Proceedings of the Second European Conference on Computer-Supported Cooperative Work
Bannon, L., Robinson, M. & Schmidt, K. (Editors)
September 25-27, 1991, Amsterdam, The Netherlands

An analysis of Design and Collaboration in a Distributed Environment

Hans Marmolin and Yngve Sundblad
IPlab, NADA, KTH, S-100 44 Stockholm, Sweden

Björn Pehrson
SICS, Box 1263, S-164 28 Kista, Sweden

The Swedish MultiG program addresses research issues in distributed multimedia workstation applications, including CSCW, and high-speed networks. This report treats some basic CSCW issues in a distributed design environment. We review and analyse relevant literature on system design and computer supported cooperation and discusses the basic issues: What is design? What is collaboration in design? What computer support is necessary for collaboration in a distributed design environment? A task analysis is performed of design and collaboration. Computer support for these tasks in a distributed environment is discussed with emphasis on generic tools for informal collaboration.

Introduction

MultiG is a Swedish cooperative research program on distributed multimedia applications and gigabit networks. It comprises projects from end-user oriented applications to very high-speed fiber optic communications. It has similar goals as the NREN gigabit network testbeds (IEEE Computer, September 1990).

The CSCW project aims at modeling computer supported cooperative design work, and specification of a distributed design environment implementing this model. It is focused on collaborative early design capture of interactive and/or embedded real time systems. The scope of the project is to propose tools for collaboration that enable designers to cooperate in a distributed environment, to specify functional, operational and interface requirements on these tools and to specify the social work situation in which these tools are to be used.

The basis for the specification of the distributed design environment is theoretical and empirical analysis of cooperative system design involving designers working at different workstations distributed geographically and/or contributing at different periods in time. Such studies should answer three questions: What is design? What is collaboration in design? What computer support is necessary for collaboration in a distributed environment? We give a first tentative answer to these questions by reviewing relevant literature on system design and computer supported cooperation and discussing different approaches to CSCW modeling such as the task and the tool approach. We summarize the findings in terms of a set of requirements on computer support for collaboration in a distributed design environment. A coming report uses these results as point of departure for modeling the CSCW environment to be used in MultiG (Marmolin et al 1991).

System design

What is system design? To answer this question we will distinguish between the design task and the design process, i.e between the activities involved in design and how these activities are performed. The design task could broadly be defined as a set of activities aiming at conceptualizing and specifying systems in accordance with the needs and requirements of the users, under existing economical and technological constraints. The design process, on the other hand, could be defined as an iterative problem solving process characterized by intuition, analysis, integration of information, and trade offs in mainly ill-defined situations.

These very broad and general definitions will be further elaborated below, but first the distinction between normative and descriptive models of design should be noted. Normative models attempt to prescribe how design should be performed, while descriptive models describe how design really is performed. Examples of normative models are top-down "structured programming" models, structured Case-methodologies etc. This report addresses only descriptive models. In our view, an understanding of the needs and the requirements on a distributed design environment demands descriptive models, that have to be based both on theoretical analysis of man´s basic capabilities to accomplish the design task and on empirical findings concerning how this task is performed today. The reason for this combined approach is that design is not a static process, but changes continuously over time as new tools and methods are developed. Specific empirical findings are then valid only for a short period of time.

The design task

There are a lot of attempts to analyse the design task into subtasks and subsubtasks or activities, see e.g Rouse and Boff (1987a). They differ in details, but they all agree that the design task includes the set of generic subtasks listed below, although the order of subtasks may differ.

- *Formulation and identification of the design problem*
  Needs, requirements and constraints are identified and the design problem is specified.
- *Understanding the problem*
  Information about the problem and possible solutions is gathered.
- *Generation of alternative solutions*
  Design options are generated and synthesized.
- *Selection and interpretation of alternatives*
  The impact of different design alternatives on the users needs are analysed and evaluated, trade offs and optimization's are made and the most acceptable alternative is selected for implementation.

Rouse and Boff (1987b) suggest that these design subtasks could be further divided into a set of activities. However, their classification is very general, as it is intended to be valid for all kinds of design tasks. A more specific list of activities could be suggested if one limits the scope to design of software systems and to the early stages of this process. Software design can be described as a mapping of the behaviour required of the application (system function, constraints, exception conditions, user actions, etc) on to the computational structure implementing this behaviour (control structures, computer architectures, data structures, algorithms, etc) using knowledge about the application domain and the software domain (Curtis et al 1988). We will use this definition as a basis for our classification of design activities, but we will integrate it with the classification proposed by Rouse and Boff and the result of some empirical studies of software design (Curtis et al 1988, Rosson et al 1988, Meister 1987). We propose that the subtasks listed above are mainly composed of the activities listed in Table I.

