

Computer Supported Cooperative Work: New Challenges to Systems Design

Peter H. Carstensen and Kjeld Schmidt

Center for Tele-Information, Technical University of Denmark

“Collaborative work is the core of our society, wrought with difficulties and benefits. It is clear that technology can change group work, and there is a good possibility that it can result in major enhancements to productivity. But, there is a lot of work to do before we fully understand how to accomplish that. Trial and error from creative system builders is too slow a discovery process. What is required is a better understanding of the nature of group work, the extent of the possibilities of the design space of technology features, and evaluation of systems in use that leads to a theory of computer supported co-operative work, which in turn can help us direct subsequent invention of new ways to do group work” (Olson *et al.*, 1993).

1. Introduction

A general trend in modern work settings seems to be that the work becomes more and more complex. Complex in the sense that it is characterized by complex problem solving and decision making activities, rule interpretation, cooperative work processes, etc. The demands for flexibility, production time, complexity in products, etc. — as it is for example reflected in the trends of ‘concurrent engineering’ (Helander and Nagamachi, 1992) — are increasing. Also the structure of the context of most work settings are becoming more and more differentiated, and is characterized by an increase in the speed of the changes (Ciborra, 1993). Many activities and situations to be handled in modern work settings often have an inescapable aspect of contingency (Suchman, 1987).

So, despite an increased level of automation within many work settings more and more work activities become complex human activities involving many people often having different areas of competence. Information from many sources has to be considered and decisions have to be made by mutually interdependent actors. Cooperative work arrangements emerge in response to different requirements and must serve different generic functions such as augmentation of capacity, differentiation and combination of specialities and techniques, mutual critical assessment, and combination of perspectives (Schmidt, 1994). When the number of people involved in work activities exceeds the limit

of a few, they need to examine the state of affairs in the field of work, for example seek answers for “when will sub-component XYZ be tested, and by whom?” or “which activities is John allocated for within the next four hours?”. Actors involved in cooperative activities will need to coordinate their activities, including, for example, meshing, allocating and scheduling others activities, actors and resources (Strauss, 1985; Schmidt, 1994). When work requires the handling of a multitude of intertwined and interdependent activities, the complexity of coordinating these activities increases tremendously.

The research field of CSCW (Computer Supported Cooperative Work) addresses how collaborative activities and their coordination can be supported by means of computer systems. A series of questions becomes central, for example: What characterizes cooperative work?; How can we model cooperative work?; Which computer based facilities should be provided?; And what are the basic characteristics of useful platforms for CSCW-systems? Computer-based support for cooperative work can be provided by offering better communication facilities, providing improved monitoring and awareness possibilities to the actors, and by aiming at reducing the complexity of the coordination activities to be conducted by the involved actors.

This paper aims at introducing some of the essential problems addressed within CSCW, and to illustrate some of the approaches taken for better understanding the central aspects of cooperative work activities and to design computer-based systems supporting collaborative work activities. Our discussion and examples are to a large extent taken from design oriented work domains. Most of the claims will, however, be generally valid.

2. CSCW as a research field

As a research field, the field of CSCW (Computer Supported Cooperative Work) is relatively new, and a generally accepted definition has not yet been established (cf. Bannon, 1993). There is still a lot of confusion and debate of which overall research questions should be addressed. This chapter is by no means an attempt to solve this. We will present what we consider the most relevant dimensions for characterizing the field, and briefly introduce the approach taken by some of the central players in these dimensions.

The literature contains, of course, many more or less different definitions of what CSCW is, what groupware is, what characterizes groupware technology, etc. Since many publications in the CSCW field include their own definitions it is more or less impossible to provide an overview. It exceeds the scope of this chapter to discuss these, but a brief introduction to some is relevant:

According to Grudin (1991) groupware, or CSCW systems, is defined very differently by different researchers. Some will consider technology providing access to shared files as CSCW systems (Crowley in Ensor, 1990); others will consider e-mail systems as CSCW systems (e.g., Kraut in Ensor, 1990); whereas for ex-

ample Bannon and Schmidt (1989) have argued, that CSCW systems must be based upon, and reflect, an understanding of the cooperative aspects of the work to be supported, i.e., e-mail that do not recognize any roles beyond the receiver and the sender is not to be considered a CSCW system. E-mail and facilities providing shared file servers, etc. should be seen as enabling technology.

The introduction of the CSCW field has also been used to argue for a paradigm shift in systems development. Grudin (1990) suggests that CSCW can be an opportunity to stress the “importance of ‘workplace democracy’—engaging the users or workers meaningfully in the design process [... although] this time-consuming, labor-intensive approach may not be equally appropriate for all development projects ” (Grudin, 1990, p. 101). Several have defined cooperative design as cooperative work and used this to argue for seeing the tradition of Participatory Design (PD) as an aspect of CSCW (e.g., Kyng, 1991). But, as argued by Bannon this seem to add confusion to both fields rather than defining the field of CSCW: “While certainly various forms of user involvement are important to development of successful CSCW systems use of [PD] techniques or ideas does not automatically signify any focus on cooperative work [...] Nor, in many cases, are PD researchers interested directly in computer support for the design practices they are proposing” (Bannon, 1993, p. 11). Participatory Design should be considered a tradition that has developed and introduced a number of approaches and techniques which, of course, should be applied when designing CSCW applications. Hughes *et al.* (1991) argue that CSCW should be seen as a paradigm shift. Instead of seeing sociology as having a service role in CSCW, CSCW should be seen as posing a challenge for sociology: “A new theoretico-empirical terrain is being formed, as much in the sociology of work and organisations as elsewhere, and the interdisciplinary confrontations invoked in CSCW can be a formative influence” (Hughes *et al.*, 1991, p. 321). This is then related to systems design, but without discussion of what characterizes design. Like the Participatory Design approach, the sociological approaches provides several useful propositions relevant for the field of CSCW, for example techniques for analyzing cooperative work, and a deeper and more coherent understanding of the magnitude of aspects of a work situation that are important to understand compared to traditional systems design.

Schmidt and Bannon (1992) discuss an approach to CSCW based upon a definition focusing on the need for understanding the nature of cooperative work in order to establish a foundation for designing information systems to support cooperative work: “CSCW should be conceived of as *an endeavor to understand the nature and requirements of cooperative work with the objective of designing computer-based technologies for cooperative work arrangements.* [...] in the conception of CSCW proposed here — as a research area devoted to exploring and meeting the support requirements of cooperative work arrangements — CSCW is basically *a design oriented research area.* This is the common ground. Enter, and you must change.” (p. 11).

CSCW must be considered primarily a design discipline. This calls for an attempt to establish a better conceptual understanding of what characterizes cooperative work and its coordination. From this general overview let us take a closer look into the complexity of cooperative work, and how CSCW technologies seek to provide support. In order to focus the description our departure and examples are taken from the nature of design work, but the discussion could be broadened out to many other work domains.

2.1 The complexity of cooperative design

While cooperative work, as noted, is inherently distributed, the distributed character of cooperative work varies, according to the complexity of the interdependence, that is, depending on factors such the distribution of activities in time and space, the number of participants in the cooperative ensemble, the structural complexity posed by the field of work (interactions, heterogeneity), the degree and scope of specialization among participants, the apperceptive uncertainties posed by the field of work and hence the variety of heuristics involved, and so on. The more distributed the activities of a given cooperative work arrangement, the more complex the articulation of the activities of that arrangement is likely to be. If more and more problems are reaching insoluble levels of complexity, as argued by Christopher Alexander in his classic study (1964, p. 3), work *conceived of as a cooperative effort* is of an entirely different order of complexity.

In work domains such as process control, the common field of work, i.e., the plant and the ongoing processes, is essentially given in advance and the task is basically to maintain or achieve a certain system state as defined in terms of a set of parameters. Thus, the interdependencies are largely known in advance and the challenge is to coordinate the distributed time-critical processes. Likewise in manufacturing, but here the challenge is to handle interdependencies which for all practical purposes are intractable due to the myriads of parts which have to undergo a variety of processes in different sequences.

By contrast, the common field of work of cooperative design is essentially emerging. Design work is, of course, situated in an environment which is given in advance. In the case of design in manufacturing and similar settings, design work is highly constrained by the manufacturing capabilities of the enterprise (machinery, skills, etc.) and the previous designs and inventories of parts as well as general standards. Likewise, software design is constrained by the interface standards of the target platform and other software systems as well as by existing code libraries. However, the focal object of the work of designers is only taking shape in and through their very work. The gist of design work can be said to consist in exploring and identifying the interactions between conflicting requirements so as to be able to decide on an acceptable compromise. Much work, including design, is characterized by being a 'wicked problem' (Rittel and Webber, 1973).

