1.1 Groupware: Systems that Support Computer-Mediated Interaction

A famous definition of the term groupware defines groupware systems as "intentional group processes plus software to support them" (Johnson-Lenz and Johnson-Lenz, 1981). This definition includes various aspects that we have to consider when designing groupware solutions:

- The core of the definition is the group. A group of users wants to interact using groupware. Naturally, the members of the group should play an important role in the design of groupware. The groupware design has the purpose of creating a solution that satisfies the user's needs. End-user requirements therefore have to be the central issue during groupware design.

- The group interacts in an intentional group process. The interaction between people needs to play an important role in the design of any groupware solution. It has to become clear who interacts with whom. How strict the intentional group process is must be considered, ranging from unplanned interaction in virtual communities up to formally structured workflows in a distributed workgroup.

- The process is supported by software. The fact that software is mentioned third here emphasizes that the software itself should be a supportive facility to ease the interaction between people. The software should be adapted to the users' needs to best fulfill its supportive role. At this point the software developer comes into play. As software supports the group process, the software developer should support the users in adapting or creating software that fits the process.

Compared to a focus on design, which has the goal of supporting the group in the manipulation of content, support for social group interaction needs a broader focus, that of the relationships between users. Tools for manipulation are in most cases used by one user (even in collaborative systems), so they affect the relationship between the user and shared artifacts. Social interaction, on the other hand, affects the relationships between users and needs to address issues like trust and privacy. In contrast to the design of tools for the manipulation of artifacts, which mainly affects human-computer interaction, the focus should thus be on human-computer-human interaction (HCHI). The design of tools therefore focuses on the interaction of the user with the artifact, and considers the human-human interaction as a marginal aspect.

To provide customized designs of groupware mechanisms, we have to make use of a design process that is flexible enough to adapt to the group's needs. Experiences with the design of single user applications have already shown that many software development projects fail because of requirements inadequacies (Dorfman, 1997).

In such cases, the customer is typically involved in the early stages of the project as a source of design requirements. This set of requirements is then implemented by the software developers and subsequently the customer assesses the result. However, if the requirements were not specified correctly, customers receive a product that does not match their needs. This means that requirements in the context of computer-mediated interaction must always address social aspects as well as technical aspects, which is why they are called socio-technical requirements.

Unfortunately, these socio-technical requirements are often less clear to the stakeholders involved in the development of groupware applications. Two factors make this part of groupware development difficult:

- While in single user tasks, such as word processing or image editing, only one actor interacts with an artifact, groupware needs to support the interaction of many users with each other. An interaction partner is thus not a technical, deterministic, artifact, but a non-deterministic human.

- Users are not as familiar with using these new opportunities for interaction compared with single-user applications.

The theory of socio-technical design views a community from two perspectives: the social system, including group processes, roles, and flow of information, and the technical system, which includes tools used within the community, such as IT infrastructure or buildings. From a socio-technical perspective these two systems are highly interrelated. A socio-technical perspective on groupware design has to be aware of three key aspects (Bikson and Eveland, 1996):

- It is difficult to predict the reciprocal effect of changes to either the social or the technical system.

- The process used to create the socio-technical system will affect the acceptance of the system.

- Both social and technical systems change over time.

The tools in the technical system, i.e. the software that supports intentional group processes (Johnson-Lenz and Johnson-Lenz, 1981), can be classified in many different ways. One popular way to classify groupware is to distinguish how it support groups. Teul et al. (1995) introduced such a model and distinguish between three different main support functionalities:

1. Communication focuses on the information exchange between cooperating group members.

2. Coordination concentrates on coordinating group tasks.

3. Cooperation adds the ability to accomplish group goals to the above support functionalities.

As all main support functionalities start with the letter C, Borghoff and Schlüchter (2000) later on called this approach 3C-classification. In their initial proposal
Teufel et al. (1995) positioned the three main functionalities in a triangle to cluster groupware applications in system classes of common functionality, i.e., communication systems, workflow management systems, shared information spaces, and workgroup computing systems.