Subtask	Activities
Formulation of the design problem	Defining and decomposing the problem Formulating requirements, criteria and constraints Planning and coordinating team activities
Understanding the problem	Learning about the application Mapping application knowledge onto computational structures Building a mental model of the system
Generation of alternative solutions	Reviewing other attempts to solve similar problems Analogizing from earlier experiences Using intuition
Selection of alternatives	Prototyping Logical and/or empirical evaluation, design validation Advocating and reporting the chosen alternative

Table I. A classification of design activities

Many of these activities could be performed either individually or in collaboration. This question will be discussed in detail below. Coordinating team activities concerns planning activities, division of work etc and this activity exists of course only in project teams. The activity learning about the application is composed of many different subactivities such as seeking information, analysing information, integrating information obtained by user interviews, user observa-

tions, task scenarios etc. The activity building a mental model of the system is a process that goes on during the whole design phase. The term mental model stands for the way humans integrate new knowledge and earlier experiences in the form of an incomplete, unclear and unspecific but concrete model of the objects and events in question, see e.g. Norman (1983). It functions as a sort of outline of what the designer must do to solve the design problem and what he has accomplished so far (Meister 1987). Reviewing other attempts means to use different kinds of external sources as e.g other systems, literature, research reports etc to get ideas. The activity prototyping means all kinds of prototypes, paper mock-ups etc. Protoyping can of course be used also for understanding the problem and for generation of alternative solutions. Logical evaluation means the use of different kinds of tools for specification and analysis, such as formal languages, simulations etc and empirical evaluation means studies of user performance and satisfaction.

We will use this description as a first tentative model of the design task. Of course, for each design task there will be a mix of the activities listed. Some of the activities may be unconscious, some may be skipped or truncated and some may not be accomplished. In general, however, we will assume that the design task consists of the activities listed above. Although this assumption is based on both theoretical and empirical studies, it has of course to be empirically tested in relevant situations. An on-going study within our project is concerned with this problem.

The design process

As mentioned, the list of subtasks given above does not mean that the design process is a structured process starting with problem formulation, ending with the selection of the best alternative. On the contrary the design process as a whole and each subtask and activity is best described as an unstructured process that varies from designer to designer and from time to time depending on a lot of mainly unknown factors. Thus, it is not possible to describe the design process in terms of a given flow of work. Instead we propose a description in terms of a set of important and interrelated dimensions of the design process.

Design as problem solving

One school of thought describes design as a top-down decomposition of objectives into requirements and physical processes, with cost-performance trade offs. This constitutes what Rouse and Boff (1987b) call the analytic view of design. This view assumes that design is a structured and ordered organized analytic process. Other have observed design as a mixture of top-down and bottom-up processes as an incremental approach. Design is assumed to emerge from bottom-up perceptions of patterns of understanding and experiences of users´ need. This could be called the artistic view of design, characterised as organic growth (Sandewall 1978).

However, design could rather be viewed as a form of problem solving (Rouse 1986). Research in human problem solving (Rouse 1983) indicates that humans approach problems on several levels at the same time moving among recognition, planning and execution activities. Pattern recognition on contextual surface features

seems then to be more important than analysis of problem structure. Thus problem representation is of fundamental importance for design. As most design problems are ill-defined at early stages, the goals are unclear and goal clarification occur in parallel with search for solutions, the analytic and artistic approach have to support each other (Klein 1987). That means, that the design process has to start with some definition of the problem, finding tentative solutions, clarifying the problem using these solutions, finding new solutions etc. The design process could then be described as a series of information transactions (Bally 1987). Partial information from the designer's mind is used as the basis for a first sketch. By externalizing this sketch, the designer can retrieve and accumulate additional information from the task environment as a basis for the next externalization. Design is then an iterative mixture of top-down or bottom-up processes.

For example, Rosson et al (1988) studying 22 design teams at different stages including both business and research teams, found that there were two dominant approaches to design, a phased development approach and an incremental one. The phased approach was characterized by a separation of design, implementation and evaluation, while in the incremental approach these steps occurred simultaneously. The incremental approach was mainly used in research projects, by small teams and in an interpretating programming environment, while the opposite was true for the phased approach. It should be noted that user testing was applied to the same degree in both approaches. Curtis et al (1988) found in a field study of large system design an important reason for applying a mixture of bottom-up and top-down processes: requirements always change during the design process as a result of different and changing needs of the customers, changes in underlying technology, misunderstandings of the application domain and unrequired enhance-ments added by programmers. Although design teams tried to solve this problem by negotiations, it was often difficult to enforce agreement across teams. Thus requirements were not as stable reference for a top-down approach as often assumed. Another possible reason for the different approaches to design can be attributed to individual differences between designers (Rouse 1986). Designers like all humans are different and apply different cognitive strategies to problem solving as serialistic or holistic thinking, impulsive divergent or reflective convergent thinking, etc. Nadler (1984) finds four types of designers, the inactivists that avoid problems, the reactivists that emphasize well proven solutions, the preactivists that designs for the future and the interactivists that emphasize designing of the future.

