning a category to a message, even if they knew that this would increase the benefit for the group. The reason for this situation was that the groupware was guidance dependent – users need to invest additional effort to achieve group benefits.

For more sources of information about acceptance of groupware systems we refer to [6], [16], [31] and [35].

6 Groupware standards

The importance of interoperability of groupware systems is discussed in chapter 5.1.2. It can be seen as the introduction and motivation for this chapter.

Hofte in [6] wrote that *interoperability between similar groupware products from different vendors is rare.* The motivation of emerging groupware standards is also the spread of use of groupware in inter-organizational settings. Standards has been established in those types of groupware which were market driven, namely:

- computer conferencing
- multimedia conferencing
- workflow management
- group scheduling and calendaring

Each of these types is discussed in detail below.

6.1 Computer conferencing

The first wide spread conference tool were *Usenet News*. Although it was not an official Internet standard *RFC 1036* made possible that hundreds of discussion groups were formed on various topics.

Another computer conferencing standard is OSI Group Communication Service. This standard was less successful in terms of adoption. This standard is developed on top of the standards for Message Handling Systems and Directory Service. It defines services and protocols for conferences and specific applications e.g. electronic voting. This standard only reached the lowest level of standards documents – Committee Draft. Afterwards the standardization process was abandoned.

After the rapid spread of the Internet and especially Web in the 90's, the IETF realized that there were not standardized for many years, a significant factor in the lack of interoperability, platform dependence, security issues, cost and market segmentation. In 2003 IETF established a working group to establish a standard for Web conferencing. The *XCON: Centralized Conferencing Working Group* has defined the following goals:

- to establish a basic floor control protocol. This goal was finished and published in [2006] as RFC 4582: Binary Floor Control Protocol (BFCP)

- to establish a mechanism for membership and authorization control
- to establish a mechanism to manipulate and describe media "mixing" or "topology" for multiple media types (audio, video, text)
- to establish a mechanism for notification of conference related events/changes (for example a floor change).

XCON defines in its charter some items that are out-of-scope namely: Voting, Fully distributed conferences, Loosely-coupled conferences (no central point of control), Far-end device control, Protocol used between the conference controller and the mixer(s), Capabilities negotiation of the mixer(s), aster-slave cascaded conferences. The working group will coordinate their activities closely with the SIPPING and MMUSIC (see chapter 6.2) working groups. The XCON working group is still active and preparing new documents at this time. For more information on XCON we refer to the official ietf.org website.

6.2 Multimedia conferencing

The International Telecommunication Union worked since 1989 on standards for multipoint multimedia conferencing. The *T.120 series* defines protocols for multipoint communication, generic conference control, multipoint still image and annotation protocol, multipoint binary file transfer and multipoint application sharing protocol. The core of these standards were adopted by several vendors and implemented in products such as Intel ProShare and others.

Other multimedia conferencing standard – Multiparty Multimedia Session Control (mmusic) - was developed by the Internet Engineering Task Force (IETF) Working group. It described the management and coordination of multiple sessions and their multiple users in multiple media over the Internet [6]. Currently the IETF states on their website that *The Multiparty MUltimedia SessIon Control (MMUSIC) Working Group was chartered to develop protocols to support Internet teleconferencing and multimedia communications. These protocols are now reasonably mature, and many have received widespread deployment.* MMUSIC also maintain and revise the specification of the Real Time Streaming Protocol (RTSP). However the mmusic is currently still active and more information can be found on their website.

Another multimedia standard is the Session Initiation Protocol (SIP). It is an application-layer control protocol for creating, modifying, and terminating sessions with one or more participants.

This protocol was firstly designed in 1996 by Henning Schulzrinne and Mark Handley. After that IETF formed the SIP working group for further work on the protocol.

SIP brought to multimedia communication these features:

- is lightweight - SIP has only six methods and reduces complexity
- it is transport-independent - SIP can be used with UDP, TCP, ATM and other protocols
- it is text-based - allows humans to read SIP messages.

Further information about current SIP working group activities can be found on the ietf.org page.

