The CSCW Paradigm for Software Development

ZELDA VILJOEN & A.L. STEENKAMP
Department of Computer Science and Information Systems
University of South Africa, PO Box 392, Pretoria
E-mail: VILJOZ@ALPHA.UNISA.AC.ZA
Facsimile: (012) 429-3434

Abstract: People work together to solve a wide variety of problems using different forms of cooperation for each class of problem. Modern technology is complex, and therefore it is unusual for an individual to attempt the development of a major project single-handedly. In an attempt to provide computer-based support for the problems that arise when two or more people attempt to cooperate to perform a task or solve a problem, the area of Computer Supported Cooperative Work (CSCW) becomes relevant. The software development process almost invariably involves cooperation that crosses group, professional, and subcultural boundaries. The complexity of software development demands that highly integrated groups of analysts, designers, and users are invilved in the process. Many development activities may occur concurrently. The area of CSCW and advanced information technology, with its enormous capabilities for transmitting and storing information, holds considerable promise for the software development process.

Keywords: Software Development, Software Development Life Cycle, Cooperation, Computer-Based Support, Computer Supported Cooperative Work (CSCW), Groupware, Object-Orientation

Computing Review Category: Software Engineering

1. Introduction

In recent years, literature about *work* has increasingly focussed on the importance of collective communication, tacit knowledge, and group activities (Greenbaum, 1988). The idea of establishing computer support for group-based work activities, which may be called *cooperative work*, is a useful and challenging one, for it breaks away from centralised and bureaucratic systems of communication and control. The shift from computer systems that support a single user working alone to those supporting a group of users working together has considerable impact. It leads to the consideration of the ways people work together in everyday life, and possible ways to support and extend their interactions. Perhaps more importantly, it suggests that the unique capabilities of computers should be embedded more firmly in ordinary work practices (Gaver, 1991).

The birth of the term *Computer Supported Cooperative Work* (now commonly abbreviated to CSCW) is attributed to the computer scientists Irene Greif and Paul Cashman who organised an interdisciplinary workshop on the development of computer systems that would support people in their work activities (Greif, 1988).

In recent years, increasing attention has been focussed on the problems of CSCW which may be broadly defined as the attempt to provide computer-based solutions to the problems that arise when people (more than a single individual) attempt to cooperate to perform a task or solve a problem

(Borenstein, 1992). *CSCW* is a subject that draws on research in various disciplines, such as psychology, sociology, computer science, and artificial intelligence.

Software development involves substantial integrated and cooperative elements. This has particular implications for the way the software system should be developed. The software development methodology should reflect these elements of cooperation (across functional areas or within the organisational hierarchy) and demands some form of *participatory* approach (Green, 1991). This leads to the belief that CSCW should be incorporated in the software development process.

2. Computer Supported Cooperative Work in Perspective

In this section, an attempt is made to provide a conceptualisation of CSCW, anchored in a framework and meta model that identifies the requirements for computer support of cooperative work. The aim is to construct a conceptual model of the field that allows those active within CSCW to have some common reference point, and some understanding of the important aspects of the field.

A broadly accepted framework for the study of CSCW systems classifies the type of work according to its temporal and geographical distribution (Ellis, Gibbs & Rein, 1991). In the two-dimensional space, four types of cooperative working can be identified as shown in Figure 1. These working types correspond to four different meeting types, or scenarios, in which the cooperative work is supported by computers.

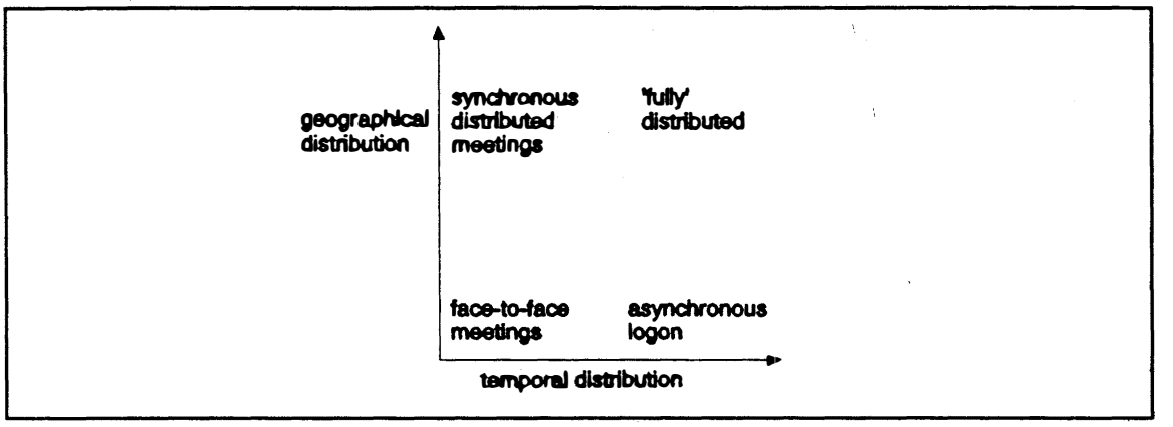

Figure 1 *CSCW Scenarios (Viller, 1991)*

What caused the emergence of an apparently "new" set of issues in the field of information technology, development, and use? To what extent were existing approaches unable to fulfil the needs of computer-based support for cooperative activities? Ongoing developments in technology, approaches to work, and people's needs in the work place have contributed to a concern for a technology that suited the actual work practices of people.

2.1 Main Dimensions of Computer Supported Cooperative Work

CSCW research and reference material contain a bewildering array of diverse technologies, applications and concerns. According to Wilson (1991), there are two major issues: the support of human groups, and the technology which can be used for that purpose. The dimensions of CSCW are the groupwork situation, the players and their tasks, and the technology and media for supporting them.

The dimensions of the CSCW field should be considered to obtain the total picture of what CSCW entails. This would be vital in working towards identifying the meta primitives of CSCW. The group work situation, the players and their tasks, and the technology and media used for supporting them should be characterised (Olson et al., 1993).

2.1.1 The Support of Groups

Major topics associated with the support of groups (Wilson, 1991) include:

- *Individual human characteristics* such as conversation patterns and the making of commitments.
- *Organisational aspects* such as the culture and structure of organisations.
- *Group work design issues* such as user involvement in the work design process, rapid prototyping and usability testing.
- *Group work dynamics aspects* such as group decision making and the cooperative or collaboration process.

2.1.1.1 The Global Characterisation of the Groupwork Situation

CSCW covers a wide spectrum of situations in which groups perform their work. Johansen (1988) characterised the different sets of situations by separating the dimensions of time and location of work, noting whether they are the same or different, as shown in Figure 2. Figure 2 is an instantiation of Figure 1 to separate the major modes of work.

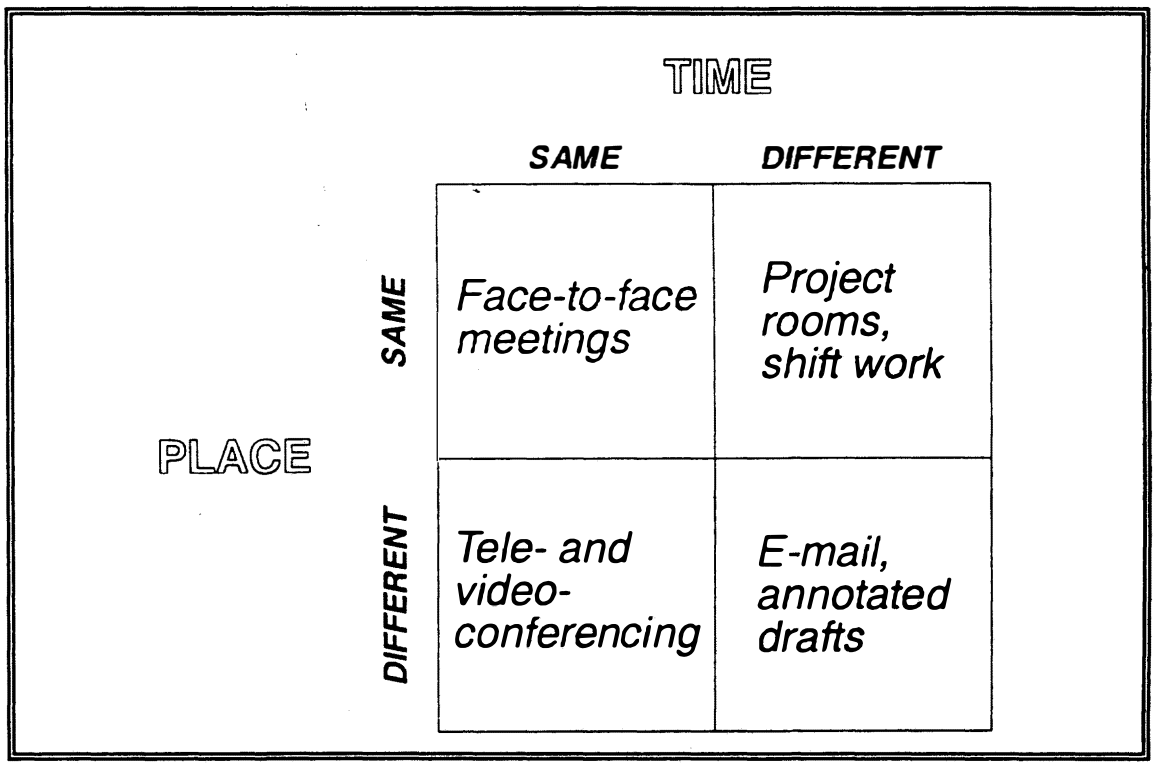

Figure 2 *Time and Location of Work (Olson, 1993)*

In the case of face-to-face meetings and informal project work, work may be done in the *same place at the same time*. Electronic meeting rooms such as Colab are intended to support this type of work