Design as intuition

Smith (1987) argues that intuition is a fundamental part of design (while others emphasize analytic aspects) and distinguishes two types of intuition of importance for design, generational and judgemental. The former concerns intuitive ideas of new concepts, the latter concerns evaluative judgements about proposed solutions.

Concrete representations play a very important role in intuition (Rouse 1986). As mentioned above the basis for many design decisions are informal experiments as e.g. rapid prototyping, analysis of analogous situations and earlier solutions, concrete representations of alternative solutions such as scenario descriptions, graphical

representations, dimensional representations etc. Bally (1987) studying the use of different kinds of representations in design using a hypothetical design task[1], found that the designers use many different representation but one at a time, going back and forth between them and that visual representations were predominant.

The use of concrete representation could be interpreted as an attempt of designers to build a mental model of the system. More seldom is a formal approach applied, in which the designers attempt to find the optimum solution using some decision theory or decision methodology. In this context it should be noted that some argue that design is a question of maximizing system effectiveness within the given constraints, while others view design as a question of producing an acceptable, but not optimal solution (Rouse 1987b). This distinction is important as if the search is not for an optimum or constrained optimal solution, then design support systems based on finding an optimal solution will not be very useful. We will argue that at least in the early stages, the design process could not be described as an optimization process, although later stages may focus on a search for optimal solutions.

Design as information gathering

Rosson et al (1988) investigated how design ideas were obtained and evaluated and found that most ideas were obtained by information gathering techniques such as task/user analysis, analysis of other systems, literature reviews etc. The techniques most used for evaluating ideas were however not the same (see Table II).

Activity	Getting ideas	Testing ideas
Information gathering	53 %	7 %
Creative thinking	16 %	9 %
Group discussions	13 %	14 %
Logical analysis	12 %	46 %
Prototyping	6 %	25 %

Table II. The activities used for getting and testing design ideas[2].

Another important aspect of information gathering concerns the integration of knowledge from different sources into an unified view. This integration could be described as a learning process (Curtis et al 1988). Designers continuously learn about the application domain, about new computational methodologies and about design and implementation decisions made by others. Also the customers undergo a learning process as they begin to understand the implications of their requirements. Curtis found that, although application domain knowledge was necessary for successful design, this knowledge was split among the software development staff. The ability to integrate such knowledge into a unified view and transform it to computational structures characterized the exceptionally good designers they met.

[1]The task was to design a product that allows the payment of credit card bills from home using the telephone.

[2]The figures given represents the percentage of projects in which designers reported the use of a certain technique. The basis for these figures is 21 projects selected by the designers, representing a wide variety of applications and system size. There were seven research projects, seven projects were concerned with site support and seven with product development.

Design as a collaborative process

Many argue that design is collaborative work. Harrison et al (1990) view design as a social construction of a technical reality focused on establishing and maintaining a shared understanding among the participants. Bødker et al (1987) describe design in terms of mutual learning in a design team of computer professionals and end users. Others view design as negotiations. Curtis et al (1988) found that negotiations about requirements and trade off solutions occur throughout the development process. Rosson et al (1988) found that these aspects were the most important and that communication and coordination is especially critical for teams using the incremental approach where system changes occur continuously and unpredictably. Kedzierski (1988) studied the activities of software designers during evolutionary development of a compiler and found that especially during later design stages most time was spent on communication activities as shown in Table III. Norcio et al (1986) conclude from a study of design activities in developing complex software modules, that discussion activities among software designers play a major role in the design process and indicate design progress. Curtis et al found, however, communications outside the team and across organizational levels to be rare and that communication problems often occurred when groups transfer inter-mediate work products and also found documentation to be very ineffective for communication. Instead each team member had several nets of people to talk to for information on issues affecting their work. Sathi et al (1988) point at another communication problem, the problem of identifying the team members that should be informed about changes made to design (in large number) at various stages of development.

Activity	Time spent
Questions to other designers	27 %
Information about changes	25 %
Complaining	13 %
Planning work	14 %
Testing commands, functions	21 %

Table III. The activities if software designers during evolutionary development[3].

Conclusions

This review of descriptive theories and studies of design points to the difficulties in trying to model this very unstructured and complex process in any structured way. The lesson to be learned is instead that any computer support has to be as flexible as the design process itself. However, it points to some important characteristics of the design task and the design process that have to be considered in developing a distributed design environment for early stages of design.

[3]The figures given represents percentage time spent of each activity. The study was based on a small sample of data recorded when the designers used the system they were developing.

