COMIC

ESPRIT BASIC RESEARCH PROJECT 6225

Computational Mechanisms of Interaction for CSCW

Abstract

The present report documents the research activities undertaken in Task 3.2 of the COMIC project.

The objective of the three years of research of Strand 3 is to develop a conceptual foundation for designing computational mechanisms of interaction for CSCW applications that can support the complex task of articulating distributed cooperative activities.

The report consists of three parts: Based on a survey of social science studies of cooperative work, Part 1 outlines a conceptual framework for the development of the concept of mechanisms of interaction. Part 3 presents the comprehensive analysis of CSCW systems (applications, shells, models) that have been analyzed in order to unravel and assess the underlying mechanisms of interaction. Part 2 brings the two bodies of evidence together, identifies requirements for mechanisms of interaction, and outlines the conceptual foundation for the design of mechanisms of interaction in CSCW systems.

Document ID	COMIC-D3.1
Status	Accepted
Type	Deliverable
Version	3.0
Date	October 4, 1993
Editors	C. Simone (Milano), K. Schmidt (Risø)
Task	3.2

Project coordinator:

Tom Rodden
Computing Department
University of Lancaster
Lancaster LA1 4YR
United Kingdom
Phone: (+44) 524 593 823
Fax: (+44) 524 593 608
Email: tom@comp.lancs.ac.uk

The COMIC project comprises the following institutions:

Gesellschaft für Mathematik und Datenverarbeitung (GMD), Bonn, Germany
Risø National Laboratory, Roskilde, Denmark
Royal Institute of Technology (KTH), Stockholm, Sweden
Swedish Institute for Computer Science (SICS), Stockholm, Sweden
Universitat Politècnica de Catalunya (UPC), Barcelona, Spain
University of Amsterdam, Amsterdam, The Netherlands
University of Lancaster, Lancaster, United Kingdom (Coordinating Partner)
University of Manchester, Manchester, United Kingdom
University of Milano, Milano, Italy
University of Nottingham, Nottingham, United Kingdom
University of Oulu, Oulu, Finland

Editors of this report:

Carla Simone
Dept. of Information Sciences
University of Milano
I-20135 Milano
Italy
Phone: (+39) 2 55006-289
Fax: (+39) 2 55006-276
Email: simone@hermes.mc.dsi.unimi.it

Kjeld Schmidt
Cognitive Systems Group
Risoe National Laboratory
P.O. Box 49
DK-4000 Roskilde, Denmark
Phone: (+45) 4677 5146
Fax: (+45) 4675 5170
Email: kschmidt@risoe.dk

ISBN 0-901800-30-9
Lancaster University, 1993
This report is available via anonymous FTP from ftp.comp.lancs.ac.uk.

Table of Contents

Computational Mechanisms of Interaction: Why, What, How, Where, Whither?

The present report documents the research activities undertaken in year 1 within Task 3.2 of the COMIC project.

The objective of the three years of research of Strand 3 is to develop a conceptual foundation for designing computational mechanisms of interaction incorporated in CSCW applications in order to support the articulation of cooperative work. Or in the words of the Technical Annex of the COMIC project:

> "The overriding objective of this workpackage is to achieve a clear understanding of the role of mechanisms of interaction in cooperative work and the requirements they must meet in terms of visibility, flexibility, etc. so as to determine the role and requirements of computational notations as means for incorporating mechanisms of interaction in CSCW applications. Based on the results of that research, computational notations for incorporating mechanisms of interaction in CSCW applications will be developed and tried out experimentally." (COMIC, 1992, p. 41).

More specifically, the objectives of Task 3.2 have been:

> "(1) to examine existing computational notations with a view to their suitability and applicability for the incorporation of mechanisms of interaction in CSCW applications and (2) to explore the feasibility of a common computational notation suitable for different types of mechanisms of interaction. This task will require an input from social science to inform the evaluation of existing notations in terms of the analysis of Task 3.1." (COMIC, 1992, p. 44).

As argued in Part 1 of the report, cooperative work is of an intrinsically distributed nature in the sense that actors may be faced with local contingencies, they may have incongruent or even complementary responsibilities, they may represent different specialized skills, they may apply different heuristics and conceptualizations, they may belong to different social worlds and so on. Because the cooperating actors are interdependent in their work, their distributed activities must be coordinated, scheduled, meshed, interrelated, integrated, in short, articulated.