In contrast to the initial approach of Teufel et al. (1995), we propose a different approach. Figure 1.1 places well-known groupware applications in two-dimensional space. The vertical axis denotes the application’s support for coordination, while the horizontal axis is used to denote the degree of communication and cooperation that an application supports. This is possible because a higher degree of communication implies a lower degree of cooperation. By placing an application in this two-dimensional space, the individual degree of communication, coordination, and cooperation can be visualized much better for each of the application types.

In particular, we distinguish the following groupware applications in Figure 1.1:

- **Audio/Video conferencing** tools allow users to communicate by various means, so they have a high degree of communication and a low degree of cooperation. Compared to workflow management systems, for example, they do not explicitly offer functionality for scheduling or organizing tasks. We thus see them at a medium degree of coordination.

- **Chat** tools have a lower degree of communication than audio/video conferencing tools, as non-verbal information is omitted when communicating via a chat application. For the same reason, the degree of coordination is reduced.

- **Group Decision Support Systems (GDSS)** are explicitly designed to support groups in decision-making. For that purpose, they offer synchronous as well as asynchronous communication tools, group votes, etc. They therefore have a high degree of communication and coordination.

- **E-mail** is the most popular groupware application. E-mail can be used for many purposes, but its main purpose is to support communication. As the communication is asynchronous and text-based, the degree of communication is reduced compared to chat tools. However, as users can structure their information when using e-mails, the degree of coordination is increased.

- **Forums** allow users to discuss a topic in which they are interested. The communicating group is therefore defined by the topic. Compared to e-mail, communication is more public. However, if used in a company, forums allow coordination of a group that is cooperating on a common task.

- **Community Systems** integrate a variety of tools and allow a large group of users, i.e., a community, to communicate, to share information, or to coordinate common activities. Often, these tools are web-based. Compared to the tools listed above, community systems have better support for accomplishing and coordinating group goals. However, the degree of communication possible is lower, as there is no possibility of communicating directly with individual community members.

- **Wikis** are web-based systems that allow users to change the content of the web pages. Wikis have their origin in the design patterns community. The first Wiki was the Portland Pattern Repository, which was created in 1995 by Ward Cunningham. As Wikis allow users to create and share content, they have a high degree of cooperation, but as they do not explicitly support communication or coordination, they are low in these respects.

- **Shared workspaces** such as BCSW (see Section 1.1) allow users to share content. In the most cases, they also allow structuring of the shared content to coordinate common tasks. For that reason, shared workspaces have a higher degree of cooperation and coordination than Wikis.

- **Multi-player games** are becoming more and more popular. They allow users to solve tasks or quests jointly, and support a number of coordination functionalities for that purpose. Communication is mainly short and used only for coordination, which explains the degree of communication, coordination, and cooperation they exhibit.
Workflow Management Systems (WFMS) are tools that allow modeling, coordination, supervision, and evaluation of a workflow by a cooperating team. For that reason they exhibit the highest degree of coordination of all tools. As their main purpose is to coordinate users in accomplishing a group goal, they have also a high degree of cooperation. WFMS only use communication for coordination purposes, for example to pass on a task or to notify about a completed task, so they show a quite low degree of communication.

Multiuser editors such as CoWord (see Section 1.2) allow cooperating users to create a shared artifact synchronously, for example a text document, drawing, or a spreadsheet, and thus accomplish group goals. This explains the high degree of cooperation of such tools. Multiuser editors use a lot of coordination functionalities as well, for example to avoid conflicting changes. Communication is not explicitly supported, thus the degree of communication is low.

Apart from the various main functionalities that are supported by a groupware application, awareness plays an important role. Of the tools listed above, multiuser editors, for example, make use of awareness widgets that show the working area of other users, with the goal to avoid conflicting changes in a shared artifact. Awareness can be seen as a mediator between these three main functionalities.

![Diagram](image)

**Figure 1.2** Relationship between communication, coordination, cooperation, and mediating group awareness

Gerosa et al. (2004) describe this as shown in Figure 1.2. In this figure, cooperating users must be able to communicate and to coordinate themselves. When communicating, users might generate commitments and define tasks that must be completed to accomplish the common group goal. These tasks must be coordinated so that they are accomplished in the correct order and at the correct time with respect to possible external restrictions. To accomplish these tasks the users have to cooperate in a shared environment. However, while cooperating, unexpected situations might emerge that demand new communication. In such communication new commitments and tasks might be defined, which again must be coordinated to be accomplished in cooperation. Apart from this cyclic aspect of cooperation, Gerosa et al. place awareness in a central position in Figure 1.2. Every user action that is performed during communication, coordination, or cooperation generates information. Some of this information involves two or even more users, and should be made available to all cooperating users so that they can become aware of each other. This helps to mediate further communication, coordination, and cooperation. Based on this information, users are able to build up a shared understanding of their common group goals and to synchronize their cooperation.