Firstly we argue that design is an iterative process in which each activity is characterized by a mixture of analytic, structured, linear, artistic, chaotic and nonlinear behaviour, depending on among other things the design phase, the state of the problem, the size of the design team, the size and kind of the design task etc. Bottom-up approaches seem to be more dominant in earlier stages, in small research oriented design tasks, when the problem is ill-defined, the design teams are small and prototyping is used. Although support for bottom-up processes seems to be very important in the distributed environment that is the concern of this study, both analytic and artistic design have to be supported. More important, the design tools should not be based on any assumptions about how the designers work. That is, they should not impose any restrictions on the design process and they should be available at any time during design.

Secondly, we will assume that earlier stages of design are characterized by intuitive information gathering processes rather than by formal analytic processes and that concrete representations play an important role for understanding and evaluating design ideas. Thus, the support given has to focus on informal cooperation. In addition tools for idea generation as story board facilities and facilities for observing other system, and tools for visualizing and describing ideas are often more valuable than analytic tools.

Thirdly, we adopt the view held by Curtis et al (1988) that good design is characterized by the ability to integrate knowledge into an unified view and transform it into computational structures. The distributed environment has to support integration of knowledge by learning and development of a common frame of reference.

Finally, we regard design as collaborative work. This means that a distributed design environment cannot function without support for coordination, cooperation and communication. Both Curtis et al (1988) and Rosson et al (1988) point to the need for informal and formal collaboration tools for change facilitation, record keeping of ideas and design concepts, for information sharing etc. As collaboration is essential for design ahis support has to be very effective and so easy to use that it does not interfere with the design activities themselves.

Collaboration in distributed design

What is collaboration? As discussed above collaboration is one of the most important components of the design process. Collaboration could be viewed from many different perspectives. Our perspective could best be described as an activity or task perspective, i.e we will focus our analysis on the different activities of collaboration during design and the needs for support that these activities demand in a distributed environment. However, we will start with a more general discussion of some important characteristics of the collaborative process.

The collaborative process

Of fundamental importance for designing usable computerized tools for cooperative work is an understanding of the collaborative process. This section attempts to give

a basis for such an understanding by describing central characteristics of the collaborative process and demands that these characteristics put on the computer support.

Collaboration as a social process

Perhaps the most important aspect of collaboration is that it is a social process, controlled by social conventions as Kraut et al (1986) concluded from their study. They interviewed 50 research teams and concluded that the most important aspect of collaboration was the establishment and maintaining of personal relationships. These form the glue that holds together the pieces of collaborative efforts, but also the source of many problems in collaboration. They pointed to the importance of geographical proximity for the development of personal relations and trust, which is crucial for collaborative work. Also Harrison et al (1990) emphasize the social aspects of collaboration. They point out that each participant in a design group becomes part of the group and must maintain working relationship with it throughout the design process and that these social processes constitute the basis for all the negotiations, commitments and responsibilities that control the design process.

Another related aspect of collaboration concerns the establishment of a common frame of reference. Each team member perceives the goals and the design problems differently depending on their knowledge and interests. In order to reach consensus, social processes focused on an understanding of each partner´s real beliefs and motives are necessary. The development of a common framework can also include more formal processes as when a reference model for a project is established. However, the problems with developing a framework is more often related to social factors, than to formal ones. The establishment of a common frame of reference is a necessary base for communication and for the interpretation and integration of information especially in multi-disciplinary design teams where much valuable information is cross disciplinary. Particularly during early phases teams spend considerable time defining terms and common views.

A common frame of reference is usually obtained by having a series of meetings where each partner describe his/her view on the problem. In a distributed environment this could be realized by some kind of electronic meeting room as proposed by Begeman et al (1986).

Collaboration as a communicative process

Another important aspect of collaboration is that it is a communicative process. For example, Johnson (1989) views collaboration as a communication process and argues that the characteristics of human collaboration can be abstracted from examinations of conversations, especially from breakdowns in conversations.

In a distributed environment collaboration has to be accomplished by communication. In any human communication process there are social rules that monitor the communication pattern in terms of social acts like persuasion, negotiation, arguing. In ordinary communication these rules are learned and signaled by a metacommunication language based on gestures, intonation etc. As Danielson et al (1986) point out, in electronic communication, the development of new rules and metacommunication tools are necessary for enabling the participants to monitor the

communication process. Winograds Speech Act model (1988) is one example of how conversational structures can be built into a message system. However careful thought should then be given about the effects of the imposed structure on social control. As argued by Harrison et al (1990), communication in design is as ambiguous as the design process itself and ambiguity is a common and healthy characteristic of communication in a design group, a precondition for creativity.

Collaboration as information sharing

Whenever people work together they must share information. In this context we will make a distinction between sharing knowledge and experience among team members as when one asks a team member about some facts, sharing design information and design results as e.g. when one subroutine is passed to another team member, and sharing different sources of background information as research reports, system descriptions etc.