6.3 Workflow management

The Workflow Management Coalition (WfMC) started their work on workflow standards in 1993. The WfMC *creates and contributes to process related standards, educates the market on related issues, and is the only standards organization that concentrates purely on process* [wfmc.org]. There are two standards that the WfMC has issued: Wf-XML and XPDL.

Wf-XML (Workflow XML) is an application specification developed for defining how the XML language is used to communicate workflow related processes and data between different workflow applications. Wf-XML extends the Asynchronous Application Service Protocol (ASAP) for SOAP with workflow functionality.

The XML Process Definition Language (XPDL) is a format to interchange Business Process definitions between different workflow products like modeling tools and workflow engines. Currently XPDL is at version 2.0 and was issued in October 2005.

XPDL is often compared to BPEL (Business Process Execution Language) – which is another workflow standard. The difference though is that BPML is conceived *of as a block-structured programming language*, and flow control (routing) is handled entirely by block structure concepts. The block structure might be recursive. On the contrary XPDL is conceived *of as a graph-structured language with additional concepts to handle blocks*. Routing is handled by specification of transitions between activities and process definitions cannot be nested. The differences between XPDL and BPEL are explained in detail in [32].

Closing with YAWL – Yet Another Workflow Language. Even the name itself tells that there are too many possible standards for workflow modeling. This also is an alternative to BPEL

which on the contrary has the advantage that it is being supported by several important workflow „players“ such as IBM or Microsoft.

However there is more work which was done by the WfMC e.g. the Business Process Modeling Notation which *provides a graphical notation to facilitate human communication between business users and technical users, of complex business processes* [wfmc.org].

There are also some open specifications defined by the BPMI.org such as the Business Process Modeling Language (BPML), and the Business Process Query Language (BPQL) which are now merged with the OMG's (Object Management Group) work on business process modeling.

The work around workflow and business process modeling is still active and is being changed over time. For this reason we only mentioned the current most important standards and specifications. The reader will find more information on the particular websites.

6.4 Group scheduling and calendaring

The history of standards which defined the calendaring and scheduling is in our opinion interesting. This reason for we decided to include a short history also in this work.

Firstly there were solutions for synchronizing calendars from e.g. Oracle or Microsoft. These though did work only within a certain organization. This caused problems in scheduling inter-organizational meetings. This reason for Netscape Communications in July 1996 announced the formation of a working group dedicated to developing standards for calendaring and scheduling. The group was formed under IETF as the CalSch – Calendaring and Scheduling working group.

The work of CalSch was divided into three main categories:

- data model and its text representation for calendar events – which successfully introduces the iCalendar specification.
- transport of calendar data via email – this resulted in iMIP – iCalendar Message-based Interoperability Protocol
- calendar access and scheduling – CAP – Calendar Access Protocol.

CalSch first built the vCalendar specification which refinement finally in November 1998 resulted in the RFC 2445 – iCalendar standard. The iCalendar standard is currently widespread used by almost all commercial as well as open source applications. It is a non-XML format for representing attribute-value pairs. It defines four core objects: VEVENT, VTODO,

VJOURNAL and VFREEBUSY and each of them defines a particular object which is used in every calendar. An example of a VTODO with a VALARM is showed on image 11:

```
BEGIN:VCALENDAR
VERSION:2.0
PRODID:-//ABC Corporation//NONSGML My Product//EN
BEGIN:VTODO
DTSTAMP:19980130T134500Z
SEQUENCE:2
UID:uid4@host1.com
ORGANIZER:MAILTO:unclesam@us.gov
ATTENDEE;PARTSTAT=ACCEPTED:MAILTO:jqpublic@host.com
DUE:19980415T235959
STATUS:NEEDS-ACTION
SUMMARY:Submit Income Taxes
BEGIN:VALARM
ACTION:AUDIO
TRIGGER:19980403T120000
ATTACH;FMTTYPE=audio/basic:http://host.com/pub/audio-
files/ssbanner.aud
REPEAT:4
DURATION:PT1H
END:VALARM
END:VTODO
END:VCALENDAR
```

Image 11: The example of a VTODO and VALARM notation in the iCalendar standard.