The complexities are compounded drastically when solving a ‘wicked problem’ involves multiple actors, in that different aspects of the problem are addressed by different actors and the interdependencies among these aspects, and hence between the actors, emerge and change as the design project unfolds. An example of these problems in cooperative work can be observed in software development. As pointed out by Frederick Brooks, software is invisible and unvisualizable: “In spite of progress in restricting and simplifying the structures of software, they remain inherently unvisualizable, thus depriving the mind of some of its most powerful conceptual tools. This lack not only impedes the process of design within one mind, it severely hinders communication among minds.” (Brooks Jr., 1987, pp. 185 f.).

The traditional strategy for managing these complexities consisted in subjecting the cooperative process to a rigorously hierarchical regime in which ‘the problem’ was defined and refined in a top-down manner and decomposed into atomic fragments which were then addressed sequentially. Multiple levels of management were then required to handle the unavoidable unforeseen interdependencies among the fragmented activities. This strategy has now collapsed in many working areas including design work.

2.2 CSCW approaches to cooperative design

Due to the growth of scientific research activities and the tighter coupling between research and production, the pace of technological change is accelerating in all branches of production. This is further enhanced by the shortening of the incubation period of product design due to the introduction of computer-aided design and engineering technologies. In order to indulge customers, the shortest possible elapse time from order to delivery is becoming a competitive advantage in its own right, and product life cycles are being cut short as yesterday’s models are superseded by tomorrow’s. This reinforces the demand on the flexibility and versatility of manufacturing enterprises. Also in order to humor customers, companies are expanding the list of models and variants in their sales catalogues. As a result, manufacturing has become beset by rampant product diversification (Gunn, 1987; Womack *et al.*, 1990).

In such environments, engineers and designers face huge challenges as they fight to manage the increasing product diversity while trying to accelerate the design process to deal with the harsh competitive realities of a global market and decreasing product life cycles. In response to these challenges the work organization is shifting towards novel organizational concepts such as concurrent engineering. However, in doing so, the coordination and integration of the myriads of interdependent and yet distributed and concurrent activities becomes enormously complex.

It thus seems as if CSCW technologies may be indispensable to such domains if the current transition towards highly responsive work organizations such as concurrent engineering is to succeed. Conversely, the domain of design and

engineering has indeed attracted much attention from CSCW researchers. Very early in the history of CSCW research related to these problems two strategies became dominant: On one hand, CSCW researchers explored different approaches to capturing 'design rationale' and supporting 'organizational memory' and in the course of this line of research a number of experimental systems such as gIBIS (Conklin and Begeman, 1988; Conklin, 1989), Answer Garden (Ackerman and Malone, 1990), and EGRET (Johnson, 1992) were developed and put to test. On the other hand, researchers, especially a team of Japanese researchers around Hiroshi Ishii, addressed the issue of supporting the cooperation among designers and other actors over distance in space by means of a series of very refined shared display facilities (e.g., Ishii, 1990; Ishii and Miyake, 1991; Ishii *et al.*, 1993; Ishii *et al.*, 1994).

Since these pioneering activities were undertaken, a large number of in-depth empirical studies have been carried out within the CSCW community (e.g., Anderson *et al.*, 1993; Carstensen *et al.*, 1995a; Button and Sharrock, 1996; Carstensen, 1996; Carstensen and Sørensen, 1996; Grinter, 1996; Potts and Catledge, 1996; Robertson, 1996; Grinter, 1997; Robertson, 1997; Tellioglu and Wagner, 1997). And in addition to these studies a range of anthropological studies of design that contribute significantly to our understanding of design as a cooperative effort (e.g., Bucciarelli, 1984; Bucciarelli, 1988; Latour, 1993; Bucciarelli, 1994) as well as a large number of studies of design from organizational perspectives (e.g., Adler, 1990) and from the perspectives of software engineering and software development methodology (Curtis *et al.*, 1988; Bødker and Grønbaek, 1996). These studies provides a basis for better to understand how CSCW technologies can assist designers and engineers and other collaborating actors in managing the challenges they are faced with. What comes across in these studies, in all of their methodological variety and the multiplicity of design settings which have been studied, is that one overriding problem is facing contemporary actors, namely the *overwhelming complexity* of handling the myriads of capricious and shifting interdependencies between distributed activities in an orderly and timely fashion.

2.3 A cooperative design work settings - An example

A simple example will illustrate the point: Foss Electric is a Danish manufacturing company that produces advanced equipment for analytical measurement of quality parameters of agricultural products, e.g., the compositional quality of milk in terms of fat content and the count of protein, lactose, somatic cells, bacteria, etc. At the time of our field study,¹ the company was engaged in a large design project called S4000 which aimed at building a new instrument for analytical testing of raw milk. The S4000 project was the first project aiming at building an integrated instrument that would offer a range of

¹ The field research of the S4000 project was done by Henrik Borstrøm, Peter Carstensen, and Carsten Sørensen.

functionalities that previously had been offered by a number of specialized instruments. In addition, as an innovation compared to previous models, the S4000 system would introduce measurements of new parameters in milk (e.g., urea and citric acid), and the performance was to be radically increased. The instrument would consist of approximately 8,000 components grouped into a number of functional units, such as cabinet, pipette unit, conveyer, flow-system, and measurement unit. Finally, the S4000 was the first Foss instrument to incorporate a personal computer (an Intel-based 486 PC) by means of which the configuration and operation of the instrument were to be controlled (through a Windows interface). Eventually, the first version of the software consisted of approximately 200,000 lines of source code.

Altogether more than 50 people were involved in the S4000 project, which lasted approximately 30 months (for the first version).

The design team was faced with quite a challenge: (1) The different subsystems, e.g., the software control system and the mechanical and chemical processes in the flow and measurement system, were intricately interdependent and might interact in unforeseen ways. (2) The S4000 project introduced measurement of new parameters in raw milk for which new technologies had to be developed and mastered. (3) The different subsystems were developed concurrently and the requirements to be satisfied by each subsystem would therefore change as other subsystems were developed. (4) Production facilities at the manufacturing plant were constantly changing as the use of existing machines was optimized and new machines and processes were introduced. Hence, the repertoire of manufacturing processes that the production function could offer to designers and that designers thus had to take into account in their decisions was continually changing. (5) Because of its technological heterogeneity, the S4000 project involved a number of specialities. The core design team consisted of designers from mechanical engineering, electronics, software, and chemistry. In addition, a handful of draught-persons and several persons from organizational entities such as production, model shop, marketing, quality assurance, quality control, service, and top management were involved to varying degrees at different stages in the course of the project. All in all, the project was significantly more complex than previous projects at Foss.

To survive these challenges, the participants took a number of measures to reduce the complexity of managing the project:

As always at Foss, all project participants from the different technical departments were moved to the same office area to create a shared physical space by means of which participants could develop and maintain shared awareness of the state of the project. Furthermore, of course, a sequence of meetings was scheduled at different intervals and, as the project took its course, a large number of ad-hoc meetings were arranged as well.

However, the amount of detailed information that had to be communicated, aligned, negotiated, etc., required more robust measures. A number of procedures and artifacts were introduced to keep track of the state of affairs and to manage

relations and dependencies among actors, tasks, and resources: an ‘augmented’ bill of materials that identified actors responsible for parts in order to support the coordination of mechanical design, process planning, and production in the construction of prototypes (Sørensen, 1994a); a CEDAC board (Cause and Effect Diagram with the Addition of Cards) for coordinating the diagnosis of faults between mechanical design and process planning (Sørensen, 1994b); and a product classification scheme supporting the distributed classification and retrieval of CAD models (Sørensen, 1994c). Some of these procedures and artifacts were invented for this project, some were re-designs of existing artifacts, and others were merely adopted.

The most dramatic measures were taken with respect to the software design process. In the early phases of the software strand of the S4000 project, the software designers felt that their overview of the state of the project was quite defective and that they needed much greater coordination. At the height of the crisis the software design goals were almost abandoned. To overcome the crisis, the software designers developed a repertoire of procedures and artifacts to ensure the monitoring and control of the integration of software components and modules.