As found by Kedziersky (1988) questions to other designers are an important way of sharing information. In a distributed environment, this points to the need of electronic message systems specially designed to support the search for advice. Information about changes could be supported by some kind of recording device as proposed by Rosson et al (1988). Such recording devices should not only record the decided changes as caused by new requirements, but also the reasons for these changes. Background and design information, relevant research literature and information about similar systems has to be shared and discussed by the team members. This points to the need of a common knowledge basis. However, as documentation is not enough as found by Curtis et al (1988), the knowledge base should also contain information about "who knows what".

Collaboration as knowledge integration

Collaboration could also be viewed as a process of knowledge integration. Integration of knowledge and experience among team members is obtained by collaborative idea generation through discussions and brain stormings etc.

Idea generation refers to activities related to the creation, development, and testing of ideas and proposed solutions to design problems. In an ordinary environment, new ideas are created, developed and tested in mainly informal situations. In a distributed environment this could be supported by electronic multimedia whiteboards in which ideas can be visualized and discussed as in Colab (Stefik et al 1988). Although the study by Rosson et al (1988) did not find group discussions to be very important for getting and testing ideas, we argue that ideas are often informally generated and tested by discussion and explanation o other team members.

Collaboration as co-working

Many emphasize the co-working aspects of collaboration. To support synchronous cooperative work execution such as co-editing or asynchronous such as reviewing, annotating etc have been a challenge for many CSCW projects.

However, with respect to this aspect of the team work, Kraut et al (1988) found that the research teams in their study developed work strategies that reduced the

need for co-working and simplified the integration of work results. Examples of such work strategies were division of labour which turns joint tasks into individual ones, encapsulation of subtasks which reduced the information sharing to the sharing of interfaces, and sequential processing which minimizes the need for information sharing to information about the final product. Thus, Kraut et al's study (1986) points to that the preferred work strategy in collaborative work is to avoid working together, i.e to decrease what we will call *the collaboration load.* If this is an effect of basic human capabilities or an effect of missing tools for cooperation could be discussed. However, a lot of groupware, as tools for co-authoring, co-editing, co-drawing (see e.g Beaudouin-Lafon 1990) are built on the assumption that people really want to synchronously accomplish tasks together. A more plausible assumption could be that information sharing is more important for collaboration than to work together on the same task at the same time. This could at least be true for professional routine tasks such as authoring, coding, drawing etc, although it may not hold for highly creative tasks such as problem solving.

Collaboration as management

Dhar and Olson (1989), emphasize the problem solving characteristics of management activities in collaboration, such as planning, monitoring, negotiation, scheduling and decision making. Planning concerns with the coordination of the activities to be performed, which often involves negotiations about commitments. Monitoring concerns decisions about how to achieve the goals.

However, Bally (1987) found that although critical for success, planning and other management activities constitute a small part of the design process (≈ 5%). Most of the efforts were devoted to routine activities as the use of well-rehearsed professional skills. The importance of management activities depend of course on the phase of the design process and the size of the design team. In planning phases and in large design groups management activities will always play an important role. In a distributed environment such activities could be supported by electronic calendars, records of commitments and cooperative project planning tools. applied Such a perspective has been used in cooperative environments for distributed design and development of software by Kurbel & Pietsch (1990), for project management by Bhandary & Croft (1990) and by Sahti et al (1988).

Generic collaborative tasks and tools

The analysis of collaboration presented above could be used as a point of departure for a classification of collaborative tasks in a distributed environment. We will distinguish between the conference task, the coworking task, the information exchange task and the management task. It should be noted that the list presented here is neither intended to be complete nor to be final. It will change as more experience of distributed design is gathered. It represents a first tentative classification based on the studies reviewed above.

The conference task could be defined as any discussion exchange of experience and knowledge between two or more team members. Such discussions could have the form of negotiations, idea generation, problem solving, briefings etc. The task

could be asynchronous or synchronous, formal or informal. In a distributed environment it could be supported by electronic conference systems, mail systems etc. Important characteristics of the collaborative process in this task are the social and communicative characteristics and the aspects concerning knowledge integration discussed above. *The co-working task* concerns any activity for synchronous or asynchronous collaborative production of some document or other kind of product. This task could be supported by distributed applications as co-editors, co-authoring and annotating systems etc. *The information exchange task* could be defined as any activity concerning the exchange of documents and other kind of information between two or more team members. In a distributed environment it could be supported by shared databases, hypertext libraries, record keeping tools and other forms of group memories. *The management task* consists of any activities aiming at coordinating and supervising the collaboration within a team. It includes such activities as planning, scheduling etc.