(Olson et al., 1993). Group work also occurs in *different places but at the same time.* Systems in this category focus on remote work and attempt to help individuals communicate effectively with a different set of channels and tools than those used in traditional face-to-face work. Video and audio teleconferencing systems fall in this category. The third common situation is work that is *neither in same place nor same time.* The situation is supported through email, conferencing, and group authoring tools. In addition to the transfer of work objects and comments, this kind of work requires the overhead of coordinating people. The fourth situation, work that takes place in the *same place but at different times*, is less common. It is seen in shift work in hospitals and factories, and in project rooms, places where all the material for a project resides, but individuals in the team come and go.

This characterisation helps separate the major modes of group work. With in-room technology, which supports each person's ability to jot down ideas as they occur and can display one or more person's work, two changes are apparent. Work gets done in the meeting. There also is a smooth swing from silent, parallel thought and development of ideas, to a focussed, one-at-a-time viewing of each person's ideas which may be shared. According to Olson et al (1993), technology has the power to blend synchronous and asynchronous work in new ways.

2.1.1.2 The Tasks of Groups

McGrath (1984) has categorised eight types of work, as shown in Table 1.

Table 1 *Task types (McGrath, 1984)*

Planning tasks	(problem solving, generating plans)
Creative tasks	(generating ideas)
Intellectual tasks	(solving problems with a correct answer)
Decision-making tasks	(solving for preference)
Cognitive conflict tasks	(conflict of view)
Mixed-motive tasks	(conflicts of motive/interest)
Contests/battles	(conflicts of power)
Performances	(psychomotor tasks)

According to Olson et al. (1993), this taxomony specifies whether the work involves cooperation or conflict, whether it involves conceptual work or motoric actions (as in sport), and which phase of development the work is in, whether the work involves generating, choosing, negotiating, or executing. This is not the only task typology possible. Steiner (1972, 1976) has examined in more detail how tasks require information from group members to be combined in order to identify how technology might support activity. However, neither of these typologies consider the kinds of tasks groups engage in when they are trying to learn, to coordinate, or to get to know each other better.

2.1.1.3 The Group Members

Groups differ in both the characteristics of the participating individuals and how they interact with each other, and in their style and pattern of management.

The individuals who comprise the group have various expertise and talents, attitudes, and personality characteristics. Equally important are features of the individuals' interactions with each other. These two factors are known as cohesiveness and structure, and both of these vary as a function of the size of the group (Forsyth, 1990).

It is important to consider how the characteristics of individuals and group structure relate to the

embedded structure in technology. A group with a pattern of democratic, cohesive, but free-for-all behaviour in unsupported work settings might react poorly to a technology that has embedded in it an autocratic method.

2.1.2 Categories of CSCW Technology

Few products built on pure CSCW principles currently exist, though there are many other products which may support the group working process in one way or another. While there is considerable overlap between CSCW and existing groupware products, CSCW has an altogether new kind of software and a new kind of user-to-user relationship (Nolle, 1993). For example, E-mail is analogous to an interoffice mail exchange, while CSCW will resemble face-to-face or phone conversations. The "shared workspace" concept of groupware remains, but in CSCW applications, the computer workspace (the files and applications being shared) is maintained in real-time as much as possible.

A variety of media have been used to support group work, and each medium presents different aspects of interaction. The primary differences have to do with whether the technology supports communication about the work, or whether it represents the work itself. Video connectivity supports conversation, including gestures and, sometimes eye contact, as well as artifacts (e.g., a model of a new landscape). On the other hand, a real-time shared editor supports the work itself, providing to all participants text, outlines, and diagrams for both viewing and changing. Bulletin boards and email blend these roles, supporting both the work and the conversation about the work. This may cause problems in understanding what a particular message means.

It is difficult to dimensionalise the space of CSCW technologies, and attempts are just beginning to appear (Ellis et al., 1991; Malone and Crowston, 1990). These efforts are still preliminary, and there is no widely accepted framework. It is, however, a very promising area of work, moving beyond the early point systems of technology towards general theories of cooperation and coordination that are needed for understanding how technology may fit into human social, organisational, and cultural practices. This is the key to developing effective tools in the future.

The main categories of technology for supporting group work (Wilson, 1991), which include:

- *Communication mechanisms* enabling people at different locations to see, hear, and send messages to each other, for example electronic mail and video conferencing.
- *Shared work space facilities* enabling people to view and work on the same electronic space at the same time. An example is remote screen sharing.
- *Shared information facilities* enabling people to view and work on a shared set of information, for example multi-user databases.
- *Group activity support facilities* to augment specific group work processes, for example the co-authoring of documents, and idea generation.

A number of CSCW products which support the group working process in one way or another have appeared. Product categories include (Wilson, 1991):

- Message systems
- Computer conferencing systems
- Procedure processing systems
- Calendar systems
- Shared filing systems
- Co-authoring systems
- Screen sharing systems
- Group decision support systems (GDSS)
- Advanced meeting rooms
- Team development and management tools.

2.1.3 Integrating the Dimensions

Determining what type of group technology will be successful depends on specifying the four dimensions. The components and dimensions introduced up to now are used in the next section to specify the CSCW meta primitives.

2.2 CSCW Meta Primitives

The meta primitives constitute the basic modelling elements of a specific paradigm. The meta primitives of the CSCW paradigm are derived from the definition of CSCW, the main components and dimensions of CSCW, and established meta models from the literature that have relevance in CSCW. The meta models from the literature that are referred to are:

- Gladstein's framework for group performance (Gladstein, 1984).
- Organigrams and Process Interaction Diagrams (Gladstein, 1984).
- The Logistics model and Petri nets (Joosten & Brinkkemper, 1993).
- An Activity meta model (Kuutti (1991), Engeström (1990)).

By virtue of the first part of its name, the "CS" part of the name CSCW, the professed objective of CSCW is to *support via computers* a specific category of work, namely cooperative work. The term *computer support* conveys a commitment to focus on the actual needs and requirements of people engaged in cooperative work. In analysing the word *computer support*, the attention is immediately focussed on support in the form of *hardware* and *software*. In the context of CSCW, computer support is achieved by means of the hardware which typically consists of *basic hardware* components, as well as *groupware technology*. Basic hardware components include network facilities, processors, peripherals such as printers and terminals, and other hardware components which may be necessary to provide basic computer support to anyone who uses a computer for work purposes. Architectural components which form part of the basic hardware include the information store, organisational database, and the information and communication servers. Groupware technology, on the other hand, includes hardware technology that should be added to the basic hardware components for the support of people who work in groups to attain specific goals in their work. Groupware technology includes video conferencing capabilities (e.g. video and audio equipment), multimedia components, electronic whiteboard components, and other CSCW technology as already mentioned. The architectural components which may be added are the CSCW managers (information, domain, activity, security, and multimedia managers).

Software may be classified as *system software*, *application software*, *middleware*, and *groupware* in the CSCW realm. The system software include the operating system software, utility software, and other system software that are necessary for a computer to function. Customised software systems and software packages which are used for specific applications are referred to as application software. The general functions performed in the CSCW environment are referred to as services. The principle is to provide as many such services as possible via sharable servers. Collectively, the systems providing these services are being called *middleware*, and examples are security services, network services, and repository services.

Groupware software refers to software that is applied for the support of group activity. The groupware provides various *shared services* for fulfilling the needs of people working in groups. The shared services are categorised as:

- Communication mechanisms.
- Shared workspace facilities.
- Shared information facilities.
- Group activity support facilities.