In much of everyday working life, the required articulation of the distributed activities is managed so effectively and efficiently that the distributed nature of cooperative work is not apparent, most of the time. As demonstrated by the body of rich empirical studies of cooperative work reviewed and discussed in Part 1, people tacitly monitor each other; they make their activities sufficiently apperceptible for others; they take each others' past, present and prospective activities into account in planning and conducting their own work; they gesture, talk, write to each other, and so on (Harper et al., 1989; Heath and Luff, 1992; Harper and Hughes, 1993; Heath et al., 1993). Accordingly, much of the research in CSCW has focused on providing enhanced means of communication, either in order to enable actors to cooperate more effectively and efficiently in spite of geographical distance, or in order to widen the repertoire of communication facilities.

However, with the complex work environments of modern industrial and administrative organizations, the problems of articulating distributed activities are at a different order of complexity. The everyday social and communication skills are far from sufficient for articulating the cooperative efforts of hundreds or thousands of actors engaged in myriads of complexly interdependent activities, perhaps concurrently, intermittently, or indefinitely.

In such settings, the articulation of the distributed activities of cooperative work requires a certain class of symbolic artifacts that stipulate and mediate articulation work and thereby reduce the complexity of articulation work. We have chosen to call these artifacts 'mechanisms of interaction'. A mechanism of interaction can be defined as a device for reducing the complexity of articulating distributed activities of large cooperative ensembles by *stipulating and mediating* the articulation of the distributed activities.

In order to serve this function, such a device must be (1) persistent in the sense that it is accessible independently of any particular situation or individual; that is, it must exist in the form an artifact; (2) it should be a symbolic artifact in the sense that it is possible to manipulate the mechanism independently of the state of the field of work that is, the artifact must have the character of a symbolic artifact; and (3) it should provide affordances to and impose constraints on articulation work.

Such artifacts have been in use for centuries — in the form of catalogues, time tables, routing schemas, kanban systems, and so on. Now, given the infinite versatility of computer systems, it is most likely — and this is the underlying contention of the work within Strand 3 of COMIC — that computer-based mechanisms of interaction can provide a degree of visibility and flexibility to mechanisms of interaction that was unthinkable with previous technologies, typically based on inscriptions on paper or cardboard. This opens up new prospects of moving the boundary of allocation of functionality between human and artifact with respect to articulation work. The challenge is to change the allocation of functionality between human and artifact, not only so that much of the drudgery of articulation work (boring operations that have so far relied on human effort and vigilance) can be delegated to the artifact, but also, and more importantly, so that cooperative ensembles can articulate their distributed activities more effectively and with a higher degree of flexibility and so that they can tackle an even higher degree of complexity in the articulation of their distributed activities!

As a device for the articulation of cooperative work, a mechanism of interaction should be distinguished from other symbolic artifacts regulating human affairs (e.g., "Stop!", "Dead Slow!", "Pharmacy", "Wine & Spirits", "Underground", "Bus Stop"). A traffic light regulating the traffic at an intersection, for example, has certain facilities in common with a mechanism of interaction in cooperative work: It is a symbolic artifact that actively stipulates and mediates interaction. However, the drivers — as drivers — are not engaged in cooperative work; they are, rather, competing for the same resource (the road) and to each and

every one of them the others are a mere nuisance. Their interaction is ephemeral and superficial and does not entail the rich multiplicity of purposive reciprocal interactions of cooperative work.

Also, again for the sake of clarification, a mechanism of interaction should be distinguished from the information objects that are ubiquitous in cooperative work settings (such as letters, memoranda, drawings and reports, and the aggregation of such documents in the form of files and libraries) and that help to articulate and mediate distributed activities. Written documents are certainly symbolic artifacts but in so far as they serve to convey information and hence merely provide a medium of communication, they are not mechanisms of interaction. However, managing the complex flow of information objects may require mechanisms of interaction such as standardized formats (e.g., prescribed forms and routing instructions on file covers) and classification schemes (e.g., library catalogues and thesauri).