Now that we have clarified our understanding of groupware, the following section presents a scenario in the not too distant future. This will serve as a running scenario throughout the book. It will relate the patterns in the book to a practical example and show how they can be applied in the scenario.
1.2 A Day with Paul Smith

Join us on a ride with a time machine. Our destination is a typical working day in the life of Paul Smith. Paul is a software engineer and works in the software development department of a leading entertainment device company in London. Currently, Paul is the project leader in the COGE project in which a Cooperative Game Engine is being developed.

Paul's company has subsidiaries all over the world. The members of Paul's team are distributed as shown in Figure 1.3. One team of developers is located in Rio de Janeiro, one in London, and a third in Hong Kong. The main customer is a large game manufacturer located in Germany, which has the goal of building an educational game that helps better understanding of water supply in African countries. The game manufacturer has a group of African pilot users located in Ethiopia and Malawi.

Most of the projects in Paul's department are performed in teams to benefit from the synergy of people with varied expertise. Currently, the following interactions are present in the COGE project: the developers from London continue to work on the results that were created only hours before in Hong Kong. Both software teams communicate to plan the internal architecture of the game engine. In other meetings, the London team collaborates with the German customer, which integrates the game engine in its project. The German customer also communicates with some of the developers in Hong Kong or Rio if time allows interaction.

Finally, the German customer interacts with their pilot users and collects suggestions from them on how the game could be improved.

For their common tasks, team members interact daily using their computing devices. Let's now take a look how Paul's typical working day starts.

6:30 AM. The alarm clock rings and Paul gets out of bed. After a shower and a shave, Paul prepares his breakfast. While eating his cereal and enjoying his freshly brewed coffee, Paul has a look at his electronic newspaper (see Figure 1.4). The electronic newspaper shows Paul the latest news in specific categories in which he is interested. Paul is an enthusiastic member of the pattern community and participates in an online community that writes, discusses, and shares patterns. He has therefore configured a special section in his electronic newspaper that shows him the latest pattern community news and information about his buddies. The daily report tells Paul what has happened in his online community and allows him to keep track of interesting discussions. A sidebar in the newspaper shows Paul's buddy list. As some of Paul's buddies are already awake, Paul has a short chat with them and agrees to arrange a meeting in the evening.

To plan his working day, Paul checks his main tasks for the day and the achievements of his colleagues during the night. In Hong Kong they have solved one of the major problems with the network protocol for the new cooperative game engine. However, the solution has raised some new problems in a module that is developed by the team in Rio. Paul therefore decides to announce a meeting with the colleagues in Rio for the afternoon. He enters the collaboration space and sends invitations to those involved.
8:30 AM. Paul leaves his house in a small neighborhood in the London suburbs, gets into his car, and sets off for his office in the city. In the car Paul recalls the destination from his favorite destinations folder. The navigation system of the car not only connects to GPS satellites, but also to the Internet to plan the best route into the city. It uses GPS to detect Paul’s position and the Internet to avoid traffic jams. Additionally, Paul sends his route to his office in an online travel portal that mediates travel mates. Travel mates are selected not only according the destination but also according to Paul’s topics of interest. The latter is quite important for Paul, as he does not want to share a ride with someone with whom he has nothing to talk about. In most cases, this allows Paul to pick up a travel mate on his way into the city.

Figure 1.5 Paul looks for a new travel mate

This morning the travel portal suggests a new travel mate (see Figure 1.5). Paul does not know this person, but the portal uses a recommendation system, and the travel mate is ranked as a trustworthy and interesting person. Paul has an additional look at the user gallery and reads the introduction of the proposed travel mate. Paul is satisfied with the suggestion and decides to stop on his way into the city and to pick up the suggested travel mate. The car navigation system calculates the estimated pick-up time and notifies the travel mate. It also keeps the travel mate aware of probable changes so that she does not have to wait too long.