The meta primitives for the computer support aspect condense to the following:

A. Software
- A.1 System software
- A.2 Application software
- A.3 Middleware
- A.4 Groupware
  - A.4.1 Shared services

B. Hardware
- B.1 Basic hardware
- B.2 Groupware technology

Turning now to the second pair of characters in CSCW, "CW" or cooperative work, a specific category or aspect of human work with certain fundamental characteristics common to all cooperative work arrangements is covered. There are many forms of cooperative work, and other terms used are collaborative work, collective work, and group work. Work is, of course, always *social* in the sense that the object and the subject, the ends and the means, the motives and the needs, the implements and the competencies, are socially mediated (Schmidt & Bannon, 1992). However, people engage in *cooperative work* when they are *mutually dependent* in their work and therefore are required to cooperate in order to get the work done. The term *cooperative work* should be taken as the general and neutral designation of multiple persons working together to produce a product or service. In terms of cooperative work, the *application domain* and the *participants* involved in group activity are important elements. The participants also fulfil different *roles* (e.g. manager, secretary, and chairman) in the group. For example, in the group situation when developing software a specific person may play the role of either domain specialist, project leader, analyst, or designer. The *level of work* element refers to the organisational level in which a participant performs cooperative work. The organisational level, level detail and job particulars are attributes of the level of work primitive. The highest level is the *universal level*, then the *wordly level*, and then the *atomic level*, the lowest level. The application domain element refers to the specific application supported by CSCW. The variety of application types include manufacturing applications, production control applications, broadcasting applications, and health care applications. The *nature of cooperative work* taking place within the application domain are also of importance. These include projects, meetings, committees, and task forces. Different *types of work* refers to either managerial, creative, physical or technical work. The specific work performed, the location of the work, and the temporal characteristics of the work are attributes of the type of work.

The meta primitives for the cooperative work aspect may now be derived. The previous discussion higlighted the important elements that are related to cooperative work and as before they will form the meta primitives. The cooperative work meta primitives are:

C. Participants
- C.1 Roles
- C.2 Level of work

D. Application domain
- D.1 Nature of cooperative work
  - D.1.1 Types of work

An *object model* illustrating these meta primitives and their relationships is represented in Figure 3. An *object* is defined as a concept, abstraction, or thing with crisp boundaries and meaning for the problem at hand. An *object model* captures the static structure of a system by showing the objects in the system and their relationhips. Most object models, including those proposed by Rumbaugh et al. (1991), Booch (1991), Coad & Yourdon (1990), and Jacobson (1987) capture the attributes and operations that characterise each class of objects. For the purpose of this discussion, the attributes and operations are not important and are not incorporated in the CSCW object model.

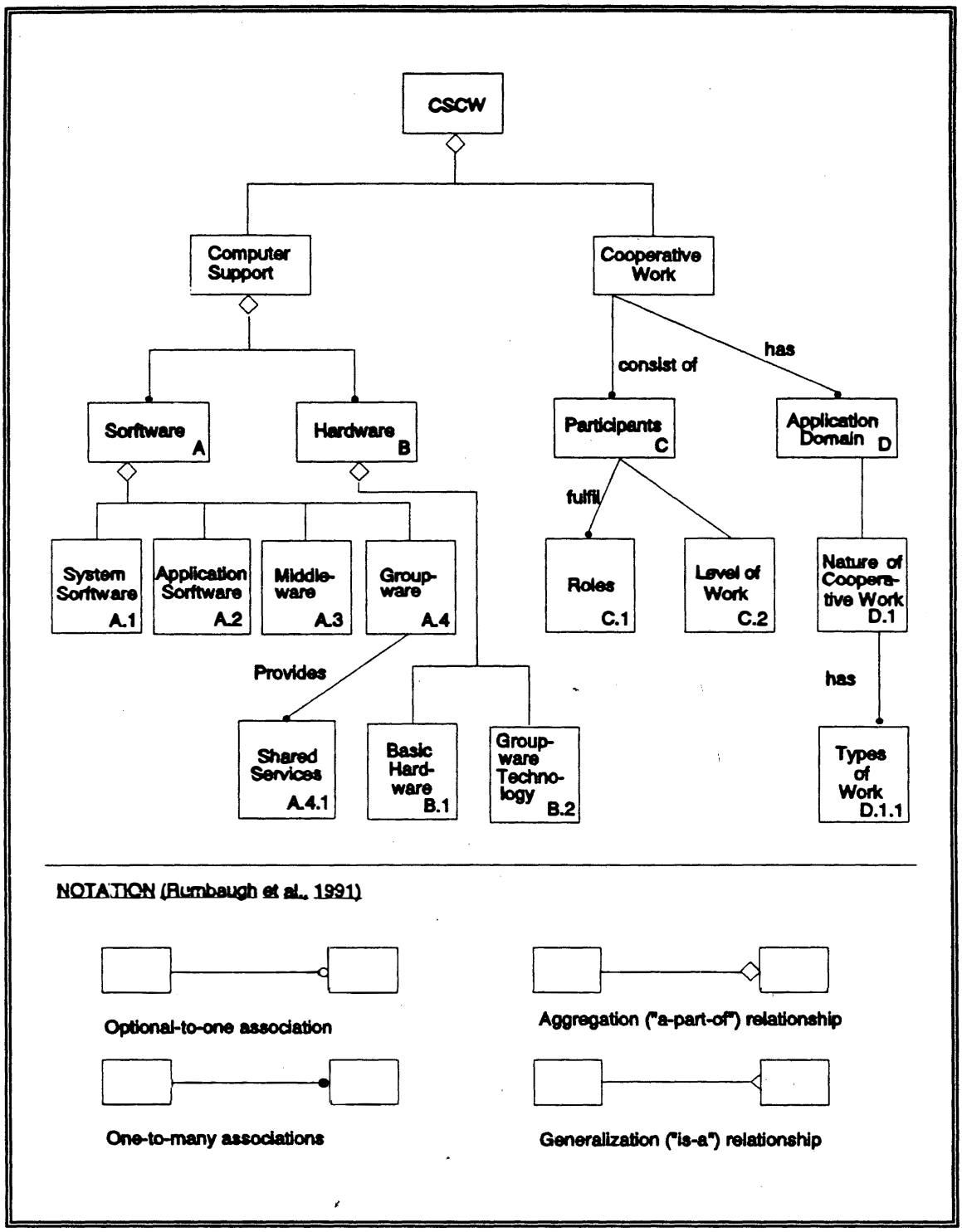

Figure 3 *The CSCW Object Model*

2.3 A CSCW Conceptual Model

Initially, a general conceptualisation of the CSCW paradigm is established, as illustrated in Figure 4. Figure 4 depicts the notion of groupware (electronic components) in support of participants involved in group activity.