The EVENT object describe an event that represents a scheduled amount of time on a calendar. The VTODO describes a to-do item in the calendar. The VJOURNAL object is defined to be used as a descriptive text note which defines e.g. the progress of a todo item. In practice only few implementations of iCalendar support VJOURNAL entries as it does not make changes to free/busy times in the calendar and though is not frequently demanded by the users. The VFREEBUSY object describes a published set of busy time of a request for free/busy time. Other object defined are VALARM (also used in the example above) and VTIMEZONE for defining time zones.

Another effort of CalSch was the iCalendar Transport-independent Interoperability Protocol – iTIP which had to be used for calendar retrieving and scheduling operations. iMIP described how to perform iTIP operations through email. Despite some implementations these specifications were not adopted and are not commonly used today.

CAP should perform calendar request actions such as search for free time in other users calendars and schedule meetings. CAP was a *entirely new protocol that was distinct from all existing application-layer protocols, although it borrowed somewhat from the Post-Office*

Protocol (POP) for its interaction style. [45]. After the last versions in July 2003 CalSch made no further progress on CAP and the CalSch working group in IETF was closed. Lisa Dusseault wrote that *After four years of development, CAP was dead.* [45].

The major problem with CalSch was that the development of CAP was progressing very slowly. The functionality of CAP was overbid by Apple's iCal application (released in 2002) which has the functionality of sharing of calendars over the Internet. The sharing of iCal was implemented over WebDAV - Web Distributed Authoring and Versioning (RFC3744). With WebDAV users could publish their calendars to WebDAV servers and anyone else could view and download events. Apple's choice of WebDAV was s success because it was by the time an open used protocol and thus was accepted easily. After that few open-source application adopted the idea of iCalendar sharing over WebDAV. As the community grow this became *de facto the first open calendaring standard* [45]. The ease of WebDAV – an extension of HTTP which introduced overwrite prevention, namespace operations and metadata - also was an advantage compared to CAP which was a completely new protocol and was not supported by internet servers by the time. *Apple showed, in a very public way, that calendars could be treated like any other Web resource accessible via HTTP* [45].

Although Apple's iCal was a great success it had also its drawbacks. The use of the calendar for corporate purposes was not absolutely ideal. The main problems were searching for free-busy times across a large set of people and managing someone others calendars. A question was raised whether to extend WebDAV to support these functions or to build a new protocol like CAP. Apple's example of success lead into forming of the CalDAV (RFC 4791) protocol which is a extension to WebDAV.

According to CalDAV's web page *CalDAV is a protocol allowing calendar access via WebDAV. CalDAV models calendar events as HTTP resources in iCalendar format, and models calendars containing events as WebDAV collections. This allows users to publish and subscribe to calendars, share them collaboratively, synchronize between multiple users and synchronize between multiple devices* [caldav.org]. In [45] three main features of CalDAV are introduced:

- calendar maintenance – users can create and edit their calendars, free time slots etc.
- calendar queries – people can search for free/busy times of others, discover who is participating in a concrete meeting.
- calendar security – users adjust the level of visibility of their calendars, and permissions to change the calendars by others.

At the time of writing of this thesis the last information about CalDAV was published on 5th of October 2006 and states that the IETF and IESG approved CalDAV to become a Proposed Standard. RFC number is yet to be chosen. Apple implemented latest CalDAV features in their iCal 3.0 which is a part of their new Mac OS X 10.5 „Leopard“ and also released the Darwin Calendar Server which is available as open source.

6.5 Other standard initiatives

Another initiative in groupware calendaring is GroupDAV. Unlike CalDAV it includes not only calendar elements but also applies contacts, notes of other types of objects. The difference to CalDAV is that, the scope of GroupDAV is only to coordinate storage of items (although more than only calendar) whereas CalDAV defines also queries and calendar operations.