9:30 AM. After picking up the travel mate and dropping her at her destination, Paul arrives in his office. A biometric security check at the entrance proves Paul’s identity, and Paul moves to the project’s group room where he meets his colleagues. Video screens show the offices of colleagues in Frankfurt, Hong Kong, and Rio de Janeiro in a permanent video stream. One of the colleagues in London starts a discussion about the project’s current problems. Paul suggests postponing the discussion until the afternoon when colleagues from Brazil will also be available. Currently, nobody is in the office in Rio, as it is not yet morning there.

As plenty of time is left before the general meeting, Paul joins a group that discusses the software architecture of the current project (see Figure 1.6). This group meets in a special room that allows 3D projections. Currently, the group is discussing parts of the architecture for the user interface. Luckily, this group is not affected by the problem that was raised by the solution from China, and makes good progress.

Figure 1.6 Paul participates in a virtual reality conference about the software architecture of the current project

When the meeting about the software architecture is over, Paul goes to his office to start up his desktop computer. He enters the group’s collaboration space and is pleased to see that everyone has accepted his invitation to discuss the new problems with the network protocol. The collaboration space then notifies Paul about newly received mails, who else is on line in the collaboration space, and open tasks. As the group has decided to use an open awareness concept, Paul can also see what everyone is currently doing by moving his mouse cursor over the images in his buddy list. This information is often used to start a spontaneous collaboration and discussion about ongoing problems. However, teammates who do not want to be disturbed indicate this in the buddy list so that the collaboration space does not allow direct communication.

1:00 PM. After a few more hours of work and a good lunch in the company’s canteen, it is time for the group meeting to discuss the new problems with the network protocol. The video screens show that the necessary people are available at all locations. Paul contacts them and announces the start of the meeting, and all his colleagues move to the group meeting room. This room is equipped with the 3D projector Paul used in the morning. This projector displays video streams for each participant from the various locations (see Figure 1.7), the virtual room for the meeting in the team’s collaboration space, and the current shared documents containing the description of the network protocol. This allows everyone to see each other and the material for discussion.
Paul opens the meeting by passing the floor to his colleague Gwan in Hong Kong. Gwan explains how they have solved one of the major problems with the network protocol. To do this, Gwan uses a virtual pointer that allows him to point to the corresponding lines in the source code. Other colleagues can discuss Gwan’s presentation using synchronous chat so that Gwan is not disturbed. They can also annotate the source code and post questions to a blackboard that will be discussed after Gwan’s explanation.

Everyone is impressed with Gwan’s presentation, although they know that his solution raises a new problem. After the open questions have been answered, Paul hands the moderation over to Rio de Janeiro and Ana explains the new problem. Ana’s presentation raises a lot of open questions on the shared blackboard. The group clusters the open questions and splits into subgroups to address these question clusters. The subgroups create new virtual rooms in the team’s collaboration space to discuss the open questions. Before the groups retreat to their new virtual rooms, Paul schedules a new meeting for tomorrow for the groups to present their results.

4:00 PM. Paul has his last meeting for the day. David, a colleague from Detroit, visits the lab. After giving David a short guided tour of the office, Paul tells him about the new problems with the network protocol. David starts smiling, as he knows how to solve part of the problem. Paul and David therefore enter the collaboration space and knock at the virtual door of the subgroup that formed this afternoon and whose questions David can answer. David offers himself as a mentor and to explain the technology that can solve part of the problem. Soon, David and the other colleagues are in deep discussion and Paul leaves to do other work in his office. Two hours later, David leaves the lab to catch his flight back to Detroit. The subgroup tells Paul that they have nominated David as an expert for specific topics in the collaboration space. This might help David with his next evaluation and wage bargaining.

8:00 PM. Paul has finally finished his most important tasks for the day. He uses his MDA¹ to connect to his online community. As soon as he is online, his friends contact him. They had thought that Paul had forgotten about their appointment. Paul had, and excuses himself for being late. Paul’s friends suggest watching a movie in one of the new cinemas downtown. A quick vote shows that all agree. They run a recommender system for movies, and after a short discussion agree on what to see (see Figure 1.8). Adriana offers to buy the tickets and reserve the seats in the cinema’s online booking system.