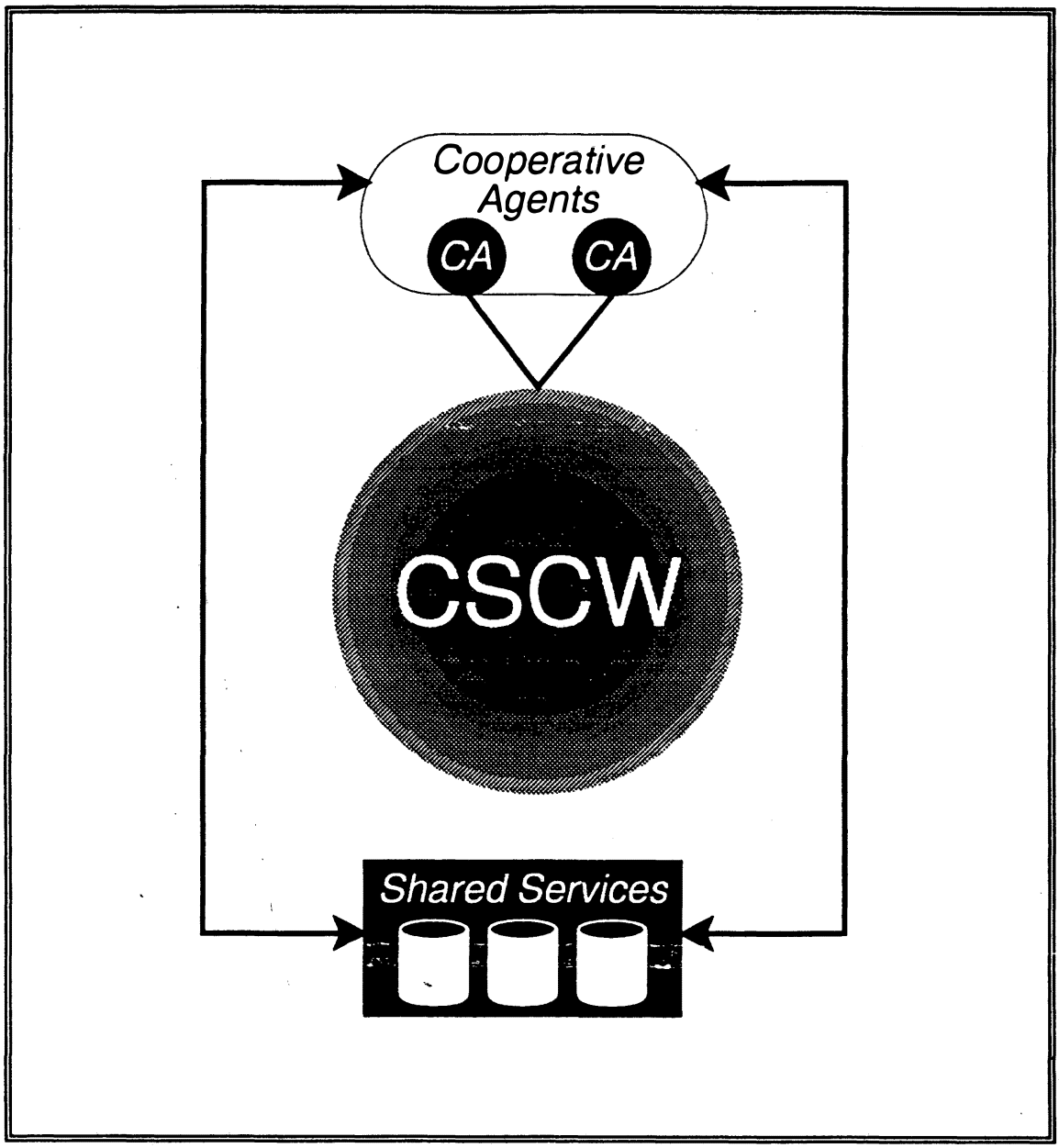

Figure 4 *General Conceptualisation of the CSCW paradigm*

The model depicted in Figure 4 comprises the following main components:

- *CSCW*, the computer supported cooperative work paradigm including all the components that are necessary for the computer-based support provided to participants engaged in group activity.
- *Cooperative Agents*, which allow the participants in the groupware communications to organise their work, to communicate and to ensure that the communications and cooperative work are performed in a coherent and consistent way.
- *Shared services*, which allow the participants to access common resources and to share common information.

The general conceptual model as abstracted in Figure 4 can now be instantiated with the meta primitives of CSCW. The first instantiation of the conceptual model is illustrated in Figure 5.

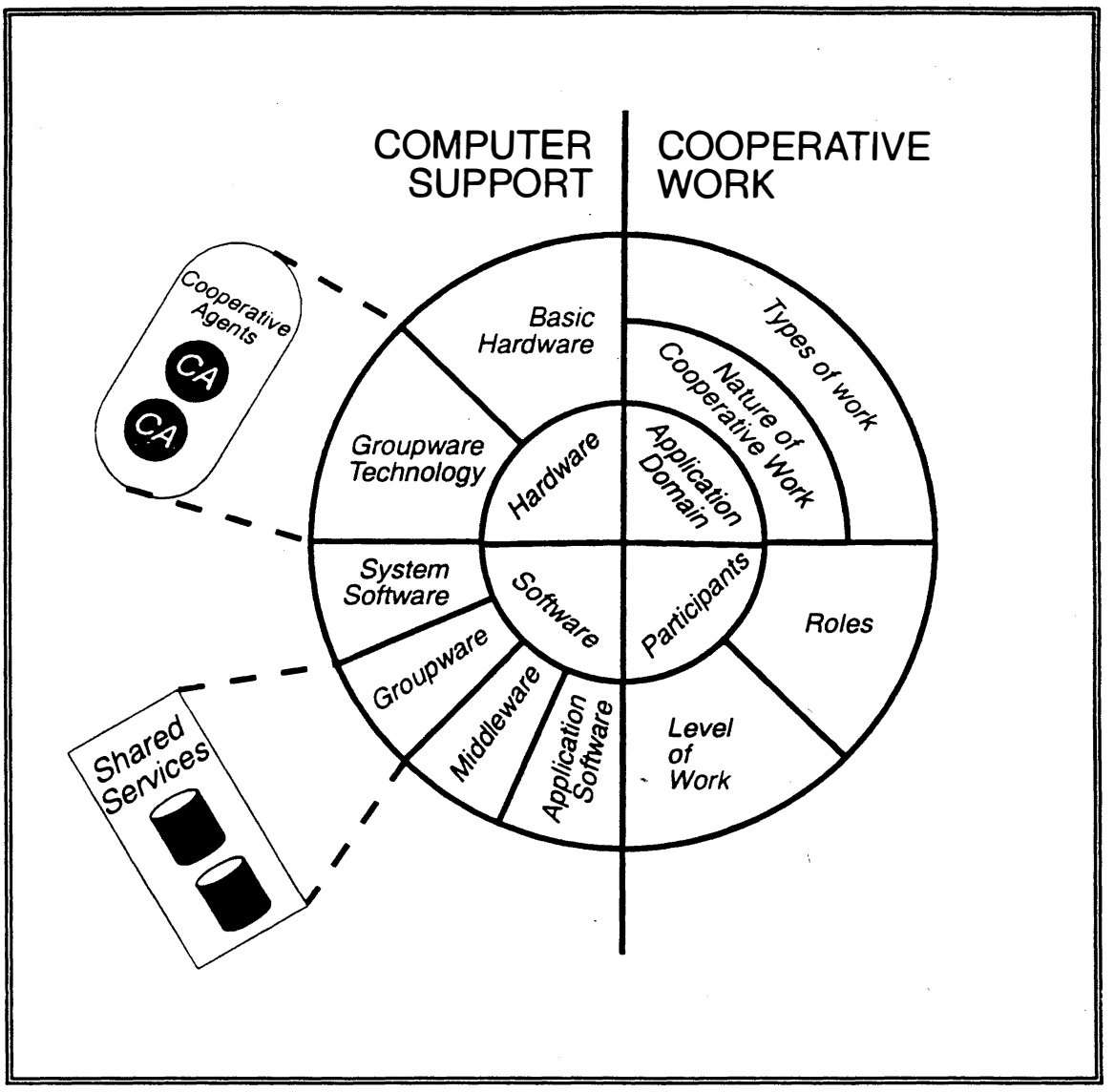

Figure 5 *Conceptual Model of CSCW Meta Primitives*

3. Software Development within a CSCW Environment

Software development according to sound *software engineering (SE)* principles aims at producing quality software systems efficiently. More specifically, the software development process involves a set of activities aiming at conceptualising, specifying, and producing software systems in accordance with the requirements of the users, under existing economical and technological constraints.

Many of the activities of the software development process could be performed either individually or in collaboration. As modern technology is complex, it is unusual for an individual to attempt the development of a major software project single-handedly. Integrated groups of analysts, designers and users work together to establish a clear understanding of the software system that should be developed. These software teams have the challenge of solving the difficult problems of task coordination and information integration.

Software development almost invariably involves cooperation that crosses group, professional, and subcultural boundaries (Bannon, 1993). Different groups, professions, and subcultures embody different perspectives - they communicate in different "jargon". The difficulties of working in such

situations where individuals in the group have different practices, traditions, and working objectives may lead to a lot of time wasted in clearing up differences between the parties involved. Different groups reflect different ways of doing things (a different ontology[1] and epistemology[2]). These ydistinct groups will be referred to as semantic communities. Efforts within the systems development process to emphasize the importance of good communicative practices between the differing semantic communities attest to the difficulties that are experienced.

Many problems are experienced in the early phases of the software development process, particularly in the analysis phase. One of the challenges that software developers face in this cycle is the overall analysis (and subsequent detailed specification) of complex and ill-defined opportunities surrounding the coupling of user needs with system functionality. The opportunities span a spectrum that ranges from easy to define (and specify) to continually evolving and almost impossible to specify. There are often extensive communication and cooperation barriers between the analysts and the users. This could lead to software developed without fully understanding (or being able to build on) what users already know about their tasks. Software developers are often forced to assess situations, determine user requirements, define system functionality, and develop software within very short time frames and at low budgets, without compromising system functionality or putting users into compromising positions.

Groupwork during the software development process introduces problems of organisation, coordination and communication. The tasks carried out by the groups can rapidly become overwhelming because of the complexity of the interactions and the amount of information that is generated. Due to the interdependence in conducting their work, cooperating workers have to articulate (divide, allocate, coordinate, schedule, mesh, interrelate, etcetera) their respective activities. Entering into cooperative work relations, the software developers must engage in activities that are, in a sense extraneous to the activities that contribute directly to fashioning the software product and meeting the requirements. A justification for incurring this overhead cost and the reason for the emergence of cooperative work formations in software development is that workers could not accomplish the task in question if they were to do it individually, at least not as well, as fast, as timely, as safely, as reliably, and as efficiently as a group could (Schmidt, 1991).