An important component in this repertoire, was the ‘software platform’ institution. Initially, the ‘software platform’ was just a point in time at which all software designers would stop coding in order to integrate their bits and pieces. For each platform integration period, one of the designers was appointed as ‘platform master’ which implied that he or she would be responsible for collecting information on changes made to the software and for ensuring that the software was tested and corrected before it was released. Before the software was released as a ‘platform’ for further development, the project schedule was updated with revised plans and tasks for the next three to six weeks. The establishment of the software platform institution was considered absolutely necessary for the S4000 project.

Moreover, the software design team devised and introduced other procedures and artifacts. Most importantly, a ‘bug report form’ with corresponding procedures for reporting, classifying, and correcting faults were introduced (see figure 1) to ensure that bugs were properly registered, that corrected bugs were duly reported, and to make the allocation of responsibilities clear and visible to all members. As a complementary measure, copies of bug forms were collected in a publicly available repository in the form of a simple binder. Let us, for the sake of illustration elaborate a little further on the bug report form.

Initials: (1) Instrument: Report no: (2) Date: (1)			
Description: (3)			
Classification: (4) 1) Catastrophic 2) Essential 3) Cosmetic			
Involved modules: (5) Responsible designer: Estimated time:			
Date of change: Time spend: Tested date: (6) <input type="checkbox"/> Periodic error - presumed corrected			
Accepted by: Date: (7) To be: 1) Rejected 2) Postponed 3) Accepted Software classification (1-5): ____ Platform:			
Description of corrections: (8)			
Modified applications:			
Modified files:			

The actors fill (or add information) in:

The testers: (1), (2), (3), and (4)
The Spec-team: (3), (4), (5), and (7)
The designers: (6) and (8)

The procedure for handling bugs:

- A tester register and classifies a bug (field 1,2,3, and 4)
- The tester sends the form to the spec-team
- The spec-team diagnose and classify the bug (field 3, 4 and 7)
- The spec-team identifies the responsible designer (field 5)
- The spec-team estimates the correction time (field 5)
- The spec-team incorporates the correction work in the work plans
- The spec-team requests the designer to correct the problem
- The designer corrects the bug and fills in additional correction information (field 6 and 8)
- The designer sends the form to the central file
- The CFM sends the form to the PM and insert copy in central file
- The PM verifies the correction
- The PM returns the form to the central file

Figure 1: A translated version of the bug report form and the overall procedure for registering, diagnosing, and correcting bugs. CFM means 'central file manager' and PM is 'platform master'. The form is a sheet of A4 paper printed on both sides. The numbers indicate who were supposed to fill in which information in the form (from Carstensen and Sørensen, 1996).

The bug report form was originally just a blank sheet of paper used by the involved actors to describe the nature of an identified software problem. During the project the form was refined into a tool for registering bugs, diagnosing and prioritizing the bugs, coordinating the correction tasks, and verifying the corrections reported.

A bug report form could be filled in by software designers, other designers, people from quality assurance, or marketing people involved in testing. The originator described and classified the problem. A team of three software designers (called the spec-team) then added information about affected modules, responsible designer, platform period, and an importance priority (as viewed from a software reliability perspective). All forms were filed in a central bug forms file using one of seven categories describing the importance of the problem and whether it had been corrected, not corrected, postponed, rejected and/or verified. Forms were successively re-classified by for example the spec-team, due to results from a integration test, etc.

A list of unresolved problems was continuously produced and accessed by the software designers. Each designer was responsible for fixing the problems (handling bug forms) and reporting back to the person responsible for verifying corrections (the platform master). The flows of forms, acknowledgments, OK-reports, etc. were complicated. One of the software designers described it in a

state-transition diagram containing more than 15 states. A thorough description of the bug report form and its use are provided in Carstensen (1996). A simple model of how the forms flowed among the actors is shown in figure 2. The flow structure were successively negotiated and changed due to, for example, involvement of new designers, recognition of mis-classification of bugs, rejections of responsibility for a specific problem.

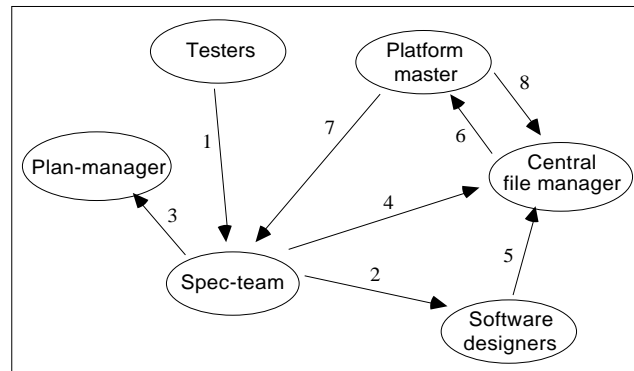


Figure 2: The different roles involved in the coordination of software S4000 testing and correction, and the flow of bug report forms between them. The arrow numbers indicates the sequence when following the standard work flow (adapted from Carstensen *et al.*, 1995b)

By relating correction activities to relevant software modules and to a given platform period, the responsible actors were automatically appointed and the rolling plans were easily updated.

The bug report form was used to collect, classify, and manage errors and suggestions. It mediated coordination information and stipulated coordination in terms of tasks, actors, software modules, and responsibilities. The main purpose of the bug report forms, the central file, the problems list, and the accompanying procedures was to ensure that all problems were registered and filed, to establish a visible organization of responsibilities, and to prescribe how a problem should be handled.

The software designers experienced the hard way that it was practically impossible to handle the distributed testing and bug registration activities of some twenty testers and designers without, *inter alia*, a bug report form and its associated procedures. By devising and introducing these constructs, they managed to alleviate the coordination crisis in the project.

The case of the S4000 project is particularly valuable because we here witness the conception and introduction of specialized artifacts for coordination purposes in response to overwhelming problems encountered in coping with the complexities of articulating the distributed and interdependent activities of cooperative design under conditions that are typical for contemporary design.

3. Requirements for CSCW systems

To assist cooperating actors in coping with the complexities described and illustrated above, CSCW technologies must fulfill requirements on at least two fronts: management of task interdependencies and management of common information spaces.

3.1 Managing task interdependencies

The obvious and fundamental way to coordinate, align, mesh, etc. myriad interdependent and yet distributed activities is to facilitate *mutual awareness* among actors. When an actor changes the state of the field of work, the field of work will, so to speak, emit signals of this change which the other actors may perceive. Support for perceiving changes directly (with their senses) or via indications is often required. If the field of work is a tightly coupled system (e.g., a piece of music being played by an ensemble), changes will affect the work of others instantly and without exception. In a loosely coupled system, on the other hand, changes will propagate over time and contingently. CSCW systems must provide awareness support that reflects these characteristics. To phrase it differently the collaborating actors should have facilities supporting them in *monitoring the state of affairs* of the field of work, for instance by providing the actors with access to the same equipment or through “crying out” when actions are taken (like on a ship bridge). Furthermore, the cooperative work arrangement may be deployed in such a way with respect to the field of work that actors should be able to perceive the state of the field of work in its entirety or in such a way that they only perceive a section thereof.

Since cooperating actors are mutually interdependent services for *monitoring each other* must also be provided, for instance, by having actors working in the same room or by providing some multimedia emulation of a ‘shared space’ (e.g., Ishii, 1990; Ishii and Miyake, 1991; Ishii *et al.*, 1993; Ishii *et al.*, 1994). Awareness support must include facilities for monitoring, directing attention, and handling over tasks at different levels of obtrusiveness and persistence.

A third aspect of monitoring has to do with status of the collaborative activities and comparing these with existing plans, etc. Planning and re-planning are essential coordination activities in collaborative settings. Planning and re-planning requires *communication* channels and structures through which the actors can negotiate. CSCW systems should thus offer communication and negotiation structures.