So a long working day finally ends, and Paul leaves his office to watch a movie with his friends. We can step back into our time machine and go on a short ride back to the present.

¹MDA is an abbreviation for mobile digital assistant which is a combination of a mobile phone and a personal digital assistant (PDA).
Chapter 1 Introduction

1.3 Outline

The scenario of Paul Smith shows one vision of the future. Our main prediction is that in future people will interact more and more using computing devices. In combination with software these computing devices will mediate interaction among people.

As the overview of groupware approaches shows, the scenario is not too far in the future, as most of the computer-mediated interaction it describes already happens in our lives, although not as an integral part of daily life. To mention a few, Paul's day starts with a look at the Periodic Report of his favorite online community, then at his Buddy List to see who else is already on line. The team is using a collaboration space that is based on virtual Rooms, Paul's colleague David acts as a Mentor, and finally Paul and his friends use a recommender system with Letters of Recommendation to select a movie for the evening.

The terms set in SMALL CAPS are patterns that are part of our pattern language for computer-mediated interaction. These and other patterns can be found in different chapters of this book, which is structured as follows:

Chapter 2 From Patterns to a Pattern-oriented Development Process introduces the reader to the theory of patterns. It looks at the original and more recent publications by Christopher Alexander. Using an end-user centered view, we transfer ideas to the domain of computer-mediated interaction. This results in a pattern form that is different than the pattern forms used in more technical pattern languages. While technical pattern languages use design diagrams or code fragments to illustrate solutions, we prefer a narrative way of presenting the patterns. This ensures that both end users and developers will be able to read the solution.

In the remaining part of this chapter, we will introduce OSDP, a pattern-oriented process for groupware development, which is based on piecemeal growth via short design and development iterations, as well as frequent diagnosis and reflection on working practices and how the application supports them.

Chapter 3 Community Support describes patterns at a high level of abstraction. The patterns in this chapter describe group processes and the use of computer technology to support such processes. Its main focus lies on the early phases of the group process. It answers questions such as:

- How to arrive in the community
- How to find out what is interesting in the community
- How to protect users

Chapter 4 Group Support provides patterns at the user interface level of a collaborative application. The patterns are both technical (describing how to design group interfaces) and social (elaborating on successful application of groupware technology). Problems solved are:

- How to modify shared material together
- How to shape places for collaboration
- How to organize textual communication
- How to become aware of other user's actions
- How to notice absent participants

Chapter 5 Base Technology discusses the technical layer of groupware applications. The patterns are mainly technical and answer the questions:

- How systems bootstrap for collaboration
- How systems manage shared data
- How systems ensure data consistency

Chapter 6 Examples of Applying the Pattern Language presents two case studies, one on BCSW and another on CoWord. These case studies show how group interaction can be supported by HCHI technology. The goal of this chapter is to put the patterns together and to illustrate how they are used by two well-known groupware applications.
Chapter 2

Related Work

This chapter discusses literature that is relevant to the proposed research. It begins by providing a high-level discussion of computer-supported cooperative work (CSCW) and groupware, and then current approaches used to design groupware systems are discussed. This chapter is divided into the following sections: Computer-supported cooperative work, Groupware design and Mobile groupware design recommendations.

2.1 Computer-supported cooperative work and groupware

Computer-supported cooperative work is a research area concerned with how computer systems should be designed to support group work and with the effect those systems have on group work patterns (Dix et al., 1998). The applications that are designed to support group work are often referred to as groupware, which has been defined as “computer-based systems that support groups of people engaged in a common task (or goal) and that provide an interface to a shared environment” (Ellis et al., 1991). The notions of a common task and a shared environment are crucial to this definition. This excludes multiuser systems, such as time-sharing systems, whose users may not share a common task. Note also that the definition does not specify that the users be active simultaneously.
A wide variety of groupware systems have been developed in recent years, and some have received widespread acceptance while others have had limited success (Grudin, 1994). Some examples include:

- Electronic mail (Sproull, 1993)
- Group calendars (Lange, 1993)
- Telemedicine applications (Horsch and Balbach, 1999)
- Co-authoring tools (Neuwirth et al., 1993)
- Group drawing tools (Greenberg et al., 1993)
- Audio- and video-conferencing tools (Bly et al., 1993)
- Workflow systems (Ellis, 1999)
- Instant messaging (Isaacs et al., 2002)
- Newsgroups and network communities (Shneiderman, 1998)
- Tabletop display groupware (Scott et al., 2003)
- Shared window systems (Lauwers et al., 1993; Lauwers and Lantz, 1993)
- Electronic meeting systems (Mentei, 1993; Nunamaker et al., 1993)
- Collaborative virtual environments (Hindmarsh et al., 1998)

Groupware systems are often classified according to the type of collaboration that they support. In this classification scheme, collaboration has a temporal and a spatial dimension, and these dimensions are commonly shown using the time-space matrix in Table 2.1 (Baecker, 1993; Dix, 1998; Ellis, 1999; Preece et al., 1994; Shneiderman, 1998). According to the matrix, modes of interaction differ along a time dimension and can be either synchronous (occurring at the same time) or asynchronous (occurring at different times). They also differ along a place dimension, and can be co-located (collaborators are in the same location) or distributed (collaborators are in different locations).
Table 2.1: Time-space matrix

<table>
<thead>
<tr>
<th>Place</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Same time</td>
</tr>
<tr>
<td>Same place</td>
<td>Face-to-face (tabletop displays, meeting support tools)</td>
</tr>
<tr>
<td>Different places</td>
<td>Synchronous distributed (shared editors, video- and audio-conferencing tools)</td>
</tr>
</tbody>
</table>

In the next sections, I briefly discuss each of the four types of groupware shown on the time-space matrix. The discussion is organized according to the following themes:

- Synchronous distributed groupware
- Synchronous co-located groupware
- Asynchronous distributed groupware
- Asynchronous co-located groupware

### 2.1.1 Synchronous distributed groupware

Synchronous distributed groupware allows users to work together at the same time even though they are in different locations (Baecker, 1993). Most of these applications provide shared workspaces where group members can create and edit shared artifacts such as images, documents, or agendas (Gutwin and Greenberg, 1999). These applications usually include real-time communication support using voice, video, or text messaging (Dix et al., 1998), and awareness features are often incorporated into the workspace to help each group member to understand others’ activities (Dourish and Bellotti, 1992; Gutwin and Greenberg, 1996).

A number of synchronous groupware tools have been developed to allow collaboration between physically distributed workers. Groupware toolkits such as TOP (Guerrero and Fuller, 2001), Rendezvous (Patterson et al., 1990), GroupKit (Roseman and Greenberg, 1996), and COAST (Schuckmann et al., 1999) are all intended to help developers build real-time groupware applications. Additionally, many groupware applications provide features that allow collaboration...
across a distance such as videoconferencing tools (Okada et al., 1994) audioconferencing tools (Rodenstein and Donath, 2000), shared whiteboards (Streitz et al., 1994), and shared editors (Olson et al., 1993).

2.1.2 Synchronous co-located groupware

Synchronous co-located groupware systems support face-to-face interactions between two or more collaborators. These systems help groups generate ideas and understanding, and common areas of support are research environments, design tasks, management meetings, and brainstorming sessions (Dix et al., 1998). These systems can provide users with a single shared interactive display (Kruger et al., 2003) or with separate individual networked clients (Bruce et al., 1992).

A range of synchronous co-located groupware systems have been developed. For example, Foster and Stefkik (Foster and Stefkik, 1986) developed Cognote to support idea generation in team meetings, and each team member has a separate networked client that allows them to enter new information into a shared information space. Pedersen (Pedersen et al., 1993) developed Tivoli, a single-display groupware application that uses a whiteboard metaphor. Users interact with the system’s large display using a stylus, and the system allows the group to save and organize their work in several different workspaces.

2.1.3 Asynchronous distributed groupware

Asynchronous distributed groupware allows distributed groups to collaborate whenever it suits each member’s schedule (Pankoke-Babatz and Syri, 1997; Manohar and Prakash, 1995). This approach frees them of the need to schedule common times to use the application, as is seen in real-time groupware applications. Information persists in the system so that it is available to users, regardless of the access time.