Nowadays, truly distributed applications seem natural in the face of a high ratio of computing power to available communication bandwidth (Borenstein, 1992). The users and software developers involved in a specific software development project may be widely separated geographically. For such cooperative development efforts, distributed solutions seem inevitable. Unfortunately, the practical success of such systems has been limited. Three problems that are not generally faced by single-user development environments should be solved (Borenstein, 1992). Firstly, there is the problem of the remote installation, where the participants of the cooperative development effort may have differing degrees of motivation for allowing the installation of the cooperative development software. Related to the problem of getting software installed at distributed sites is the problem of getting the participants in the software development process to use the distributed development software once it is installed. Getting software to work on a wide variety of platforms is a major task, and one that has to be carefully considered before establishing a fully distributed development environment. Complex organisations are characterised by distributed decision making, and require a sharing of perspectives among participants if they are to coordinate activities and adapt to changing circumstances.

The problems have resulted in the realisation of a need to unleash all the resources of cooperative work: horizontal coordination, local control, mutual adjustment, critique and debate, and self-organisation (Schmidt, 1991). Tools and techniques for cooperation between end users, management and professional designers should be applied in the software development process.

---

1 *Ontology* is a branch of metaphysics dealing with the nature of being (Oxford Dictionary, 1982).

2 *Epistemology* is the theory of the method or grounds of knowledge (Oxford Dictionary, 1982).

Advanced information technology, with its extensive capabilities for transmitting and storing information would seem to hold considerable promise for groupwork (Gorry, 1988), and hence the role of CSCW. This also has relevance for the software development process.

3.1. Support for the Cooperative Work in Software Development

The increasing size of, and limited time for the development of software is the basis for the growing importance of distributed development teams (Hahn et al., 1990). The support of this type of group activity may increase synergy and paralellism making the software development process more productive. The issues of group activity, concurrent development, distributed development, and technological support for group activities motivate the use of CSCW in software development.

Within the area of cooperation the work of software development teams constitutes a natural and important application. The shift in emphasis that seems to be implied by the emerging paradigm of computer supported cooperative work for this application is twofold (Hahn, 1990):

- Software development by teams, though being constrained by technical requirements, is recognised as a *social* process comprising the formal or informal interactions of the members of the team (*group work*) within an organisational setting.

- Social processes (*work procedures*) in task-oriented groups underlie particular conditions for *negotiations*, *commitments*, and *responsibilities* that are essential for smoothly accommodating dynamic changes of the project environment.

In discussing teamwork support, it is useful to distinguish between different levels of support and different scales of cooperation (Jarke et al., 1992). In terms of support levels, *connectivity* (the ability to technically exchange data), *interoperability* (the ability to exchange semantically meaningful information), and *cooperation* (the ability to enhance individual work by contributing towards a common goal) are relevant. In terms of cooperation scale, *collaboration*, *communication* and *coordination* can be distinguished. According to Ellis et al. (1991), collaboration refers to joint work on a common object, whereas communication refers to the exchange of pieces of information (messages or development objects). Collaboration and communication tools augment human ability for cooperation in small groups by breaking barriers that have existed in traditional real-time and asynchronous group work (Jarke et al., 1992). On the other hand, coordination structures (and sometimes limits) the flow of communication and the way of collaboration in a typically larger group, to overcome overloading.

In establishing computer support for software development, some important characteristics of the process should be considered (Marmolin et al.,1991). Firstly, software development is an iterative process in which each activity is characterised by a mixture of analytic, structured, linear, creative, chaotic and nonlinear behaviour. The behaviour depends on the development phase, the state of the problem, and the size and skills of the development team. The bottom-up approach seems to be dominant in small research oriented development tasks, when the problem is ill-defined, the design teams are small and prototyping is used. Although support for processes is important, both analytic and creative software development activities should be supported.

Secondly, the earlier stages of software development are characterised by intuitive information gathering processes rather than by formal analytic processes. Concrete representations play an important role in understanding and evaluating development ideas. Thus, the support given has to focus on informal cooperation. According to Marmolin et al. (1991), tools for idea generation (story board facilities) and facilities for observing other systems, and tools for visualising and describing ideas are often more valuable than analytic tools.

Thirdly, the view in which good development is characterised by the ability to integrate knowledge into a unified view and transform it into computational structures is adopted. Support is necessary for integration of knowledge by learning and development from a common frame of reference.

Finally, software development is regarded as cooperative work. Thus, support for collaboration, coordinationand communication are necessary. Marmolin et al. (1991) refer to the need for informal and formal cooperative tools for recordkeeping of ideas and development concepts, change facilitation, information sharing, and project management. Support for cooperation, which is essential in software development, has to be very effective and so easy to use that it does not interfere with the development activities themselves.

The cooperative tasks in a distributed software development environment can be classified as either conference tasks, co-working tasks, information exchange tasks, or management tasks (Marmolin et al., 1991). A *conference task* is a discussion exchange of experience and knowledge between two or more team members. These discussions may take the form of negotiations, idea generation, problem solving, and briefings. The task could be asynchronous or synchronous, formal or informal and have social and communicative characteristics. It could be supported by electronic conference systems and mail systems. A *co-working task* is any activity concerned with the cooperative production of a document or other kind of product in a synchronous or asynchronous way. This task could be supported by distributed applications including co-editors, co-authoring and annotating systems. An *information exchange task* is an activity concerned with the exchange of documents and other information between two or more team members. It could be supported by shared databases, hypertext libraries, record keeping tools and other forms of group memories. The *management task* is an activity for coordinating and supervising of cooperation within a team. It includes planning and scheduling tools which may be supported by PERT and GANTT tools.

The classification of the generic cooperative tasks can be used as a basis for the establishment of support for the cooperative development environment. Next, the requirements of tools for the support of cooperative software development will be examined.

3.2 Requirements of CSCW in Distributed Software Development Environments

Based on the examination of the software development process and cooperative or group work, the following set of general functional requirements are relevant for groupware support in a distributed software development environment (Marmolin et al., 1991):

- *Support informal cooperation.* This is a most important requirement of a distributed software development environment that is also hard to fulfil. This requirement implies that the environment should support communication of social behaviour patterns, establishment and development of personal relations, and drop in meetings.

  Moreover, interactions of groups require support beyond the formal level of technical communication lines such as e-mail and electronic conferencing systems (Hahn et al., 1990). The social protocols that underlie group communication have to be accounted for in terms of human strategies and policies for argument exchange, contract assignment and decision making.

  Support is needed beyond basic *multi-user* facilities which are used to partition teams with standard schemes of concurrency control. The support of group interactions should account for human cooperative techniques such as negotiations, and commitments.

- *Support sharing and record keeping of software development information.* Research has shown the need for supporting sharing and record keeping of important development information, especially in larger teams or groups. Thus the environment should support record keeping and sharing of requirement, design and implementation information, development deliverables, development ideas, commitments and work plans.

- *Support sharing of background knowledge.* This requirement refers to important preconditions for cooperation. These include 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 development problem.

Tools need content-oriented specification of knowledge beyond language facilities. According to Hahn et al. (1990), proper tool support and properly controlled tool integration should consider domain knowledge of the underlying project, working procedures and languages used for support specification, design and implementation.

- *Support presentations of ideas.* Concrete representations are very important in software development. Therefore the distributed software development environment should support different ways of presenting and visualising ideas for other team or group members. Examples are visualisation tools and story board or white board facilities.

- *Support strategies reducing the need for co-working.* Efficient tools for reducing the "collaborative load" may be more important than tools to support co-working. Division and integration of work, encapsulation and sequential processing should be supported.

- *Support co-working.* There will always be a need for co-working. Asynchronous co-working in particular should be supported by annotating and reviewing systems. The support of synchronous co-working are not as important, except for support of interface design together with users. at other locations.

- *Support management activities.* Management should be supported in terms of planning, monitoring, reviewing and control of software development projects. An example of a management tool is electronic calenders which have been proved not to be very useful.

Although the main concern is with distributed software development environments, it is not assumed that groupware should be a substitute for all face-to-face meetings. Complex cooperative work involves a continuing need for face-to-face meetings. This is especially applicable in initiating and planning of the cooperative work. It is believed that well designed distributed software development environments may both reduce the need for face-to-face meetings and provide new and more effective ways of cooperation.