GroupDAV defines itself on their website as *an effort to create a "down-to-the-earth" protocol to connect Open Source groupware clients with Open Source groupware servers* [groupdav.org]. Furthermore they define their scope as follows: *A major goal of GroupDAV is to keep the protocol as simple as possible and to stay focused on real world issues with Open Source and free software applications.*

7 Conclusion

During the elaboration of this thesis we analyzed more than forty literature sources from diverse areas of computer science. With the respect to the goals stated in Chapter 1, we found relevant answers and proposals for solutions. Let us summarize these results in the following indents:

- *Are the terms groupware and CSCW in literature correctly and unambiguously defined? Is there a definition that defines groupware according to present view on groupware? If not, we will try to do so.*
  There were many definitions found in literature, although some of them were out of date. We found one definition that, in our opinion, defines groupware correctly and unambiguously, namely the definition of groupware by Tom Brick. See chapter 2.3.2.
- *What are the main requirements on groupware systems by users? Is it possible to create a classification for groupware systems? Are there any existing classifications and are they relevant in present time? How can these categories be defined? Are these categories disjoint?*
  We described generic groupware requirements in Chapter 5.2.2. There were existing classifications of groupware according to several factors. We described the usefulness of them and used the relevant ones in the groupware characteristics table (appendix A). We created a classification system for groupware according to functional criteria which are defined in chapter 4 and shown in appendix B. Categories described in Chapter 4 are not disjoint, but they provide the user with a more detailed information which would not be possible with disjoint categories.
- *Are there other (groupware independent) factors that influence the choice of a concrete groupware system?*
  Yes, there are other factors that influence the choice of groupware applications in a relevant way. The most important ones are listed in chapter 3.4. and they are included in the groupware characteristics table (appendix A).
- *What specific problems can be identified in groupware development process? Which of these problems are seen as most critical in respect to groupware history?*
  Groupware design issues are described in detail on chapter 5. We believe that groupware development is in many respects different to other software development. One of the most important differences is the fact that groupware includes people, as well, and this raises many social questions and consideration on the system. Most

critical problems in groupware development are caused by incorrect gathering of group requirements, lack of interoperability and technical problems in solving presence and awareness.

- *Are there any standards which should be satisfied by groupware systems?*
  Some of groupware development standards are currently used, some though were not successful. There are also some standards initiatives that are currently in development.

During the study and work on this thesis we encountered further open questions and challenges in the research field of CSCW. Therefore we would like to close this thesis with a proposals for future work which could be based on some aspects or views discussed in this thesis.

The reader certainly noticed that the most important technical challenges that we see in future groupware development is the focus on usability in two respects. First, improving *user-interface design* for enhancing of awareness and improving the ease of use of the applications. Second, the groupware interoperability which is very important for real life groupware use. From the social point of view, the process of gathering group requirements might be described and formalized to help designers to realize more precisely what the groupware should bring to the group as a whole, not only to some individuals of it. Finally we appreciate the standardization initiatives of WfMC, GroupDAV and CalDAV which may lead into wide spread and accepted standards for groupware development, and in this way help groupware to broaden its boundaries.

References

[1] STEIN, Dominik. *Definition und Klassifikation der Begriffswelt um CSCW, Workgroup Computing, Groupware, Workflow Management.* Seminararbeit WS 96/97, Universität GHS Essen, 1997.

[2] BRICK, Tom. *Groupware.* http://www.usabilityfirst.com/groupware, 1998

[3] GRUDIN, Jonathan. *Computer-supported cooperative work: history and focus*. IEEE Computer, Volume 27, Number 5, May 1994

[4] EASTERBROOK, Steve. *Coordination Breakdowns: Why Groupware is so Difficult to Design.* Proceedings of the 28th Hawaii International Conference on System Sciences, 1995

[5] OSTER, Gérald, URSO, Pascal, MOLLI, Pascal, IMINE, Abdessamad. *Real time group editors without Operational transformation*. INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE (INRIA) – Rapport de reserche N° 5580, May 2005

[6] HENRI TER HOFTE, G. Working Apart Together - Foundations for Component Groupware. Enschede, The Netherlands, 1998

[7] Pfeifer, John. *The Domain and Goals of CSCW.* University of Calgary, 1995

[8] TWIDALE, Michael B., NICHOLS, David M. *A Survey of Applications of CSCW for Digital Libraries.* Technical Report CSEG/4/98, Computing Department, Lancaster University, 1998