These tasks can be mapped onto the different design tasks listed in Table I. There is of course no perfect mapping, but it is possible to describe the main collaborative tasks during each design task as shown in Table IV

Design Subtask	**Collaborative task**			
	Conference	Co-working	Information	Management
Formulation of the problem	1	4	2	3
Understanding the problem	2	4	1	3
Generation of solutions	2	4	1	3
Selection of alternatives	1	2	4	3

Table IV. The relation between design tasks and collaborative tasks[4].

At this stage of research, Table IV should just be regarded as a set of hypothesises, that have to be tested empirically. According to this table the most important collaborative tasks during design are the conference task and the information exchange task, while less important are the co-working and the management task.

Groupware supporting these tasks could be designed for support of the specific needs of distributed design. However, one then has to model these tasks and build in assumptions about how the task is performed by the users. As little is known about social processes, such models will not be very valid as shown by the many CSCW systems that have failed to be useful. In addition, one then creates a system controlled design environment instead of a user controlled and this is especially dangerous for early design phases that are not very formalized but creative and artistic. To put the user in control means that systems should be designed as toolboxes, as a set of "independent" and powerful tools that the users control and use according to their ideas of how to accomplish the tasks (Bødker et al 1987, Bødker 1989). We therefore argue for a tool-based approach as opposed to the task-based discussed above. Such an approach aims at designing an user controled environment that facilitate for the users to do what they want, without limitations

[4]The figures indicate the importance, in order of priority, of being able to accomplish a certain collaborative task during a certain design task in a distributed environment.

and assumptions imposed by the system (Schneiderman 1989). Instead of designing collaborative tools based on some analysis of the design task or collaboration task to be fulfilled, one could attempt to design very generic collaborative tools (as generic as the telephone), such as the shared folder, the shared window, the video window etc, that the users can use and combine as they want in order to accomplish the collaborative design tasks. Some combination of tools could be used for conferencing, some other for co-working, some for understanding the problem and some other for selection of alternatives etc. In some situations could the tools be used for formal in other for informal collaboration, in some for one-to-one collaboration in other for many-to many collaboration etc. However this set of tools should be designed as an integrated environment. Tools for collaborative work can be isolated both with respect to other collaborative tools and with respect to other application used by the users. Three aspects of integration are important, i.e. user interface integration, flow of control and flow of data. User interfaces to the collaborative tools have to be integrated into the user's desktop in a consistent way, it must be possible to access functions in other tools from any tool, and one must be able to transfer the results from using one tool to any other tool.

This design paradigm is similar to the paradigm of the Workaday World proposed by Moran and Anderson (1990) as it is not task oriented, but focuses on the social process of collaboration and on giving the users tailorable tools that they can control and attend to according to their needs. According to Moran and Anderson (1990) these tools should not only support the users, they should enhance and encourage people in their work and allow creative deployment and development of job skills. The first problem is then to identify a sufficient and necessary set of basic tools for collaboration. Next each tool has to be designed so that it can be used for different kinds of collaborative tasks in a distributed design environment.

Conclusions

This discussion of collaboration points at some important characteristics that have to be considered when designing groupware for a distributed design environment.

Firstly, the most important form of collaboration in a distributed design environment seems to be informal collaboration. There is a lot of evidence for the importance of informal collaboration in design, especially in early stages (see e.g. Weinberg 1971, Kraut et al 1986). Thus, although both formal and informal collaboration have to be supported in a distributed environment, support for informal cooperation seems to be more important as concluded also in the discussion of the design task. We will assume that the most common form of collaboration in a distributed environment will be asynchronous and synchronous collaboration one toone, but there will also be a need for many-to-many collaboration.

Secondly, there are a lot of different collaborative tasks that have to be supported in a distributed design environment. A set of groupware applications thus has to be designed, where each application is adapted to the characteristics of the corresponding collaborative task. Another solution is to design a few generic tools for collaboration that can be used in different ways. Which approach is to be preferred is an open question that should be further studied, but we will adopt the latter

approach. One easons is our belief that a design paradigm as the one discussed by Moran and Anderson (1990) is necessary for creating a user controlled, creative, flexible and tailorable collaborative environment. However, we will use this analysis of the design task and the different collaborative tasks and the results of other on-going empirical studies as a basis for identifying the necessary and sufficient set of generic tools and for specification of the requirements on these tools. In addition we investigate the possibilities to tailor these tools to the more specific needs of the design situation as e.g. the need for some device for recording and communicating design changes, commitments, the need for specific tools for co-working, division of work etc.

Thirdly it can be concluded that computer support for cooperative work should not only facilitate task accomplishment but also support social processes as productive personal relationships, negotiation and the development of a common frame of reference. In addition electronic communication should not only support the transfer of messages, but also the monitoring of the communication pattern by a metacommunication language. Such a language should be able to transfer social communication signals such as gestures.

Finally this discussion indicates that it is not as important to support management activities and collaborative task accomplishment as information sharing and knowledge integration. However, support for the three work strategies for reducing collaboration load discussed above may be of great importance, i.e support for division and integration of work, encapsulation and sequential processing.