3.3 CSCW Technology for Software Development

There are numerous examples of both commercial products and research prototypes for most of the major categories of CSCW technologies. Many isolated tools are already available to support software development processes. These include tools for modelling during analysis and design, resource management, project control, or hypertext facilities for project documentation. However, these tools tend to have limitations in terms of a sound coverage of the *knowledge* of the software domain, and a centralised view of project planning. They provide island solutions incapable of being integrated. Recently, theories and tools have appeared that take a less centralised viewpoint of software projects and at the same time account for the human factor inherent in work procedures. Initially, a software development environment incorporating these theories and tools were perceived as a forum of communication considering notions from speech act theory and other, more ad-hoc conversational models as the basis of tools such as typed messaging or conferencing systems (Hahn et al., 1990) . However, a pure conversation perspective is inadequate neglecting technical aspects of software development.

With so many CSCW technologies available, there is now a trend in CSCW research towards integration along numerous dimensions. What is needed is an *integrated project support environment* (IPSE) consisting of multi-user tools suitable for computer-aided group work in software projects. The requirements of CSCW in software projects that were determined in the previous chapter are used to establish a framework for cooperative distributed software development which may form the foundation for integrated project support environments.

3.4 CSCW Software Development Environments

The idea of a Software Development Environment (SDE) as a comprehensive, integrated set of tools supporting the complete software development process has been the topic of research over the last few years. Supporting the software development process means to support the modelling techniques of the process, and the development and maintenance of all kinds of documents including requirements specifications, software design specifications, code listings, technical documents, and manuals. The ultimate goal of a SDE is to improve the quality of the final product, to support reuse in and across software projets and to free developers from routine work (Peushel et al., 1991).

A major challenge for future SDEs is the support of (possibly large) teams of software developers who may even be geographically distributed. Such distributed software environments exist, but the available machine support is usually restricted to local or wide area networks and corresponding low-level protocols that only enable simple file transfer, rudimentary configuration management support, and a mail system (Peushel et al, 1991). The need is for coordinating access to shared information on different levels of granularities (e.g. form complete systems of modules or documents down to procedure definitions in a single module). There is also a need for dedicated message servers for conveying information about project states, critical tasks to do, and getting feedback. Examples of research projects which tackled the problem of team support are MARVEL (Kaiser and Feiler, 1987), ARCADIA (Belz et al., 1988), ALF (Benali et al., 1990), and MERLIN (Peushel et al., 1991). A further achievement of such an environment is the computer-supported integration of development and management activities. It is envisaged that the petri net-based DesignNet Model of the OISEE project which presents the universal level of the development process model, as previously illustrated in Figure 3.1, should be computer-supported for the integration of development and management activities.

3.4.1 Generic Facilities

This section outlines the facilities that a software development support tool must provide in order to meet the goals of flexibility and active support. These generic facilities were compiled by Kaplan et al. (1992) in a paper which reports on the development of an open, flexible and active support environment for software development based on the *ConversationBuilder*. The first set of activities is concerned with users being able to work on and relate among arbitrary sets of activities. These facilities are necessary to:

- Provide the ability to specify new kinds of activities to the system. The specification of such situations are called protocols.
- Allow the user to have as many activities running simultaneously as is useful.
- Allow the user to relate activities to one another as is suitable, in order to make an activity subordinate to another, or group of other activities.
- Be able to impose obligations on other users, or other contexts, from any context, and be able to manipulate one's own obligations. This entails handing the obligations over to others, changing their context, declaring them complete and refusing offered obligations.
- Allow the user to switch among activities at will and as effortlessly as possible.

The second set of facilities is concerned with helping the user to determine his current position in the system, and the options available to him. These facilities are to:

- Help users determine how they got into a particular context.
- Indicate the legal actions which may be performed in a given context.

- Perform any of the legal actions.
- Indicate to the user the obligations which must be met before a task is complete.
- Enable the user to understand the relationships (if any) that exist among the data that are present in the system.
- Enable the users to understand the relationships among all the contexts in which they are involved.
- Allow members of a group to be aware of what ather group members are doing.

The third set of facilities is miscellaneous, but necessary. These facilities are to:

- Allow the construction of arbitrary networks (hypertext systems) of data, in which both nodes and links can be typed, so that both shared and private data of various types can be modelled in the system.
- Continue to provide traditional support facilities, such as compilation, version control, editing and mailing.
- Allow the incorporation of new tools as required by users.

3.4.2 Cooperative Models for Software Development

Repositories are the central information servers for design environments. Early repositories generally provided the service of storing evolving objects. However, there is evidence especially in the software engineering domain that repository technology will soon be used to integrate whole environments, even organisations (Jarke et al., 1992). Figure 6 illustrates these environments which incorporate human agents, their local workplaces and tools, and structured communication either directly by message-passing over the network or indirectly by shared information in the repository. In such cases, the repository matures to an active center of communication for complex cooperative development processes.

With development in the area of computer communications, information represented in different formats, such as voice, graphics, images and text can be processed, stored, retrieved and distributed. In the area of communications, new technologies are available for developing integrated networks capable of providing the level of service required by different media. This is motivated by technology transformations in the following domains (Karmouch, 1993):

- The emergence of high-speed networks with powerful workstations and new storage technology imposes a new way of processing information.
- Distributed system configurations are possible where several powerful workstations share resources at different sites by communicating through local area networks (LANs). Other configurations may cover wider areas such as LAN interconnection through MANs and wide area networks (WANs).
- Faster networks, high performance processing and storage systems directly manipulate new types of information such as video, voice and image, all integrated in a single entity (the so-called multimedia document).
- Optical fibre technology has overcome the limitations of current networking technology in providing high speed networks, permitting the transfer of large amounts of multimedia information over a single channel in an integrated and synchronised manner.

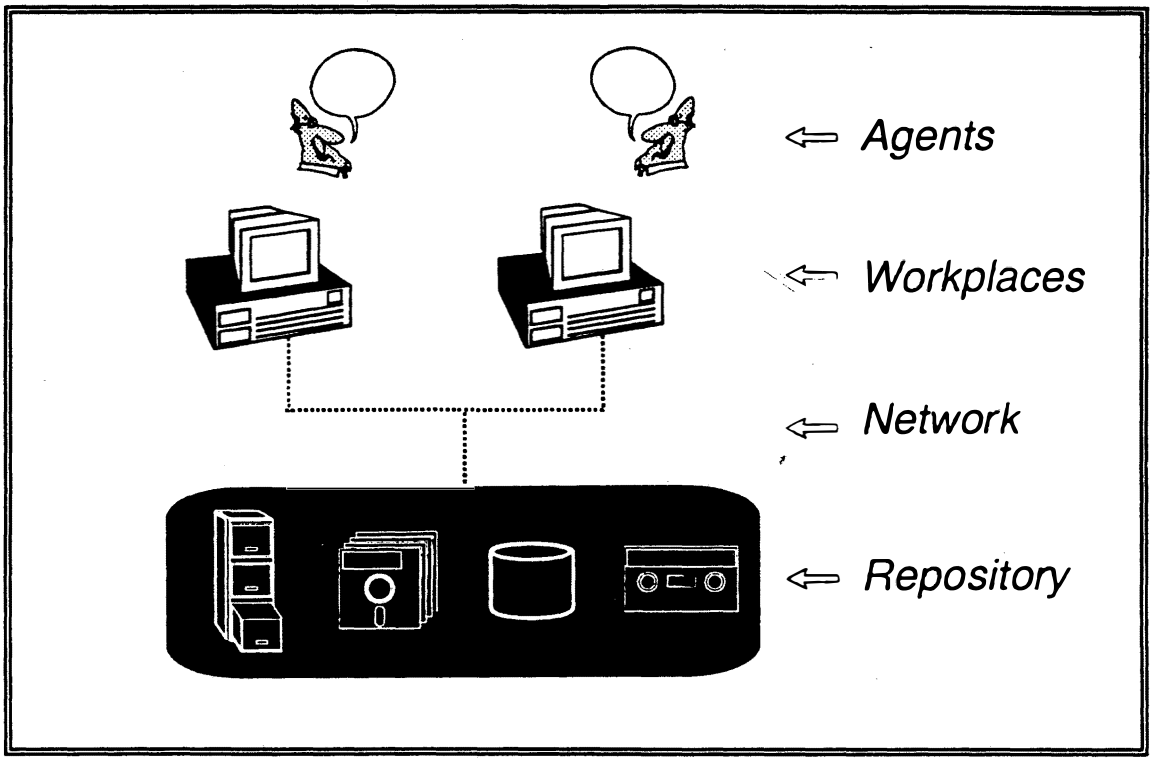

Figure 6 *A typical Software Development Environment (Jarke et al., 1992)*

The architecture that is proposed for computer-supportedcooperative work in software development should be able to fulfil the future needs of multimedia distributed cooperative work. The architecture is adopted from an architecture established by Brodie and Ceri (1992) for the new generation information systems called Intelligent and Cooperative Information Systems (ICISs). These systems provide forms of cooperation and intelligence. Intelligent and Cooperative Information Systems will involve large numbers of heterogeeous, intelligent agents distributed over large communication networks. The agents may be humans, humans interacting with computers, humans working with computer support, and computer systems performing tasks without human intervention. Core technology, previously described as services provided by middleware required to support the advanced features of ICISs include (Brodie & Ceri, 1992):

- Data/object/knowledge/information managers (DBMS, OODBMS, KBMS, file systems, distributed object management).
- Presentation services/user environment - windows, forms.
- Communication infrastructures - RPC, peer-to-peer messaging, queued messaging, X-400, mail.
- Security nanagers - authentication, encryption, access control.
- Reliability managers - transaction manager, recovery manager, log manager.
- Advanced/distributed operating systems services - resource allocation.
- Naming services - global name directory and management.
- Libraries/library services.
- Control - job and request scheduling.
- Distributed computing / programming services.
- Interoperability services - information and language translation, data interchnage, information/object migration, copy management, other transparency services.
- Network services.
- General services - sorting, maths, and data conversion.
- Repository services.