[9] SCHMIDT, Albrecht, LAUFF, Markus, BEIGL, Michael. *Handheld CSCW.* Proceedings of the Workshop on Handheld CSCW at CSCW '98, 14 November, Seattle, 1998

[10] GREENBERG, Sual, BOYLE, Micheal. *Moving Between Personal Devices and Public Displays.* Proceedings of the Workshop on Handheld CSCW at CSCW '98, 14 November, Seattle, 1998

[11] RUBART, Jessica, WANG, Weigang, HAAKE, Jörg M. *Supporting Cooperative Activities with Shared Hypermedia Workspaces on the WWW.* Proceedings of WWW2003 Conference, Budapest, 2003

[12] KOCH, Michael. Community-Unterstützungssysteme - Architektur und Interoperabilität. Technische Universität München, Fakultät für Informatik, 2003

[13] HESS, H. *Storage of Groupware Objects in WebDAV (GroupDAV).* Draft of openGroupware.org, 2004

[14] F. Dawson, D. Stenerson. I*nternet Calendaring and Scheduling Core Object Specification (iCalendar).* IETF - RFC 2445, 1998

[15] Ochoa, Sergio. An Architectural Pattern for Computer-Mediated Communication in Groupware Systems. Paper of International Conference on Enterprise Information Systems, 2002

[16] KOCH, Thomas. *Groupware on the Internet*. Technical University Graz, Diploma Thesis, 1998

[17] HILL, Jason M. *A Direct Manipulation Toolkit for Awareness Support in Groupware.* University of Saskatchewan, Master Thesis, 2003

[18] TOLLMAR, Konrad. *Towards CSCW Design in the Scandinavian Tradition.* Stockholm University, Doctoral Dissertation, 2001

[19] CHUNYUAN, Liao, YUANCHUN, Shi, GUANGYOU, Xu. *A Task Management Model in CSCW.* Dept. Of Computer Science, TsingHua University, Beijing, P.R.China, 1998

[20] BANNON, Liam J., HUGHES, John A. *The Context of CSCW.*

[21] RAMA, Jiten, BISHOP, Judith. *Survey and comparison of CSCW Groupware applications.* Proceedings of SAICSIT 2006, 2006

[22] ROTH, Jörg, UNGER, Claus. *An extensible classification model for distribution architectures of synchronous groupware.* Proceedings of Fourth International Conference on the Design of Cooperative Systems, 2000

[23] STOREY, Margaret-Anne. *Visualization, Knowledge Management and CSCW.* Course notes of Information Visualization and Knowledge Management, University of Victoria, 2003

[24] PINELLE, David. *A Survey of Groupware Evaluations in CSCW Proceedings.* Proceedings of 9th IEEE Workshop on Enabling Technologies - Infrastructure for Collaborative Enterprises, 2000

[25] GRÜNBACHER, Paul, HALLING, Michael, BIFFL, Stefan. *An Empirical Study on Groupware Support for Software Inspection Meetings.* Proceedings of the 18th IEEE International Conference on Automated Software Engineering (ASE'03)

[26] PINELLE, David, GUTWIN, Carl. *A Review of Groupware Evaluations.* Proceedings of the 9th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, 2000

[27] PENDERGAST, Mark O. *A Comparative Analysis of Groupware Application Protocols.* ACM SIGCOMM Computer Communication Review, 1998

[28] ZABELE, Stephen, ROHALL, Steven L., VINCIGUERRA, Ralph L. *High Performance Infrastructure for Visually-intensive CSCW Applications.* Proceedings of the 1994 ACM conference on Computer supported cooperative work, 1994

[29] FOUSS, Jonathan D., CHANG, Kai H. *Classifying Groupware.* Proceedings of the 38th annual on Southeast regional conference, 2000

[30] MANDVIWALLA, Munir, OLFMAN, Lorne. *What Do Groups Need? A Proposed Set of Generic Groupware Requirements.* ACM Transactions on Computer-Human Interaction (TOCHI), 1994

[31] SCHLICHTER, Johann. *Computergestützte Gruppenarbeit.* Student Script für Computergestützte Gruppenarbeit, 2003