However, task interdependencies are often of an order of complexity where the provision of facilities for mutual awareness and ad hoc interactions is insufficient (Carstensen *et al.*, 1995a; Carstensen, 1996; Carstensen and Sørensen, 1996; Grinter, 1996; Grinter, 1997; Tellioglu and Wagner, 1997). Other means are required which make task interdependencies tractable. We have called such means coordination mechanisms (Schmidt and Simone, 1996). A coordination

mechanism is, simply put, a coordinative protocol with an accompanying artifact, such as, for instance, a standard operating procedure supported by a certain form. It offers a 'precomputation' (Norman, 1991) of task interdependencies and is thereby instrumental in reducing the space of possibilities facing a competent actor in a given situation. However, the distributed nature of cooperative work is not eradicated by coordination mechanisms. They are merely local and temporary closures which assist actors in managing an otherwise overwhelming complexity. They are used in a distributed way and evolve through a process of local adaptations. The basic requirement for a CSCW environment which supports the construction and use of computational coordination mechanisms is that the mechanisms established are robust in view of the distributed nature of cooperative work: Since work is distributed they must be maintainable in a distributed manner; Since coordination mechanisms often are established bottom-up (Carstensen and Sørensen, 1996) they must be linkable; Since it will not be possible to foresee all deviations they must provide include local control and be of a malleable nature; And since they must be malleable the embedded flows and structures must be visible to the actors (see Schmidt and Simone, 1996 for an elaborate discussion of these requirements).

3.2 Managing common information spaces

As described above cooperating actors need to keep track of the field of work — for designers the joint design, for navigators the ship and the environment, etc. — as it evolves as a result of their distributed activities, in order to be able to see the emerging context of their own contribution, inspect impact of the contributions of others, assess likely interactions, etc. The crucial problem here is that of indexation, that is, the provision of means that allow an individual to assign a publicly visible and permanent 'pointer' to each item so as to enable other individuals to locate the items relatively easily and reliably. To cope with this problem, actors use 'classification schemes', i.e., a conceptual structure (an ordered arrangement of categories) imprinted upon an artifact (for instance, a catalogue or a graph). In the case of the S4000 project the software designers simply used the directory system of the development environment to manage the database of software modules. The conditions of much larger design projects (e.g., automobiles, power plants, airplanes) require far more sophisticated facilities for keeping track of the vast system of information objects in which the evolving design is expressed. The same applies to the problem of managing the vast heterogeneous and distributed repositories of the wider design environment (previous designs, stocks of components, code libraries). While sophisticated indexation facilities exist and have been in widespread use for decades (e.g., group technology), the problem that is coming to the fore in large scale projects and design environments — and this problem is typically ignored in work on 'organizational memory' — is that the classification schemes required for distributed actors to be able to handle these repositories *evolve over time*, in

response to changing conditions and changing conceptualizations and that this evolutionary process is itself distributed. Thus, in large-scale settings actors cannot always negotiate each proposed change to the classification scheme in order to ensure consensus, and even if they are able to negotiate proposed changes the agreed-to categories will typically have local implications and interpretations which are not necessarily known globally within the given community. That is, the classification scheme evolves in a process which is only partially concerted and which involves local innovations which may only gain general acceptance over time (if ever). CSCW systems should provide basic structures that can be used for establishing and maintaining conceptual structures, negotiate these, and still be open and flexible to local interpretations.

3.3 Current approaches and CSCW systems

Inspired by Johansen (1988), CSCW technologies are often characterized by means of a two-times-two matrix separating synchronous vs. asynchronous communication (time dispersion) and distributed actors vs. non-distributed actors (geographic dispersion).

One of the prevalent approaches to design of systems to support synchronous interaction and cooperation is the idea of establishing a shared workspace. The idea is, based on lessons from early What-You-See-Is-What-I-See systems (e.g., Stefik *et al.*, 1987), to provide the users with all their individual and familiar tools. These should be integrated in a computer-based system by which the users can communicate, interact, cooperate, share workspaces (like documents and boards), etc. dynamically over space constraints (Ishii and Miyake, 1991). The goal is to provide a “seamless collaboration media” (Ishii *et al.*, 1994). The illustration systems typically contain shared workspaces where the actors can see each others manipulations, video cameras so the actors can see each other, voice transmission allowing oral communication, etc. Some more simple systems provides only some of the facilities, e.g., shared whiteboard facilities (e.g., Pedersen *et al.*, 1993), or video conferencing systems (e.g., Isaacs *et al.*, 1994). Another tradition concerns collaborative writing, i.e., systems that enables a number of users distributed in space to work on the same document at the same time, e.g., GROVE (Ellis *et al.*, 1991), or ShrEdit (Olson *et al.*, 1990). Also the idea of establishing a media space integrating video and audio technology with network technology so that face-to-face communication can be supported although the actors are distributed in space has been explored. Xerox’s RAVE system (Gaver *et al.*, 1992) is the most well-known. This system has been used for a number of studies resulting in discussions on, for example, control mechanisms in this kind of systems (Dourish, 1993), how privacy influences the use of, and can be handled in, media spaces (Bellotti and Sellen, 1993), ideas for supporting awareness (Gaver, 1991), and how to establish a joint frame of reference for the interaction in such systems (Gaver *et al.*, 1993).

Some of the first systems (approaches) for asynchronous interaction were e-mail and conferencing systems. Engelbart was one of the first to discuss these as means for collaboration (cf., Engelbart and Lehtman, 1988).

One type of systems in the asynchronous category is labeled 'organizational memory', i.e., "the record of an organization that is embodied in a set of documents and artifacts" (Conklin, 1992, p. 133). Focus is on application of a data (or artifact) oriented paradigm for recording information can be expanded to include relevant recording of the context in which the result have been produced, and the process followed when producing these results. The central idea is to build systems supporting traceability, re-use of ideas, establishing consensus, etc. An example of such systems is Answer Garden (Ackerman and Malone, 1990) that provides facilities for routing questions within an organization to relevant experts, or gIBIS which support discussing, establishing, and keep track of the design rationale for a software design project (Conklin and Begeman, 1988). A comparable system is EGRET (Johnson, 1992) that supports 'exploratory group work' by providing a structure of schemes describing tasks, specifications, etc. EGRET keeps track of the variations by offering functionality for listing registered deviations from the schemes.

Another predominant approach is the idea of developing work flow systems automating or supporting the flow of tasks, data, actors, and recurrent or non-recurrent events in order to improve the efficiency and effectiveness of organizations. In this approach (or paradigm) organizations are seen as networks of intertwined flows of work rather than of physical entities and structures. The basic units used for modeling the work thus become the workflows. DOMINO (Kreifelts *et al.*, 1991) is an example of a system in which office processes are modeled in terms of information flows including concurrent flows and processes and alternatives courses of activity. The Coordinator (Winograd, 1986), and later Action Workflow Management System (AWMS) (Medina-Mora *et al.*, 1992) are workflows modeled in terms of 'conversations' (requests and offers, agreements, assertions, and assessments). AWMS is designed for monitoring business processes (the work conducted) and workflows (the sequences of actions) and automating repetitive tasks. Tasks are defined by requests and commitments expressed in the workflow loop. The workflow is composed actors, procedures, information, tasks, and management. Focus is on coordination activities. The structure of the workflow is defined by the language acts through which people coordinate, and each workflow loop concerns a single task. The system allows the definition of rules for routing documents, spreadsheets, forms, software programs, etc. among tasks. The architecture allows interoperability among different applications and across different platforms (Simone *et al.*, 1993).

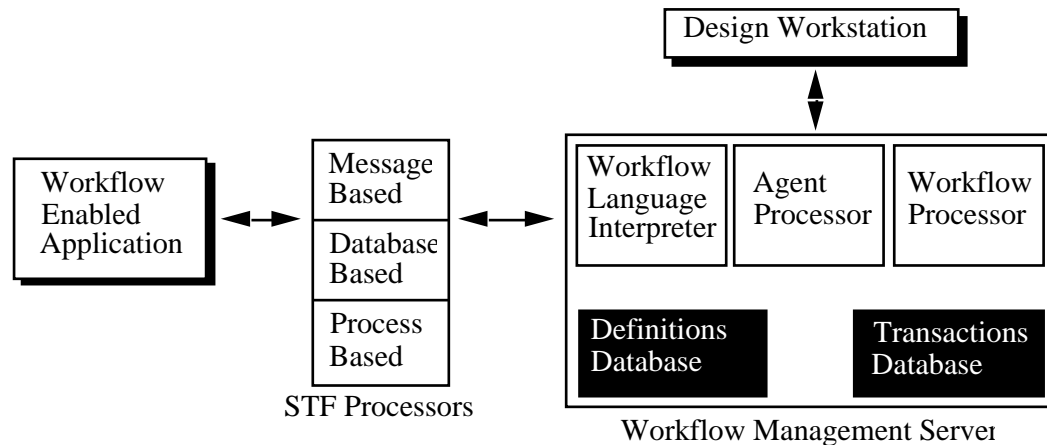


Figure 3: The architecture of Action Workflow Management System (adapted from Simone *et al.*, 1993).