Most asynchronous distributed groupware systems use client/server architecture, and information about the group’s activities is stored on the server so that client applications can retrieve updates whenever it suits the user’s schedule (Pankoke-Babatz and Syri, 1997). As users interact with the client application, information is passed on to the central server so that it is available to others. This strategy is used in a number of systems including TeamRooms (Roseman and Greenberg, 1996) and GroupDesk (Fuchs et al., 1995). On a more limited scope, USENET and bulletin board systems provide a central shared space for group communication.
2.1.4 Asynchronous co-located groupware

Asynchronous co-located groupware systems support collaboration between people at a single site, but at different times. These systems provide a central location for collaboration support, and users interact with the systems when it suits their schedule.

Asynchronous co-located groupware systems are varied in their architectures and uses. For example, GeoNotes (Espinoza et al, 2001) allows users to place virtual notes that are attached to real world locations. The notes can be accessed by others when they visit that location using mobile phones and PDAs, and workers are alerted when they come into close physical proximity with a note. Dix et al. discuss argumentation tools that are used by design teams to record design decisions and arguments that led to those decisions (Dix et al., 1998). These systems are typically used at a single site, and workers commonly utilize the system asynchronously.

2.2 Mobile groupware

With recent shifts toward increased mobility in the Western workforce (Dahlbom and Ljungberg, 1998), mobile collaboration has increasingly become an important issue in CSCW. However, efforts to understand the implications that mobile work and mobile collaboration have for the design of technology are still a research subject (Alarcon et al., 2006; Aldunate et al., 2006; González et al., 2005; Guerrero et al., 2006; Perry et al., 2001). Mobile groups are highly varied in the ways they organize work, in the physical dispersion of mobile workers, and in the styles of collaboration that take place between workers (Andriessen and Vartiainen, 2006; Luff and Heath, 1998; Wiberg and Ljungberg, 1999). To help make sense of this diversity, exists efforts to describe and classify these variations by focusing on specific types of mobility (Kristoffersen and Ljungberg, 2000), types of physical distributions that occur in mobile groups (Luff and Heath, 1998), and levels of coupling between mobile collaborators (Pinelle and Gutwin, 2005; Churchill and Wakeford, 2001).

Luff and Heath consider the question of physical dispersion of workers in mobile settings, and they identified three types of mobile distributions: micro-mobility, local mobility, and remote mobility (Luff and Heath, 1998). Micro-mobility is described as the way an artifact can be moved and manipulated in a relatively circumscribed, “at hand” domain, but it is also suggested that it includes “ways of providing and receiving information whilst co-present with others”. Local mobility describes mobility around a single worksite. For example, an individual might move
between different rooms or floors in a building. Remote mobility describes individuals who move around different locations or worksites.

Remotely mobile groups differ from the other types of groups on the CSCW time-space matrix since the time and place dimensions vary depending on each worker’s location and schedule. Collaboration in these groups has many of the same problems that are encountered in stationary distributed groups (Gutwin and Greenberg, 1999; Mark, 2002). However, since place and schedules vary, it is also difficult for workers to stay aware of others’ locations and availabilities (Fagrell et al., 2000; Pinelle and Gutwin, 2005), and it can be difficult for workers to establish any type of intentional synchrony, even when technologies are utilized (Brown and Chalmers, 2003; Brown and O’Hara, 2003).

In spite of ongoing advances in mobile computing platforms and networks, technical hurdles make it difficult to develop groupware for remotely mobile groups. In groups, members often need to coordinate their activities, stay aware of others’ activities, and explicitly communicate with each other (Pinelle and Gutwin, 2005); However, the wide area wireless networks that are needed to support remote mobility are less reliable than wired networks (Satyanarayanan, 1995; Edwards and Mynatt, 1997; Edwards et al., 1997), and group interaction is often challenging to support when synchrony and timeliness of information is an issue.