The aim of providing structured support for the articulation of distributed activities is shared by many researchers in CSCW. However, most of the mechanisms of interaction in existing CSCW systems are experienced as excessively rigid, either because the embedded mechanism of interaction is not visible and cannot be changed, e.g., THE COORDINATOR (Winograd and Flores, 1986; Flores et al., 1988), or because the facilities for changing the mechanism do not support respecification of the mechanism by the actors themselves, at the semantic level of articulation work, e.g., DOMINO (Kreifelts et al., 1991a; Kreifelts et al., 1991b). By contrast, in view of the situated character of cooperative work, mechanisms of interaction should be conceived of as local and temporary closures (Gerson and Star, 1986; Schmidt, 1991; Schmidt, 1993a). Accordingly, the primary objective of the research in Strand 3 of COMIC is to support actors in accessing and manipulating the mechanism of interaction in such a way that they are able to handle contingencies and adapt the stipulations of the mechanisms to changing requirements in their environment.

Now, a number of recent research projects attempt to make CSCW applications flexible at the level of articulation work, e.g., EGRET (Johnson, 1992) and CONVERSATIONBUILDER (Kaplan et al., 1992). While this research is very promising, the approach taken in Strand 3 of the COMIC project differs in one important respect, namely the attempt to develop a general notation that is comprehensive enough to specify any mechanism of interaction and yet at the same time supports the specification of mechanisms of interaction in terms of articulation work, by the actors themselves in a cooperative manner.

Since mechanisms of interaction are local and temporary closures, no mechanism will be able to handle all aspects of articulation work in all work domains. Particular mechanisms of interaction will invariably be designed to support cooperating actors in particular aspects of articulation work that are particularly complex and possibly even domain-specific. Thus, in order not to create artificial barriers between different aspects of articulation work, mechanisms of interaction

must be designed in such a way that they can be linked to other mechanisms, locally and temporarily.

A computational mechanism of interaction should thus be conceived of as an abstract device incorporated in a particular software application (e.g., a CASE tool, an office information system, a CAD system, a production control system, etc.) so as to support the articulation of the distributed activities of multiple actors with respect to that application. The concept of mechanisms of interaction has been developed in order to facilitate the design of domain-specific software applications in such a way that they incorporate the mechanisms of interaction as accessible and manipulable devices that support the articulation of distributed cooperative activities with respect to these applications — without imposing on actors an undue impedance between articulation work and work. As abstract devices, the mechanisms of interaction are intended to facilitate the design of domain-specific applications *in such a way* that they support the fluid interrelation of articulation work with respect to the *multiple applications* required to do the work in a particular setting, without imposing any impedance on the articulation of cooperative work with respect to different application. For instance, in the case of mechanical design, project management tools, CAD tools, process planning systems, classification schemes for common repositories (of previous designs, components, work in progress, drawings, patents), and so on. To avoid creating an impedance between the articulation of cooperative work with respect to different applications, a general notation for specifying mechanisms of interaction is required.

Two complementary approaches to the problem have been adopted. On one hand, a comparative study of empirical field study cases has been undertaken (Task 3.1). The purpose of that investigation is, in general, on the basis of available social science evidence, to develop a conceptual framework of cooperative work and its articulation that can deepen our understanding of articulation work and, in particular, to explore and develop the hypothesis of ‘mechanisms of interaction’ and to derive design requirements from the way in which symbolic artifacts are used for articulating cooperative activities in real world settings.

On the other hand, a comparative study of current CSCW systems (applications, shells, and models) has been conducted (Task. 3.2). The purpose of this investigation has been to unravel and define the mechanisms of interaction incorporated in these systems — in order to put the concept of mechanism of interaction to test and to take the initial steps towards developing a general conceptual foundation for the design of mechanisms of interaction for CSCW systems.

The whole research process has had a distinct inductive character and the two approaches have been applied in a highly interactive and iterative manner. The findings from one line of investigation have been used and, often, proved too coarse or simply inadequate when applied to the other line of investigation. Thus,

the categories and criteria applied in the analysis of current CSCW systems have been under continuous development. The work in Strand 3 has progressed roughly as follows:

Based on the initial findings from the ongoing comparative analysis of existing field studies, a tentative attempt to define mechanisms of interaction and outline the requirements was suggested by Risø in COMIC-Risø-3-1 (Schmidt, 1993b). This initial definition of mechanisms of interaction and identification requirements was then, in turn, refined and put to the test by Milano in the form of a an analysis of a number of existing CSCW systems. The categories developed by Milano and presented in COMIC-Milan-3-1 (Simone et al., 1993) was then adopted as a framework for the comprehensive analysis of mechanisms of interaction in existing CSCW systems. Altogether, circa 27 CSCW systems were analyzed — a complex cooperative task involving six sites: Lancaster, Manchester, Milano, Risø, SICS, and UPC. As the analyses of existing systems developed, a number of issues emerged, in particular the issue of the semantic level of mechanisms of interaction. This problem was addressed in COMIC-Risø-3-8 (Schmidt et al., 1993a) in which a strategy for the comparative analysis of existing computational mechanisms of interaction was outlined. This approach was implemented in COMIC-Milan-3-3 (Schmidt et al., 1993b) which was subsequently developed into the core text of the deliverable (Part 2).

The structure of the deliverable reflects the dual approach of the research. Part 1 presents the comparative study of a number of empirical studies of cooperative work and outlines the conceptual framework based upon which the concept of computational mechanisms of interaction has been developed. The text is an revised version of the internal report from Task 3.1 from February 1993 (COMIC-Risø-3-3).

Part 3 presents the comprehensive analysis of CSCW artifacts. Altogether, some 27 CSCW systems (applications, shells, models) have been analyzed thoroughly in order to unravel and assess the underlying mechanisms of interaction. All of the analyses are based on first-hand experience with the systems as well as the available documentation.

Part 2 brings the two bodies of evidence together, identifies requirements for mechanisms of interaction, and outlines the conceptual foundation for the design of mechanisms of interaction in CSCW systems. Part 2 thus represents the core text of the deliverable. For this reason, Part 2 has been written in such a way that it can be read independently of the two other parts.

Some of the initial assumptions of the research in Strand 3 broke down rather early. In particular:

- The original plans for Strand 3, as expressed in the Technical Annex, did not make a clear distinction between mechanisms of interaction and notations. It was believed that the required facilities for actors' access and manipulation of the mechanism would somehow be provided by the

notation itself (COMIC, 1992, p. 44). As a result of the initial work in Strand 3, it was soon realized that the very concept of mechanisms of interaction should be taken more seriously — in the sense that computational mechanisms of interaction should be conceived of as *abstract software devices* composed not only of a notation but also of a set of facilities for access and manipulation of the mechanisms.

- At the outset, the issue of the semantic level of the mechanism of interaction was not recognized as particularly problematic. The problem became evident to us, however, with the publication of the work on OVAL by Malone and associates (Malone et al., 1992). Again, the implications of realizing this, was that it was necessary to pay far more attention to the very concept of mechanism of interaction and its role in the architecture of CSCW systems, as opposed to solely and directly addressing the issues of notation .
- It was also believed, initially, that computational mechanisms of interaction could be given the required visibility and flexibility by means of one notation. In the course of the research in the first year, we concluded that multiple notations are needed in order to provide the required flexibility and generality.

Consequently, the original plan of addressing the issue of notations and their characteristics forthwith and directly was felt to be unrealistic. In stead of restricting the research to notations that were readily available, the research focused on a comprehensive comparative analysis of existing CSCW systems, irrespective of whether they provided any notation in the form of a specification language, in order to unravel and characterize the underlying mechanisms of interaction through a process similar to ‘reverse engineering’.

The comparative analysis of existing CSCW systems (combined with analysis of empirical studies of cooperative work in real world settings) helped us to make major progress in terms of developing the concept of mechanisms of interaction into an operational concept.

All in all, substantial progress has been achieved in the first year’s research within Strand 3.

1. By grounding the sociological concept of articulation work in a conception of cooperative work as constituted by mutual dependence in and through a common field of work (Part 1), articulation work has been made a crucial concept for defining mechanisms of interaction. A mechanism of interaction is thus defined as a device for reducing the complexity of articulating distributed activities of large cooperative ensembles by *stipulating and mediating* the articulation of the distributed activities.

2. Salient dimensions of articulation work has been identified in order to provide a preliminary basis for determining the semantic level required of notations for designing mechanisms of interaction and offers a tentative set of

categories of primitives of the generic notation for computational mechanisms of interaction.

3. A conceptual framework has been developed for conceptualizing the requirements of computational mechanisms of interaction in a systematic way as well as for designing and evaluating computational mechanisms of interaction. According to this framework, a computational mechanism of interaction should be conceived of as an abstract device incorporated in a software application so as to support the articulation of the distributed activities of multiple actors with respect to that application and the field of work it represents. This framework distinguishes two complementary constitutive aspects of a computational mechanism of interaction: notations (Section 2.3) and facilities (Section 2.5).