Other approaches aim at supporting the coordination and cooperation going on. The idea is to make models of the work processes and then specify different kinds of support for these. Focus has often been on planning and structuring the work, and then specify models of the work in specification languages, rules, PERT-charts, etc. (Swenson *et al.*, 1994). Such systems are usually shells or environments that can be ‘programmed’ for a specific situation. An early example is Officetalk (Ellis and Nutt, 1988). It was intended to support cooperative management and preparation of office documents. It contained facilities for handling in and outgoing mail, defining different forms to be used, etc. A newer example is ConversationBuilder (Kaplan *et al.*, 1992b) which provide “appropriate mechanisms for the support of collaboration rather than specific policies. Policies can be built out of mechanisms, if the right mechanisms are provided” (Bogia *et al.*, 1993). The conversation for action theory (Winograd and Flores, 1986) has been used as a basis, and ConversationBuilder is basically based upon an understanding of both work and coordination work, and that these on the one hand are intertwined and on the other hand need to be supported ‘orthogonal’ to each other. ConversationBuilder have active support of “*orthogonality*. By this we mean searching for ways of supporting tasks while remaining in some sense orthogonal to the work of the tasks themselves” (Kaplan *et al.*, 1992a, p. 1).

A third interesting system is Regatta (Swenson *et al.*, 1994). It provides support for planning work processes. This is done by modeling the communication required to coordinate the relevant tasks. The modeling is intended to be done in an iterative process by the users themselves: “process plans can be created and modified by the end user allowing users to experiment and find the best processes” (Ibid., p. 16). Regatta was the basis system for developing the well-known TeamWARE Flow from Fujitsu ICL. According to the manufacturer TeamWARE Flow is intended for actors who require a workflow system that can adapt to constantly changing requirements from the organization. The key component is a graphical planner for specifying plans and flows. This tool

provides facilities for the involved actors for specifying and re-specifying the plans. This is done by defining roles, assess qualification, evaluation forms, and possible choices for each activity.

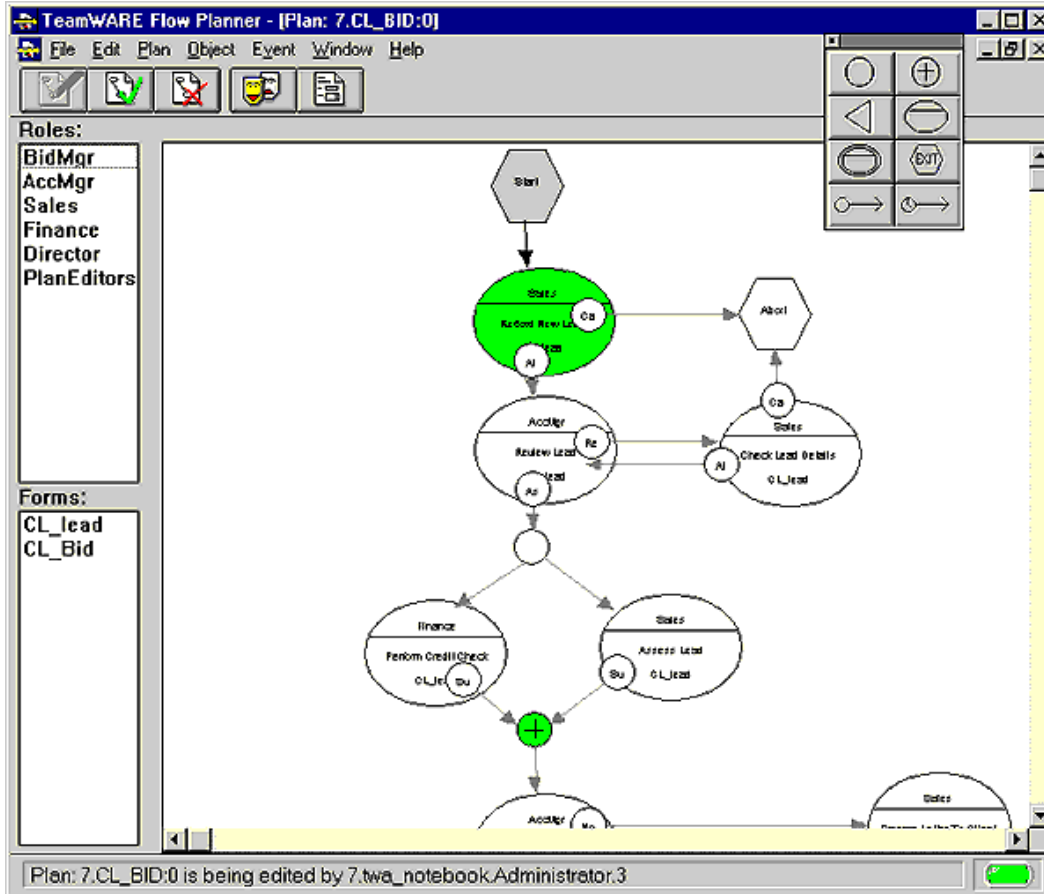


Figure 4: An example of the Fujitsu ICL TeamWARE Flow planner in which the actors can specify their workflows in terms activities and related possible choices, related roles and evaluation forms, assess qualification, etc.

The characteristics of cooperative work described previously indicates that CSCW must have facilities that are quite different from traditional IT systems. This implies that many of the existing platforms (operating systems, database systems, etc.) are insufficient. Much work within CSCW has therefore addressed development of new architectures and platforms for designing CSCW systems. Several of the platforms specified for supporting the development of CSCW systems have already been mentioned. Both The Coordinator, ConversationBuilder, and Regatta can be defined as platforms for building CSCW applications. Languages including structures that specifically supports the development of CSCW applications have been specified. A well-known example is the specification language OVAL (Malone *et al.*, 1992) which is intended to be a general notation for expressing coordination work. The basic primitives (objects, views, agents, and links) are general and at a very low level that do not reflect the relevant primitives used when coordinating work in actual work

settings at a proper semantic level (cf., Schmidt *et al.*, 1993). Another attempt is Ariadne specifying 'coordinative artifacts' as a primary category available to actors for the purpose of defining coordination mechanisms. Ariadne is implemented in an agent based architecture which offers a specialized interoperability language (Simone and Divitini, 1998).

Discussions of what kind of requirements the nature of CSCW application sets up for the basic software (operating system, database system, communication protocols, etc.) have also been taken. Some of these have discussed the implications for the 'distribution technology' and of the problem that cooperating CSCW application users are separated geographically (Rodden *et al.*, 1992; Schmidt and Rodden, 1996). Shen and Dawan (1992) have explicitly addressed control aspects of CSCW applications, and others have discussed how architectures supporting CSCW must be organized (e.g., Bentley *et al.*, 1992; Jeffay *et al.*, 1992). Based on these discussions and ideas attempts to build basis platforms for CSCW systems exist (e.g., Trevor *et al.*, 1993).

The two-times-two matrix suggested by Johansen is not very useful for describing most of today's existing CSCW systems since these have different facilities falling in different boxes. We would therefore suggest to categorize the systems and platforms in a two-times-two matrix where one dimension separates tightly and loosely coupled interaction among the collaborating actors, and the other dimension separates between seeing the computer as a medium for interaction or a regulator of interaction (cf. figure 5 below).

	The computer as a medium of interaction	The computer as a regulator of interaction
Tightly coupled interaction	Mutual awareness <ul style="list-style-type: none"> • Media space (RAVE) • ClearBoard (Ishii) • Collaborative VR systems 	Adjustment <ul style="list-style-type: none"> • Floor control • Shared object services (Rodden et al.)
Loosely coupled interaction	Conceptual structures <ul style="list-style-type: none"> • gIBIS • EGRET 	Coordination <ul style="list-style-type: none"> • Coordinator • DOMINO • ConversationBuilder • OVAL • Regatta • Ariadne

Figure 5: A simple categorization of existing CSCW technologies.

4. Challenges for CSCW-systems design and research

As indicated research and systems design work within CSCW are confronted with a number of hard challenges.

First of all a much better and well conceptualized understanding of cooperative work and its complexity is required. Collaborating actors monitor and cope with immense complex structures in their field of work. However, in order to be able to provide systems for communicating, monitoring, articulating activities, etc. with respect to the field of work we have to understand how field of work is conceptualized and how the typifications applied evolve over time and during work.