Ideally, these services should be transparently available to any infformation system or IS development

environment in the distributed computing environment. Collectively, the systems providing these services are called middleware and the principle is that it should provide as many services as possible via sharable servers. Figure 7 gives an illustration of the layers of the corresponding distributed architecture.

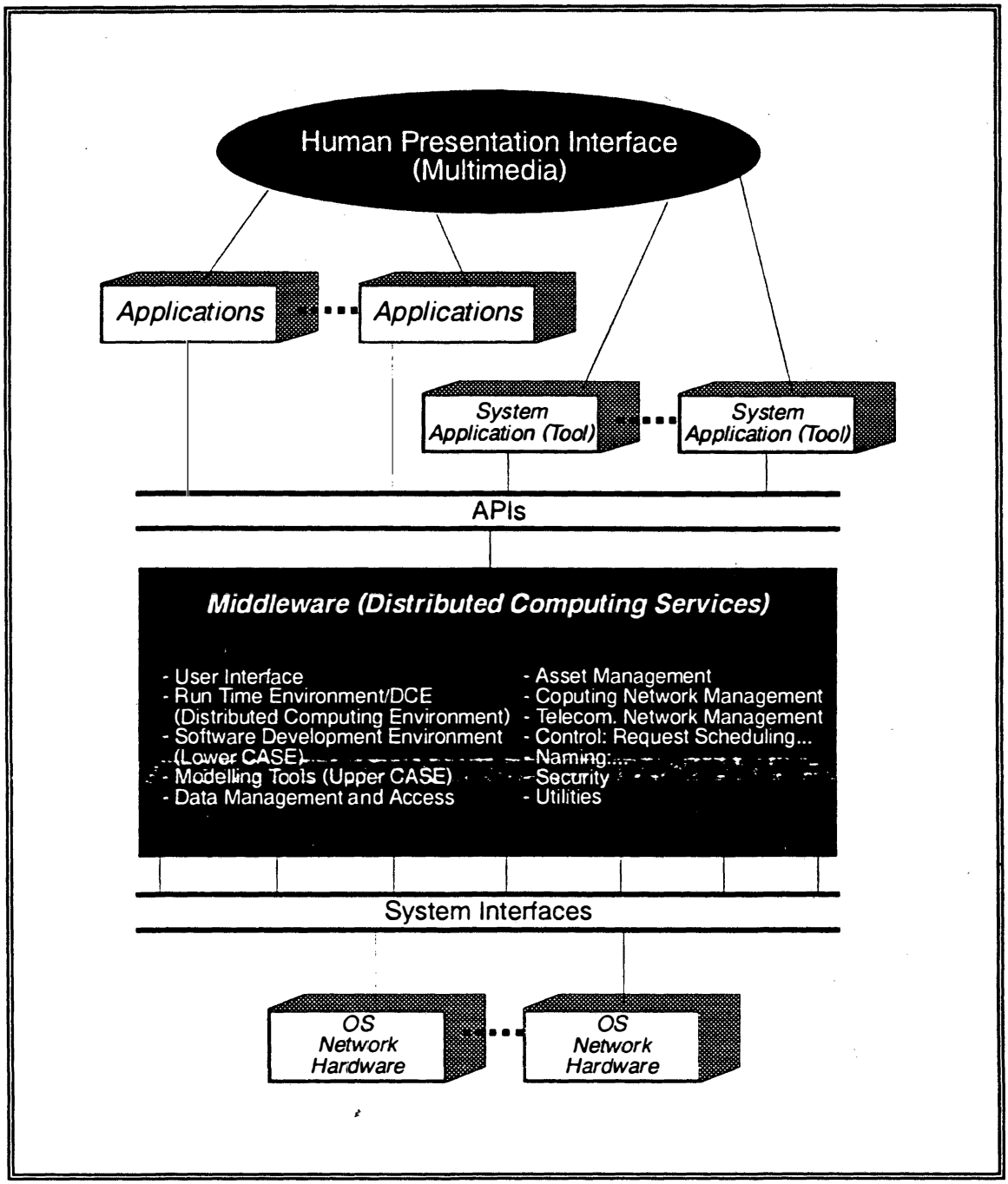

Figure 7 *Distributed Computing Architecture* with Middleware (Brodie & Ceri, 1992)

A multimedia configuration, as established by Karmouch (1993) for cooperative applications is depicted in Figure 8. Three separate networks are used - Ethernet for data, PBX for voice, and Broadband for video. When high speed networks such as FDDI, DQDB and B-ISDN become commercially available, all of the media may possibly be integrated into one network. An extension of the workstation level is a separate TV monitor to display video. A videowindow may also be integrated in the graphical screen, depending upon the image quality required for an application.

Servers may be classified to fall within two categories: information servers and communication servers. Information servers can be further classified into a database server and storage servers.