General conclusions

What computer support is necessary for collaboration in a distributed environment? We have proposed a classification of generic collaborative tasks that could be used as a basis for the development of groupware applications supporting distributed design. However, we argue that a more tool oriented approach should be tested first. The next step in our project will be to specify in detail the functional, operational and interface requirements on these tools and to model the social work situation in which these tools are to be used. Our analysis of design and collaboration can be summarized in terms of the following set of general functional requirements on groupware support for design in a distributed environment. They are listed in a tentative order of priority.

Support informal collaboration. As discussed above this is perhaps the most important requirement on a distributed design environment, but it is also the one that will be hardest to fulfil. This requirement means that the environment had to support communication of social behaviour patterns, establishment and development of personal relations, drop in meetings etc.

Support sharing and record keeping of design information. As put forward by many researchers in this area, there is really a need for supporting sharing and record keeping of important design information especially in larger teams. Thus a distributed design environment should support record keeping and sharing of

requirement-, design- and implementation changes, design results, design ideas and design concepts, commitments, work plans etc.

Support sharing of background knowledge. This requirement points to other important preconditions for cooperation, namely the need for a common frame of reference, for sharing application domain knowledge and for exchanging knowledge about similar systems and other solutions to the design problem.

Support presentations of ideas. As concrete representations are very important in design, the distributed environment has to support different ways of presenting and visualizing ideas for other team members. This could be done different visualization tools, by story board facilities etc.

Support strategies reducing the need for co-working. It may be more important to have efficient tools for reducing the "collaborative load" than tools supporting co-working. Thus one has to support division and integration of work, encapsulation and sequential processing.

Support co-working. Although one supports the strategies mentioned above, there will always be a need for co-working. Especially asynchronous co-working should be supported by annotating and reviewing systems. To support synchronous co-working does not seem to be as important, except for support for distributed interface design together with end-users at other places. For management activities, we will not propose any support, as other projects have shown that tools like electronic calenders etc are not very useful (Grudin and Poltrock 1990).

Although we are concerned with a distributed design environment, we do not assume that groupware will be a substitute for all face-to-face meetings. Galegher argues (1990) that complex collaborative work involves a continuing need for face-to-face meetings, especially in initiating and planning of the collaborative work. We believe that a well designed distributed environment can both reduce the need for face-to-face meetings and offer new and more effective ways of collaboration.

References

Bally J.(1987): "An experimental view of the design process", in W. Rouse and K. Boff (eds.): *System Design*, Elsevier Science Publishing Co, New York, 1987, pp. 65-83.

Beaudouin-Lafon B.(1990): "Cooperative Development of Software", in S. Gibbs and A.Verrijn-Stuart (eds.): *Multi-User Interfaces and Applications*. North-Holland, 1990, pp. 103-115.

Begeman M., Cook P., Ellis C., Graf M., Rein G., Smith T.(1986): "Project Nick: Meetings Augmentation and Analyses", *Proceedings of CSCW'86*, Dec 1986, pp. 1-6.

Bhandary N., Croft B. (1990): "An Architecture for Supporting Goal-Based Coooperative Work", in S. Gibbs and A.Verrijn-Stuart (eds.): *Multi-User Interfaces and Applications*, North-Holland, 1990, pp. 337-355.

Bødker S.: "A Human Activity Approach to User Interfaces". *HCI*, vol. 4, no. 3, 1989.

Bødker S., Ehn P., Kyng M., Kammersgaard J., Sundblad Y.(1987): "A Utopian Experience", in G.Bjerknes , P.Ehn, and M.Kyng (eds.): *Computer and Democracy*, Avebury, Aldershot, 1987.

Curtis B, Krasner H, and Iscoe N.(1988): "A field study of the software design process for large systems", *Communications of the ACM*, Vol 31, no. 11, November 1988.

Danielsen T., Pankoke-Babatz U., Prinz W., Patel A., Pays P., Smaaland K., Speth R.(1986): "The AMIGO project", *Proceedings of CSCW'86*, Dec 1986, pp. 229-246.

Dhar V., Olson M.(1989): "Assumptions underlying systems that support work group collaboration", in M. Olson (ed.): *Technological support for work group collaboration*, Lawrence Erlbaum Ass, Norwood, 1989, pp 33-51.