For the purpose of designing usable and useful computer systems for cooperative work settings we need to know what makes work situations complex to competent actors and how computer systems may be of assistance to reduce or otherwise cope with this complexity. In 'the natural attitude' (Schutz, 1953) of every praxis, an actor will not take the infinite number of possible perspectives, points of view, or levels of abstractions into consideration, before acting:

"He takes it for granted that his fellow-man will understand his thought if expressed in plain language and will answer accordingly, without wondering how this miraculous performance may be explained. Furthermore, he does not search for the truth and does not quest for certainty. All he wants is information on likelihood and insight into the chances or risks which the situation at hand entails for the outcome of his actions." (Schutz, 1944)

Unless there are reasonable reasons to investigate the situation at a different level of abstraction, an actor will stick to those perspectives and levels of abstractions that are relevant to his or her project. That is, the relevant level of abstraction at which to analyze the complexity of cooperative work is not something which can only be determined arbitrarily or subjectively, or which must be determined through philosophical dispute (which probably amounts to the same). It is a *researchable* issue: what is the relevant perspective and level of abstraction to an actor 'in the natural attitude'? This needs to be determined empirically.

Secondly, and in close relation to the previous point, we have to establish a better understand of awareness. Mutual awareness is a mode of articulation work used for establishing an elementary or undifferentiated consciousness of or sentience to the state of the cooperative effort. The provision of information pertaining to mutual awareness is achieved in the course of doing the work, through the emission and dissemination of requisite signals, signs, and cues. The acquisition of information pertaining to mutual awareness is non-intrusive in the sense that it does not intervene in the flow of work. The ability of being mutual aware of each other can be seen as a common social skill. However, when people are distributed in time and space of if to many actors are involved, which type of information should be provided by a CSCW system and how?

A third overall problem has to do with flexibility. Since cooperative work cannot be prescribed to the level of detail that computer based systems require, we have to establish basic building blocks and platforms so that the actors themselves can establish a CSCW system fulfilling their needs. The problem here is mainly, which building blocks should be provided, and how do we on the one hand ensure a high degree of flexibility and on the other provide building blocks at a semantic level that makes sense in relation to cooperation and coordination activities?

So far, much of the CSCW technology has failed to fulfill the needs due to shortcomings in the existing technology. New and better platforms and operating systems have to be established. These must be based on an improved understanding of the essential characteristics of cooperative work and how these characteristics can be transformed into new requirements for the basic technology upon which we build our CSCW systems.

Acknowledgments

The original field work at Foss Electric was partially funded by the Esprit BRA 6225 COMIC project. Thanks to Carsten Sørensen and Henrik Borstrøm who participated in the field study.

Bibliography

- Ackerman, Marc S., and Tom W. Malone: "Answer Garden: A Tool for Growing Organizational Memory," in *Proceedings of ACM Conference on Office Information Systems*, , ACM Press, 1990, pp. 31-39.
- Adler, Paul: "Managing High Tech Processes: The Challenge of CAD/CAM," in *Managing Complexity in High Technology Organizations*, edited by M. A. V. Glinow and S. A. Mohrman, Oxford University Press, Oxford, 1990, pp. 188-216.
- Alexander, Christopher: "Notes on the Synthesis of Form," , 1964.
- Anderson, Bob, Graham Button, and Wes Sharrock: "Supporting The Design Process Within An Organisational Context," in *ECSCW '93. Proceedings of the Third European Conference on Computer-Supported Cooperative Work, Milan*, edited by G. D. Michelis, C. Simone and K. Schmidt, Kluwer Academic Publishers, 1993, pp. 47-59.
- Bannon, Liam: "CSCW: An Initial Exploration," *Scandinavian Journal of Information Systems*, vol. 5, 1993, pp. 3-24.
- Bannon, Liam, and Kjeld Schmidt: "CSCW: Four Characters in Search of a Context," in *ECSCW '89. Proceedings of the First European Conference on Computer Supported Cooperative Work, Gatwick, London, 13-15 September, 1989*, 1989. - Reprinted in *Studies in Computer Supported Cooperative Work. Theory, Practice and Design*, J. M. Bowers and S. D. Benford, Eds. North-Holland, Amsterdam etc., 1991, pp. 3-16.
- Bellotti, Victoria, and Abigail Sellen: "Design for Privacy in Ubiquitous Computing Environments," in *ECSCW '93. Proceedings of the Third European Conference on Computer-Supported Cooperative Work, 13-17 September 1993, Milan, Italy*, edited by G. De Michelis, C. Simone and K. Schmidt, Kluwer Academic Publishers, Dordrecht, 1993, pp. 77-92.

- Bentley, Richard, Tom Rodden, Peter Sawyer, and Ian Sommerville: "An Architecture for Tailoring Cooperative Multi-User Displays," in *CSCW '92. Proceedings of the Conference on Computer-Supported Cooperative Work, Toronto, Canada, October 31 to November 4, 1992*, edited by J. Turner and R. Kraut, ACM Press, New York, 1992, pp. 187-194.
- Bogia, Douglas P., William J. Tolone, Celsina Bignoli, and Simon M. Kaplan: "Issues in the Design of Collaborative Systems: Lessons from ConversationBuilder," in *Design of Computer Supported Cooperative Work and Groupware Systems. 12th International Workshop on "Informatics and Psychology", Schärding, Austria, June 1-3, 1993*, 1993.
- Brooks Jr., Frederick P.: "No Silver Bullet: Essence and Accidents of Software Engineering" (*Computer*1987); in T. DeMarco and T. Lister: *Software State-Of-The-Art: Selected Papers*, vol. 20, Dorset House Publishing, USA, 1990, pp. 14-29.
- Bucciarelli, Louis L.: "Reflective practice in engineering design," *Design Studies*, vol. 5, no. 3, July 1984, pp. 185-190.
- Bucciarelli, Louis L.: "Engineering Design Process," in *Making Time. Ethnographies of High-Technology Organizations*, edited by F. A. Dubinskas, Temple University Press, Philadelphia, 1988, pp. 92-122.
- Bucciarelli, Louis L.: *Designing Engineers*, MIT Press, Cambridge, Mass., and London, 1994.
- Button, Graham, and Wes Sharrock: "Project work: The organization of collaborative design and development in software engineering," *Computer Supported Cooperative Work. The Journal of Collaborative Computing*, vol. 5, no. 4, 1996, pp. 369-386.
- Bødker, Susanne, and Kaj Grønbaek: "Users and designers in mutual activity: An analysis of cooperative activities in systems design," in *Cognition and Communication at Work*, edited by Y. Engeström and D. Middleton, Cambridge University Press, Cambridge, 1996, pp. 130-158.
- Carstensen, Peter, Carsten Sørensen, and Henrik Borstrøm: "Two is Fine, Four is a Mess — Reducing Complexity of Articulation Work in Manufacturing," in *COOP'95. Proceedings of the International Workshop on the Design of Cooperative Systems, January 25-27, Antibes-Juan-les-Pins, France*, INRIA, Sophia Antipolis, 1995a, pp. 314-333.
- Carstensen, Peter H.: *Computer Supported Coordination*, Writings in Computer Science (No. 61), Department of Computer Science, Roskilde University, Roskilde, Denmark, 1996.
- Carstensen, Peter H., and Carsten Sørensen: "From the social to the systematic. Mechanisms supporting coordination in design," *Computer Supported Cooperative Work. The Journal of Collaborative Computing*, vol. 5, no. 4, 1996, pp. 387-413.
- Carstensen, Peter H., Carsten Sørensen, and Tuomo Tuikka: "Let's talk about bugs!," *Scandinavian Journal of Information Systems*, vol. 7, no. 1, 1995b, pp. 33-53.
- Ciborra, Claudio U.: *Teams, markets and systems. Business innovation and information technology*, Cambridge University Press, Cambridge, 1993.
- Conklin, J.: "Design rationale and maintainability," in *Proceedings of 22nd Annual Hawaii International Conference on System Sciences*, edited by B. Shriver, vol. II, IEEE Computer Society, 1989, pp. 533-539.
- Conklin, Jeffrey: "Capturing Organizational Memory," in *Groupware'92*, edited by D. Coleman, Morgan Kaufmann, 1992, pp. 133-137.
- Conklin, Jeff, and Michael L. Begeman: "gIBIS: A Hypertext Tool for Exploratory Policy Discussion," in *CSCW '88. Proceedings of the Conference on Computer-Supported Cooperative Work, Portland, Oregon, September 26-28, 1988*, ACM, New York, N. Y., 1988, pp. 140-152.
- Curtis, Bill, Herb Krasner, and Niel Iscoe: "A Field Study of the Software Design Process for Large Systems," *Communications of the ACM*, vol. 31, no. 11, November 1988, pp. 1268-1287.