[32] SHAPIRO, Robert. *A Comparison of XPDL, BPML and BPEL4WS.* Technical University Graz, 2002

[33] HILL Ralph D., BRINCK Tom, ROHALL Steven L., PATTERSON John F., WILNER Wayne. *The Rendezvous Architecture and Language for Constructing Multiuser Applications.* ACM Transactions on Computer-Human Interaction, Vol 1, 1994,

[34] ELLIS, Clarence (Skip), WAINER, Jacques. *A Conceptual Model of Groupware.* Proceedings of the 1994 ACM conference on Computer supported cooperative work, 1994

[35] LAUSEN, Holger. *Groupware Evaluation.* DERI – Digital Enterprise Research Institute - SemanticWeb.org, 09.01.2004

[36] DELOTTE, Olivier, DAVID, Bertrand, CHALON, René. *Task Modelling for Capillary Collaborative Systems based on Scenarios.* Proceedings of the 3rd annual conference on Task models and diagrams, 2004

[37] PATTERSON, John F. *A Taxonomy of Architectures for Synchronous Groupware Applications.* SIGOIS Bulletin, April 1995/Vol. 15, No. 3

[38] GUTWIN, Carl, FEDAK, Christopher, WATSON, Mark, DYCK, Jeff, BELL, Tim. *Improving Network Efficiency in Real-Time Groupware with General Message Compression.* Proceedings of the 2006 20th anniversary conference on Computer supported cooperative work, 2006

[39] BEAUDOUIN-LAFON, Michel, KARSENTY, Alain. *Transparency and Awareness in a Real-time Groupware System.* Proceedings of the 5th annual ACM symposium on User interface software and technology, 1992

[40] GREENBERG, Saul, MARWOOD, David. *Real Time Groupware as a Distributed System: Concurrency Control and its Effect on the Interface.* Proceedings of the 1994 ACM conference on Computer supported cooperative work, 1994

[41] TER BEEK, Maurice H., MASSINK, Mieke, LATELLA, Diego, GNESI, Stefania, FORGHIERI, Alessandro, SEBASTIANIS, Maurizio. *A Case Study on the Automated Verification of Groupware Protocols.* Proceedings of the 27th international conference on Software engineering, 2005

[42] HUANG, Elaine M. *Design and Analysis of Groupware for Large Displays.* CHI '05 extended abstracts on Human factors in computing systems, 2005

[43] CÍGER, Ján. *Computer-Supported Cooperative Work - Lecture 1*. Lecture notes, Department of Medialogy, Aalborg University Esbjerg, 2006

[44] ROSS, Charles A. My Life With Groupware. College of Undergraduate Studies, Cincinnati, Ohio, 2001

[45] WHITEHEAD, Jim, DUSSEAULT, Lisa. *Open Calendar Sharing and Scheduling with CalDAV*. IEEE INTERNET COMPUTING March/April 2005 (Vol. 9, No. 2), 2005

[46] FEILNER, Markus. *Auf der Suche nach dem Groupware-Standard*. http://www.linux-magazin.de/, 2007

List of used images:

Abstrakt

V diplomovej práci uvedieme čitateľa do vednej disciplíny „Počítačom podporovaná kooperatívna práca“ (Computer Supported Cooperative Work - CSCW) a groupware. Uvedieme samotné definície CSCW a groupware, definície súvisiacich pojmov, ako aj históriu v tejto oblasti. Následne zosumarizujeme najdôležitejšie faktory, podľa ktorých je možné kategorizovať groupware systémy a uvedieme možné funkcionality groupware aplikácií. Rozhodneme o relevancií každého z uvedených faktorov. Vybrané faktory budú slúžiť ako základ pre samotnú kategorizáciu. Ďalším cielom práce je poukázať na najčastejšie problémy pri vývoji groupware aplikácií s rôznych pohľadov. Prácu uzatvára kapitola, ktorá prezentuje aktuálne akceptované štandardy v oblasti groupware ako aj dôvody neúspechu štandardizačných iniciatívit v minulosti.

kľúčové slová: *Groupware, CSCW, groupware definície, história groupware, kategorizácia groupware, problémy vývoja groupware, šťandardy v groupware*