Galegher J.(1990): "Computer-Mediated Communication for Intellectual Teamwork: A field experiment in group writing", *Proceedings of CSCW'90,* October 1990. pp. 65-78.
Grudin J., Poltrock S.(1990): *Computer supported cooperative work and groupware.* Tutorial at CHI'90, Seattle, April 1990.
Harrison S., Minneman S., Stults B., Weber K.(1990): "Video: A design medium", *SIGCHI Bullentin*, vol, 21, no, 3, January 1990
Ishii H.(1990): "TeamWorkStation: Towards a seamless shared workspace", *Proceedings of CSCW'90,* October 1990, pp. 13-26.
Johansen R.(1989): "User approaches to computer supported teams", in M. Olson (ed.): *Technological support for work group collaboration,* Lawrence Erlbaum Ass, Norwood 1989, pp.1-33.
Johnson B.(1989):"How is work coordinated?Implications for computer-based support", M Olson (ed.):*Technological support for work group collaboration,* Lawrence Erlbaum1989,pp. 51-65.
Kedzierski B.(1988): "Communication and management support in system development environments", in I. Greif(ed.):*Computer-Supported Cooperative Work,* M Kaufman,1988,pp.253-269.
Klein G.(1987): "Analytical versus regognitional approaches to design decision making", in W. Rouse and K. Boff (eds.): *System Design*, Elsevier, New York, 1987, pp. 175-187.
Kraut R., Galegher J., Egido C.(1986): "Relationships and tasks in scientific research collaborations", *Proceedings of CSCW'86,* Dec 1986, pp. 229-246.
Kurbel K., Pietsch W.(1990): "A Cooperative Work Environment for Evolutionary Software Development", in S. Gibbs and A.A Verrijn-Stuart (eds.): *Multi-User Interfaces and Applications.* Elsevier Science Publisher, North-Holland , 1990, pp. 115-131.
Marmolin H., Sundblad Y., Pehrson B.(1991): "TheKnowledgeNet - An Environment for Distributed Design.", IPLab-MultiG Technical Report, 1991. (Forthcoming)
Meister D.(1987): "A cognitive theory of design and requirements for a behavioural design aid", in W. Rouse and K. Boff (eds.): *System Design*, Elsevier, New York, 1987, pp. 29-245.
Moran T. And erson R.(1990): "The Workaday World As a Paradigm for CSCW Design", *Proceedings of CSCW'90,* October 1990, pp. 381-393.
Nadler G.(1984): "System methodology and design", *Proceeding of 1984 IEEE System, Man and Cybernetics Conference,* Halifax, Nova Scotia,, October 1984, pp. 427-437.
Norcio A., Chmura L.(1986): "Design activity in developing modules for complex software.", in B.Soloway and S.Lyenger (eds.): *Empirical studies of programmers*, Ablex , 1986, pp. 99-117.
Norman D.(1983): "Some observations on mental models", in: D. Gentner and A.L. Stevens (eds.): *Mental models.* Hillsdale, New Yersey: Lawrence Erlbaum Associates Inc, 1983.
Rosson M, Maass S., Kellogg W.(1988): "The designer as user: Building requirements for design tools from design practice",*Communications of the ACM,* Vol 31, no. 11, November 1988.
Rouse W., Boff K. (1987a): "Workshop themes and issues: The psychology of system design.", in W.Rouse and K.Boff (eds.): *System Design*, Elsevier, New York, 1987, pp. 7-19.
Rouse W., Boff K. (1987b): "Designers, Tools, and Environments.", in W. Rouse and K. Boff (eds.): *System Design*, Elsevier, New York, 1987, pp. 43-65.
Rouse, W.(1983): "Models of human problem solving: detection, diagnosis and compensation for system failure", *Automatica*, vol. 19, pp. 613-625, 1983
Rouse, W.(1986): "On the value of information in system design: A framework for understanding and aiding designers",*Information Processing and Management*, vol.22, no.2,pp.217-228, 1986
Sandewall E.(1978): "Programming in the Interactive Environment: The Lisp Experience", *ACM Computing Surveys*, vol. 10, no. 1, 1978.
Sathi A, Morton T, Roth S.(1988): "Castillo: An intelligent project management system", in I. Greif (ed.):*Computer-Supported Cooperative Work.* M Kaufman, San Mateo,1988,pp.269-311.
Schneiderman B., Kearsley G: *Hypertext Hands On,* Reading, MA: Addison-Wesley, 1989.
Smith J.(1987): "Intuition by design.", in W. Rouse and K. Boff (eds.): *System Design*, Elsevier , New York, 1987, pp 305-319.
Stefik M., Foster G., Bobrow D., Kahn D., Lanning S., Suchman L.(1988): "Beyond the Chalkboard: Computer support for collaboration and problem solving in meetings", in I. Greif (ed): *Computer-supported cooperative work.* M Kaufman, San Mateo, 1988, pp. 335-367.
Weinberg G.(1971): *The Psychology of Computer Programming,* chapter 4-5, Van Nostrand Reinhold Company, New York, 1971.
Winograd T.(1988): "A language perspective on the design of cooperative work", in I. Greif (ed.): *Computer-supported cooperative work.* M Kaufman , San Mateo, 1988, pp. 623-765.