- Dourish, Paul: "Culture and Control in Media Space," in *ECSCW '93. Proceedings of the Third European Conference on Computer-Supported Cooperative Work, 13-17 September 1993, Milan, Italy*, edited by G. De Michelis, C. Simone and K. Schmidt, Kluwer Academic Publishers, Dordrecht, 1993, pp. 125-137.
- Ellis, C. A., S. J. Gibbs, and G. L. Rein: "Groupware: Some issues and experiences," *Communications of the ACM*, vol. 34, no. 1, January 1991, pp. 38-58.
- Ellis, Clarence A., and Gary J. Nutt: "Office Information Systems and Computer Science," in *Computer-Supported Cooperative Work*, edited by I. Greif, Morgan Kaufmann, San Mateo, 1988, pp. 199-247.
- Engelbart, Douglas, and Harvey Lehtman: "Working together," *Byte*, December 1988, pp. 245-252.
- Ensor, Bob: "How can we make groupware practical," in *CHI'90 Human Factors in Computing Systems, Seattle, Washington*, edited by J. C. Chew and J. Whiteside, acm Press, 1990, pp. 87-89.
- Gaver, William, Thomas Moran, Allan MacLean, Lennart Lövstrand, Paul Dourish, Kathleen Carter, and William Buxton: "Realizing a video environment: EuroPARC's RAVE system," in *CHI '92 Conference Proceedings. ACM Conference on Human Factors in Computing Systems, May 3-7, 1992, Monterey, California*, edited by P. Bauersfeld, J. Bennett and G. Lynch, ACM Press, New York, 1992, pp. 27-35.
- Gaver, William, Abigail Sellen, Christian Heath, and Paul Luff: "One is not enough: Multiple views in a media space," in *INTERCHI'93 Conference on Human Factors in Computing Systems, Amsterdam*, edited by S. Ashlund *et al.*, ACM Press, 1993, pp. 335-341.
- Gaver, William W.: "Sound Support for Collaboration," in *ECSCW '91. Proceedings of the Second European Conference on Computer-Supported Cooperative Work*, edited by L. Bannon, M. Robinson and K. Schmidt, Kluwer Academic Publishers, Amsterdam, 1991, pp. 293-308.
- Grinter, Rebecca E.: "Supporting articulation work using software configuration management systems," *Computer Supported Cooperative Work. The Journal of Collaborative Computing*, vol. 5, no. 4, 1996, pp. 447-465.
- Grinter, Rebecca E.: "Doing software development: Occasions for automation and formalisation," in *ECSCW '97. Proceedings of the Fifth European Conference on Computer-Supported Cooperative Work, 7-11 September 1997, Lancaster, U.K.*, edited by J. A. Hughes *et al.*, Kluwer Academic Publishers, Dordrecht, 1997, pp. 173-188.
- Grudin, Jonathan: "Groupware and Cooperative Work: Problems and Prospects," in *The Art of Human Computer Interface*, edited by B. Laurel, Addison-Wesley, ??, 1990, pp. ??-??
- Grudin, Jonathan: "CSCW: The convergence of two development contexts," in *CHI '91. ACM SIGCHI Conference on Human Factors in Computing Systems, New Orleans, April 28-May 2, 1991*, ACM Press, 1991, pp. 91-97.
- Gunn, Thomas G.: *Manufacturing for Competitive Advantage. Becoming a World Class Manufacturer*, Ballinger, Cambridge, Mass., 1987.
- Helander, Martin, and Mitsou Nagamachi (ed.): *Design for Manufacturability — A Systems Approach to Concurrent Engineering and Ergonomics*, Taylor & Francis, London, 1992.
- Hughes, John, Dave Randall, and Dan Shapiro: "CSCW: Discipline or Paradigm? A sociological perspective," in *ECSCW '91. Proceedings of the Second European Conference on Computer-Supported Cooperative Work*, edited by L. Bannon, M. Robinson and K. Schmidt, Kluwer Academic Publishers, Amsterdam, 1991, pp. 309-323.
- Isaacs, Ellen A., Trevor Morris, and Thomas K. Rodriguez: "A Forum for Supporting Interactive Presentations to Distributed Audiences," in *CSCW '94. Proceedings of the Conference on Computer-Supported Cooperative Work, Chapel Hill, North Carolina, October 24-26, 1994*, edited by T. Malone, ACM Press, New York, N.Y., 1994, pp. 405-416.

- Ishii, Hiroshi: "TeamWorkStation: Towards a Seamless Shared Workspace," in *CSCW 90, Los Angeles, CA, October 7-10 1990, New York, N.Y.*, ACM press, 1990, pp. 13-26.
- Ishii, Hiroshi, Kazuho Arita, and Takashi Yagi: "Beyond Videophones: TeamWorkStation-2 for Narrowband ISDN," in *ECSCW '93. Proceedings of the Third European Conference on Computer-Supported Cooperative Work, 13-17 September 1993, Milan, Italy*, edited by G. De Michelis, C. Simone and K. Schmidt, Kluwer Academic Publishers, Dordrecht, 1993, pp. 325-340.
- Ishii, Hiroshi, Minoru Kobayashi, and Kazuho Arita: "Iterative Design of Seamless Collaboration Media," *CACM*, vol. 37, no. 8, 1994, pp. 83-97.
- Ishii, Hiroshi, and Naomi Miyake: "Torward An Open Shared Workspace: Computer and Video Fusion Approach of TeamWorkStation," *CACM*, vol. 34, no. 12, 1991, pp. 36-50.
- Jeffay, K., J. K. Lin, J. Menges, F. D. Smith, and J. B. Smith: "Architecture of the Artifact-Based Collaboration System Matrix," in *CSCW '92. Proceedings of the Conference on Computer-Supported Cooperative Work, Toronto, Canada, October 31 to November 4, 1992*, edited by J. Turner and R. Kraut, ACM Press, New York, 1992, pp. 195-202.
- Johansen, Robert: *Groupware. Computer Support for Business Teams*, The Free Press, New York and London, 1988.
- Johnson, Philip: "Supporting Exploratory CSCW with the EGRET Framework," in *CSCW '92. Proceedings of the Conference on Computer-Supported Cooperative Work, Toronto, Canada, October 31 to November 4, 1992*, edited by J. Turner and R. Kraut, ACM Press, New York, 1992, pp. 298-305.
- Kaplan, Simon M., William J. Tolone, Elsa Bignoli, Douglas P. Bogia, and Alan M. Carroll: "Orthogonal Support Aids Collaborative Tasks," in *Schärding '92, Austria*, 1992a.
- Kaplan, Simon M., William J. Tolone, Douglas P. Bogia, and Celsina Bignoli: "Flexible, Active Support for Collaborative Work with Conversation Builder," in *CSCW '92. Proceedings of the Conference on Computer-Supported Cooperative Work, Toronto, Canada, October 31 to November 4, 1992*, edited by J. Turner and R. Kraut, ACM Press, New York, 1992b, pp. 378-385.
- Kreifelts, Thomas, Frank Victor, Gerd Woetzel, and Michael Weitass: "Supporting the design of office procedures in the DOMINO system," in *Studies in Computer Supported Cooperative Work. Theory, Practice and Design*, edited by J. M. Bowers and S. D. Benford, North-Holland, Amsterdam etc., 1991, pp. 131-144.
- Kyng, Morten: "Designing for cooperation: Cooperating in design," *Communications of the ACM*, vol. 34, no. 12, 1991, pp. 65-73.
- Latour, Bruno: *We Have Never Been Modern*, (French original, 1991), Translated by Catherine Porter, Harvard University Press, Cambridge, Mass., 1993.
- Malone, Thomas W., Kum-Yew Lai, and Christopher Fry: "Experiments with Oval: A Radically Tailorable Tool for Cooperative Work," in *CSCW '92. Proceedings of the Conference on Computer-Supported Cooperative Work, Toronto, Canada, October 31 to November 4, 1992*, edited by J. Turner and R. Kraut, ACM Press, New York, 1992, pp. 289-297.
- Medina-Mora, Raul, Terry Winograd, Rodrigo Flores, and Fernando Flores: "The Action Workflow Approach to Workflow Management Technology," in *CSCW '92. Proceedings of the Conference on Computer-Supported Cooperative Work, Toronto, Canada, October 31 to November 4, 1992*, edited by J. Turner and R. Kraut, ACM Press, New York, 1992, pp. 281-288.
- Norman, Donald A.: "Cognitive Artifacts," in *Designing Interaction. Psychology at the Human-Computer Interface*, edited by J. M. Carroll, Cambridge University Press, Cambridge, 1991, pp. 17-38.