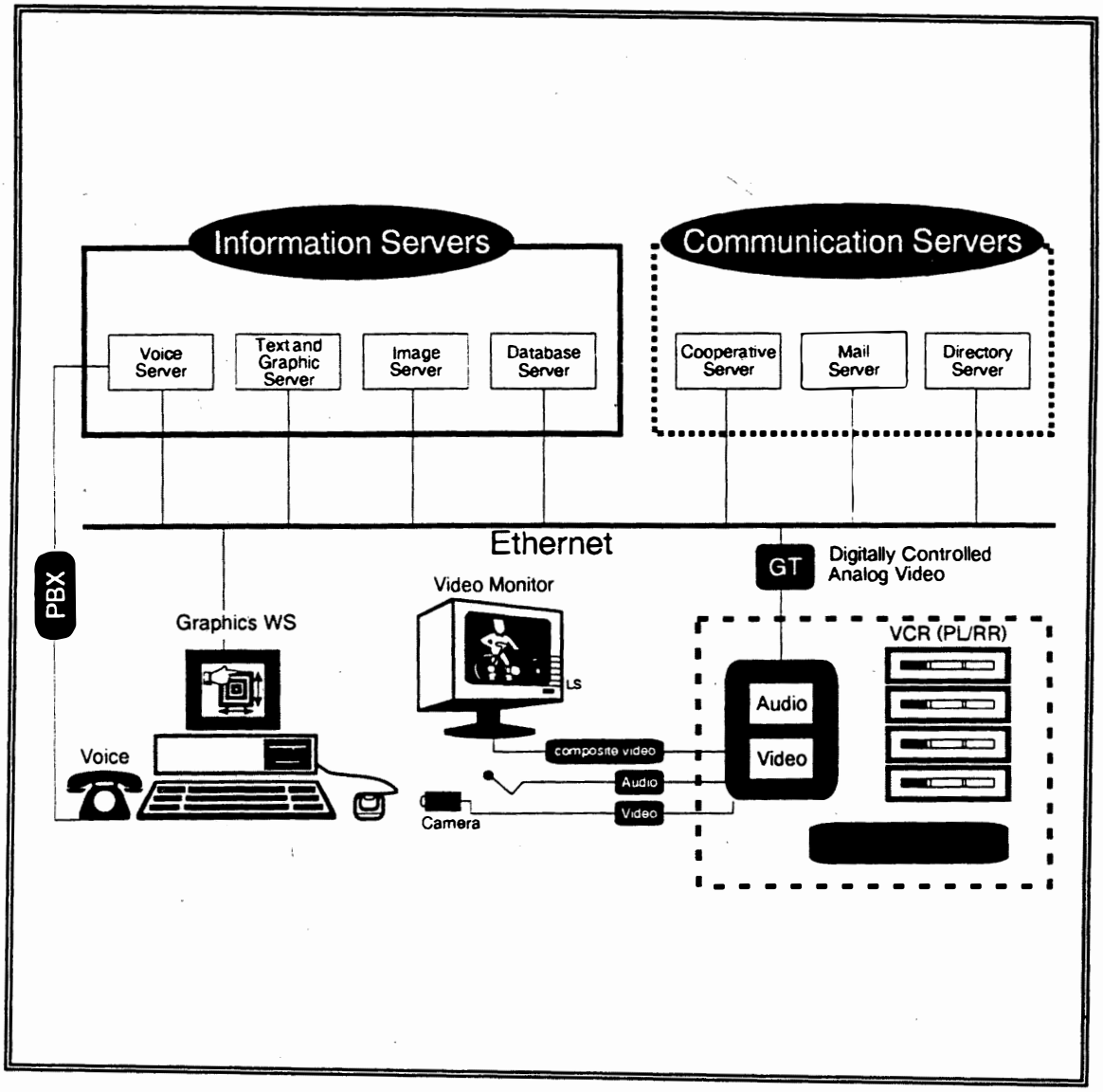

Figure 8 *Multimedia Platform for Cooperative Applications (Karmouch, 1993)*

4. Conclusion

This article reports on an investigation of how the CSCW paradigm could be incorporated into the software development process. This investigation was motivated by the fact that software development is almost always carried out by groups and that technology support is essential for a quality, on-time and within budget software project. Specific requirements for computer-aided support for the cooperative work during software development were examined. The generic facilities that software development environments should provide for supporting cooperative work were determined to form the basis for the establishment of architectures.

References

Books:

1. Booch, G. 1994. *Object-oriented Analysis and Design. With Applications.* Second Edition. Benjamin/Cummings Publishing Company, Inc.

2. Coad, P., and Yourdon, E. 1991. *Object-oriented Analysis.* Second Edition. Yourdon Press.

3. Forsyth, D.R. 1990. *Group Dynamics.* 2nd edn. Brooks/Cole Publishing Company, Pacifc Grive, CA.

4. Greif, I. 1988. *Computer-Supported Cooperative Work: A Book of Readings.* San Mateo, California: Morgan Kaufmann Publishers.

5. Jacobson, I., Christerson, M., Jonsson, P., Overgaard, G. 1992. *Object-Oriented Software Engineering.* Addison-Wesley

6. McGrath, J.E.. *Groups: Interaction and Performance.* Prentice-Hall, Inc., englewood Cliffs, NJ.

7. Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F., Lorensen, W. 1991. *Object-Oriented Modeling and Design.* Prentice-Hall International, Inc.

8. Steiner, I.D., 1972. *Group Process and Productivity.* Academic Press. New York.

Contributions in Books:

1. Steiner, I.D. 1976. Task performing groups, in *Contemporary Topics in Psychology.*, edited by J.W. Thibaut, J.T. Spence, and R.C. Carson. General Learning Press, Morristown, NJ.

Journal Articles:

1. Bannon, L. J. 1993. CSCW: An initial exploration. *Scandinavian Journal of Information Systems.* 5:3-24.

2. Bannon, L., & Schmidt, K. 1992. Taking CSCW seriously: Supporting articulation work. *Computer-Supported-Cooperative Work (CSCW).* Kluwer Academic Publishers, The Netherlands, 1:7-40. 4, 67-83.

3. Belz, F.C., Clarke, L.A., Osterweil, L., Selby, R.W., Taylor, R.N., Wileden, J.C., Wolf, A.L., Young, M. 1988. Foundations in the ARCADIA environment architecture, in *[SDE88]*, 1-13.

4. Benali, K., Boudjlida, N., Charoy, F., Derniame, F.-C., Godart, C., Griffiths, P., Gruhn, V., Jamart, P., Legait, A., Oldfield, D.E., Oquendo, F. 1990. The presentation of the ALF-Project, in eds. Madhavji, N., Schäfer, W. and Weber, H., *Proceedings of the First Conference on System Development Environments and Factories I*, 75-90.

5. Borenstein, N.S. 1992. Computational Mail as network infrastructure for Computer-Supported Cooperative Work. *Proceedings of the ACM 1992 Conference on Computer-Supported Cooperative Work.* Toronto, Canada, Oct 31-Nov 4, 67-83.

6. Brodie, M.L., & Ceri, S. 1992. On intelligent and cooperative information systems: A workshop summary. *International Journal of Intelligent and Cooperative Information Systems.* 1(2):249-289.

7. Ellis, C.A., Gibbs, S.J., & Rein, G.L. 1991. Groupware: Some issues and experiences. *Communications of the ACM.* 34(1):38-58.

8. Engeström, Y. 1990. Activity theory and individual and social transformation. *2nd International Congress for Research on Activity Theory.* Lahti, Finland, May 21-25.

9. Gaver, W.W. 1991. Sound support for collaboration. *Proceedings of the Second European Conference on Computer-Supported Cooperative Work.* Amsterdam, The Netherlands, September 25-27, 293-308.

10. Gladstein, D.L. 1984. Groups in context: A model of task group effectiveness. *Administartive Science Quarterly.* 29:499-517.

11. Green E. et al. 1991. Office systems development and gender: Implications for Computer-Supported Cooperative Work. *Proceedings of the Second European Conference on Computer-Supported Cooperative Work.* Amsterdam, The Netherlands, September 25-27, 293-308.

12. Greenbaum, J. 1988. In search of cooperation: An historical analysis of work organization and management strategies. *Proceedings of the Conference on Computer-Supported Cooperative Work.* Portland, Oregon,September 26-28, 102-114.

13. Hahn, U., Jarke, M., Rose, T. 1990. Group work in software products. *Multi-User Interfaces and Applications.* Elsevier Science Publishers, p 83-101.

14. Jacobson, I. 1987. Object-oriented development in an industrial environment. *Proceedings of OOPSLA '87.* SIGPLAN Notices, 22(12), p 183-91.

15. Jarke, M., Maltzahn, C., Rose, T. 1992. Sharing processes: Team coordination in design repositories. *International Journal of Intelligent and Cooperative Information Systems.* 1(1):145-167.

16. Joosten, S., and Brinkkemper, S. 1993. Modelling of working groups in Computer Supported Cooperative Work. *Proceedings of the 18th International Conference on Information Technologies and Programming.*

17. Kaiser, G.E., and Feiler, P.H. 1987. An architecture for intelligent assistance in software development. *Procedure of the 9th International Conference on Software Engineering.* Monterey, California, p 180-188.

18. Kaplan, S.M., Tolone, W.J. Carroll, A.M., Bogia, D.P., Bignoli, C. 1992. Supporting collaborative software development with ConversationBuilder. *Proceedings of the ACM 1992 Conference on Computer-Supported Cooperative Work.* Toronto, Canada, Oct 31-Nov 4, 11-20.

19. Karmouch, A. 1993. Multimedia distributed cooperative system. Computer Communications, 568-580.

20. Kuutti, K. 1991. The concept of activity as a basic unit of analysis for CSCW research. *Proceedings of the Second European Conference on Computer-Supported Cooperative Work.* September 25-27, Amsterdam, The Netherlands, 249-264.

21. Malone, T.W., and Crowston, K. 1990. What is coordination theory and how can it help design coopeartive work systems. *Proceedings of the Conference on Computer Supported Cooperative Work.* Los Angels, October 7-10, 357-370.

22. Marmolin, H., Sundblad, Y., Pehrson, B. 1991. An analysis of design and collaboration in a distributed environment, 147-162.

23. Nolle, T. 1993. Groupware: The next generation. *Business Communications Review,* 23(8):54-58.

24. Olson, J.S., Card, S.K, Landauer, T.K., Olson, G.M., Malone, T, Leggett, J. 1993. Computer Supported Cooperative Work: Research issues for the 90s. *Behaviour & Information Technology.* 12(2):115-129.

25. Peuschel, B., Schäfer, W., Wolf, S. 1991. A knowledge-based software development environment supporting cooperative work. *International Journal of Software Engineering . 2(1):79-106.*

26. Schmidt, K. 1991. Riding a tiger, or Computer Supported Work. *Proceedings of the Second European Conference on CSCW,* 1-16.

27. Schmidt, K., and Bannon, L. 1992. Taking CSCW Seriously: Supporting articulation work. *Computer-Supported Cooperative Work (CSCW).* 1:7-40.

28. Wilson, P. 1991. Computer Supported Cooperative Work (CSCW): origins, concepts and research initiatives. *Computer Networks and ISDN Systems.* 23:91-95.

NOTES

NOTES

NOTES