For mobile workers who work across a wide area, both interference and signal strength change frequently due to changes in location as well as natural variability. Some of the direct effects are periodic disconnections, loss of data, and long delays due to congestion, retransmission, or low bandwidth. Several techniques have been offered that lessen some of these consequences under particular circumstances. Data replication (Mascolo et al., 2002a; Ratner et al., 2001) and caching increase availability of information during periods of disconnection and reduce delays. Consistency problems can be mitigated using optimistic replication schemes (Satyanarayanan, 2002), automatically resolving conflicts when they happen, and representing conflicts to the user. Adaptive strategies allow systems to make better use of their available resources, which can also lessen delay problems and help to make smooth transitions from connected and disconnected states. Although these techniques have made many mobile collaboration problems more manageable, it is still difficult to mitigate, predict, and cope with wide area mobility problems at the user, application, and infrastructure levels (Jing et al., 1999).
At the application level, mobility issues have been addressed using asynchronous groupware that allows workers to carry out their work offline since network access may only be available intermittently (Litiu and Zeitoun, 2004; Fagrell et al., 2000; Kistler and Satyanarayanan, 1992). In this approach, work is carried out on a client application that can be disconnected from a centrally accessible server, and the work is stored until a network connection is available. When network access becomes available, the client and server “synch up.” Local work is forwarded to the server so that it is available to others, and the server sends the user information about others’ activities. When stored data conflicts with changes that others have made, conflict resolution techniques may be utilized. Several systems use this approach, including Coda (Satyanarayanan et al., 1990; Kistler, 1996), Bayou (Terry et al., 1995), FieldWise (Fagrell et al., 2000).

2.3 Groupware design

Groupware designers must deal with the challenges of developing systems that support complex human-human interactions, and that fit target groups’ tasks and their social and organizational work contexts. The need to account for human-human interaction in groupware designs means that traditional design approaches are often inadequate for developing software to support groups. To address this need, groupware designers have adopted four different approaches to design: 1) incorporate social science approaches into the design process, 2) use single user design approaches that consider users and their work contexts, 3) use groupware-specific analysis and evaluation approaches, and 4) use design recommendations and frameworks based on others’ experiences.

Social science approaches. Social science theories and approaches have been used to conduct and analyze field observations, and to guide groupware design. Approaches that have been discussed in CSCW literature include: ethnography (Shapiro, 1994; Blythin et al., 1997; Hughes et al., 1994), activity theory (Collins et al., 2002; Miettinen and Hasu, 2002; Fjeld et al., 2002), and grounded theory (Grinter et al., 1999; Grinter 1998; Fitzpatrick et al., 1996).

Single user approaches. Several techniques that are used for single user development have been used to design groupware systems (Halverson, 2002). These approaches are based on field observations and on developing an understanding of users’ tasks and work settings. These include: Contextual Design (Beyer and Holtzblatt, 1998), participatory design (Greenbaum and Kyng, 1991; Muller 1991), and user centered design (Norman and Draper, 1986).
**Groupware analysis and evaluation approaches.** Several approaches have been developed for analyzing group tasks and/or evaluating the usability of groupware applications. These include: the mechanics of collaboration (Gutwin and Greenberg, 2000), groupware walkthrough (Pinelle and Gutwin, 2002), groupware task analysis (van der Veer and van Welie, 2002), collaboration usability analysis (Pinelle et al., 2003), and heuristic evaluation for groupware (Baker et al., 2002).

**Design recommendations and frameworks.** Several design recommendations and frameworks have been created to provide guidance on designing for groups that operate in specific domains or that have specific characteristics (Brown and Chalmers, 2003; Guerrero and Fuller, 2001; Luff and Heath, 1998; Lukosch and Schümmer, 2005; Neyem et al., 2006b; Pinelle and Gutwin, 2005; Scott et al., 1993). These recommendations are commonly based on the observations, experiences, and insights of developers and researchers working in the field. Lukosch and Schümmer discusses that groupware frameworks provide solutions for the development of groupware application, but have properties that complicate their usage and do not sufficiently support groupware developers (Lukosch and Schümmer, 2006). They argue that a pattern approach to support the technical aspects (or design recommendations) of groupware development serve as educational and communicative vehicle for reaching on how to design groupware applications and foster the reuse of proven solutions. Furthermore, they argue that patterns describe solutions to recurring issues in groupware development and foster the communication between developers and end-users, since they need a common language and understanding of the problem space.