- Olson, Judith S., Stuart K. Card, Thomas K. Landauer, Gary M. Olson, Thomas Malone, and John Leggett: "Computer-supported co-operative work: research issues for the 90s," *Behaviour & Information Technology*, vol. 12, no. 2, 1993, pp. 115-129.
- Olson, Judith S., Gary M. Olson, Lisbeth A. Mack, and Pierre Wellner: "Concurrent editing: The group's interface," in *INTERACT'90 - The Third Conference on Human-Computer Interaction*, Elsevier Science Publishers, 1990, pp. 835-840.
- Pedersen, Elin Rønby, Kim McCall, Thomas P. Moran, and Frank G. Halasz: "Tivoli: An Electronic Whiteboard for Informal Workgroup Meetings," in *INTERCHI'93 Conference on Human Factors in Computing Systems, Amsterdam*, edited by S. Ashlund *et al.*, ACM Press, 1993, pp. 391-398.
- Potts, Colin, and Lara Catledge: "Collaborative conceptual design: A large software project case study," *Computer Supported Cooperative Work. The Journal of Collaborative Computing*, vol. 5, no. 4, 1996, pp. 415-445.
- Rittel, Horst W. J., and Melvin M. Webber: "Dilemmas in a general theory of planning," *Policy Sciences*, vol. 4, 1973, pp. 155-169.
- Robertson, Toni: "Embodied actions in time and place: The cooperative design of a multimedia, educational computer game," *Computer Supported Cooperative Work. The Journal of Collaborative Computing*, vol. 5, no. 4, 1996, pp. 341-367.
- Robertson, Toni Joan: "Designing Over Distance: A Study of Cooperative Work, Embodied Cognition and Technology to Enable remote Collaboration," Submitted for the Degree of Doctor of Philosophy, School of Computing Sciences, University of Technology, Sydney, 1997. 195 pp.
- Rodden, Tom, John A. Mariani, and Gordon Blair: "Supporting Cooperative Applications," *CSCW*, vol. 1, no. 1-2, 1992, pp. 41-68.
- Schmidt, Kjeld: *Modes and Mechanisms of Interaction in Cooperative Work*, Risø National Laboratory, P.O. Box 49, DK-4000 Roskilde, Denmark, 1994. [Risø-R-666].
- Schmidt, Kjeld, and Liam Bannon: "Taking CSCW Seriously: Supporting Articulation Work," *CSCW*, vol. 1, no. 1-2, 1992, pp. 7-40.
- Schmidt, Kjeld, and Tom Rodden: "Putting it all together: Requirements for a CSCW platform," in *The Design of Computer Supported Cooperative Work and Groupware Systems*, edited by D. Shapiro, M. Tauber and R. Traunmüller, North-Holland Elsevier, Amsterdam, 1996, pp. 157-176. - Originally published at the Workshop on CSCW Design, Schärding, Austria, 1-3 June 1993.
- Schmidt, Kjeld, and Carla Simone: "Coordination Mechanisms: Towards a Conceptual Foundation of CSCW Systems Design," *Computer Supported Cooperative Work. The Journal of Collaborative Computing*, vol. 5, no. 2-3, 1996, pp. 155-200.
- Schmidt, Kjeld, Carla Simone, Peter Carstensen, Betty Hewitt, and Carsten Sørensen: "Computational Mechanisms of Interaction: Notations and Facilities," in *Computational Mechanisms of Interaction for CSCW*, edited by C. Simone and K. Schmidt, University of Lancaster, Lancaster, England, 1993, pp. 109-164.
- Schutz, Alfred: "The stranger: An essay in social psychology" (*American Journal of Sociology* 1944); in A. Schutz: *Collected Papers*, vol. 2, Martinus Nijhoff, The Hague, 1964, pp. 91-105.
- Schutz, Alfred: "Common-sense and scientific interpretations of human action" (*Philosophy and Phenomenological Research*, September 1953); in A. Schutz: *Collected Papers. Vol I. The Problem of Social Reality*, edited by Maurice Natanson, Martinus Nijhoff, The Hague, 1962, pp. 3-47.

- Shen, HongHai, and Presun Dawan: "Access Control for Collaborative Environments," in *CSCW '92. Proceedings of the Conference on Computer-Supported Cooperative Work, Toronto, Canada, October 31 to November 4, 1992*, edited by J. Turner and R. Kraut, ACM Press, New York, 1992, pp. 51-58.
- Simone, Carla, and Monica Divitini: "Ariadne: a flexible use of knowledge on work processes," in *Information Technology for Knowledge Management*, edited by U. Borghoff and R. Pareschi, Springer Computer Science, 1998. - In press.
- Simone, C., M. A. Grasso, and A. Pozzoli: "Coordinator, AWMS, DOMINO, UTUCS, WooRKS, CHAOS, TaskManager, and Lotus Notes," in *Computational Mechanisms of Interaction for CSCW*, edited by C. Simone and K. Schmidt, University of Lancaster, Lancaster, England, 1993, pp. 171-216.
- Stefik, M., D. G. Bobrow, G. Foster, S. Lanning, and D. Tatar: "WYSIWIS revised: Early experiences with multiuser interfaces," *TOIS*, vol. 5, no. 2, April 1987, pp. 147-167.
- Suchman, Lucy A.: *Plans and situated actions. The problem of human-machine communication*, Cambridge University Press, Cambridge, 1987.
- Swenson, Keith D., Robin J. Maxwell, Toshikazu Matsumoto, Bahram Saghari, and Keith Irwin: "A business process environment supporting collaborative planning," *Collaborative Computing*, vol. 1, 1994, pp. 15-34.
- Sørensen, Carsten: "The Augmented Bill of Materials," in *Social Mechanisms of Interaction*, edited by K. Schmidt, Department of Computing, Lancaster University, Lancaster, England, 1994a, pp. 221-236.
- Sørensen, Carsten: "The CEDAC Board," in *Social Mechanisms of Interaction*, edited by K. Schmidt, Department of Computing, Lancaster University, Lancaster, England, 1994b, pp. 237-246.
- Sørensen, Carsten: "The Product Classification Scheme," in *Social Mechanisms of Interaction*, edited by K. Schmidt, Esprit BRA 6225 COMIC, Lancaster, England, 1994c, pp. 247-256. - [COMIC Deliverable 3.2].
- Tellioglu, Hilda, and Ina Wagner: "Negotiating boundaries: Configuration management in software development teams," *Computer Supported Cooperative Work. The Journal of Collaborative Computing*, vol. 6, no. 4, 1997, pp. 251-274.
- Trevor, Jonathan, Tom Rodden, and Gordon Blair: "COLA: a Lightweight Platform for CSCW," in *ECSCW '93. Proceedings of the Third European Conference on Computer-Supported Cooperative Work, 13-17 September 1993, Milan, Italy*, edited by G. De Michelis, C. Simone and K. Schmidt, Kluwer Academic Publishers, Dordrecht, 1993, pp. 15-30.
- Winograd, Terry: "A language/action perspective on the design of cooperative work," in *CSCW '86. Proceedings. Conference on Computer-Supported Cooperative Work, Austin, Texas, December 3-5, 1986*, ACM, New York, N.Y., 1986, pp. 203-220.
- Winograd, Terry, and Fernando Flores: *Understanding Computers and Cognition: A New Foundation for Design*, Ablex Publishing Corp., Norwood, New Jersey, 1986.
- Womack, James P., Daniel T. Jones, and Daniel Roos: *The Machine that Changed the World: The Story of Lean Production*, Rawson Associates, New York, 1990. [Reprint published by Harper Collins Publishers, New York, 1991].