4. The research has established that multiple notations are required for controlling and specifying the behavior of the mechanism. At this point, three notations are hypothesized to be required and sufficient: In addition to the notation incorporated in the mechanism of interaction (the α notation), a β notation for specifying and respecifying the α notation by means of available domain-specific primitives and rules (for instance, reallocating a certain type of task to another type of actor); and a γ notation for specifying and respecifying the β notation (for instance, defining a new class of tasks). The β and γ notations are required in order to make mechanisms of interaction sufficiently flexible; in addition, the γ notation is required to support the fluid meshing of different aspects of articulation work by enabling mechanisms to be linked.

5. Based on the comparative analysis of social science studies of cooperative work (Part 1), the comprehensive analysis of existing CSCW systems (Part 3) resulted in the identification and initial development of a number of categories of requirements with respect to actors' ability to access and manipulate computational mechanisms: support of user control of execution of the mechanism; support of redesign of the mechanism by actors; making the mechanism visible to actors; support of governing the propagation of changes to the mechanism; support of making the history of the evolution of a mechanism available; support of maintaining consistency between mechanisms and the organizational context; and support of linking different mechanisms in different applications.

6. A less tangible but not less important result is the very fact that we have succeeded, in the work in year one of Strand 3, to bridge the conceptual gap between social science and computer science. (Sure, it is a ramshackle makeshift construction, but it did enable us to cross the gap!). Based on an analysis of empirical sociological field studies, a conceptual framework was been developed that was then applied in a comprehensive analysis of existing CSCW systems. Based on these bodies of empirical and theoretical analysis, the first steps were taken towards turning the concept of mechanisms of interaction into an operational design construct.

The research on computational mechanisms of interaction is still in its early stages and some fairly big problems are as yet unresolved. The following topics of future work should be highlighted here:

1. A deeper understanding of articulation work in general and in particular the use of symbolic artifacts to stipulate and mediate articulation work is required. To obtain that, new field studies have been undertaken at Risø in order to specifically investigate mechanisms of interaction in large scale cooperative work settings such as production and distribution of technical documentation in manufacturing, production control in flexible manufacturing, and concurrent design in the manufacturing of electronic equipment. Furthermore, Manchester will supply field studies of articulation work on design and development projects and on articulation work in a high-tech sales force. Finally, as part of Lancaster's work for COMIC are case studies of document use to inform the design of the Shared Object Service in Strand 4. This work will, because of its ethnographic orientation, also provide significant input to the case studies which inform the work of Strand 3.

2. The three notations required of a mechanism of interaction have only been identified and loosely described. They need to be developed into rigorous design devices. Thus, for the next year, a set of experimental studies of designing α and β notations for specific application domains will be undertaken. Based on current field studies, the Risø group will be developing mechanisms of interaction for supporting cooperative work in the production and distribution of technical documentation in manufacturing, in production control in flexible manufacturing, and in concurrent design in the manufacturing of electronic equipment. With the same general objective, Milano will be working on the mechanisms of interaction supporting work articulation with respect to the task, actor/role, and resource dimensions.

3. The relationships between the different notations must be explored in detail. The requirements for each the three notations and for the representations that are necessary for the implementation of the required facilities of access and manipulation must be specified.

With the objective of identifying general features and requirements of languages suitable for the description, specification, and analysis of computational mechanisms of interaction a number of experiments will be performed. A large body of formal calculi and models may be relevant. Some key features of mechanisms of interaction, however, challenge the functionality of many of these models in interesting ways, including aspects of information access and visibility, and of dynamic reconfigurability, prompted by user intervention or changes in the work setting. Of particular interest from this perspective is the π-calculus of Milner, Parrow, and Walker (Milner et al., 1992). This calculus has recently been proposed explicitly to support the modelling of systems with dynamically changing interconnection structure. Stockholm will be investigating its suitability as a foundation for the description of mechanisms of interaction, in the first hand through case studies.

4. The γ notation is an especially crucial and tough issue. With the identification of salient dimensions of articulation work, the semantic level of articulation work has been loosely defined. Still, a far more exact definition of what constitutes the semantic level of articulation work is required. With this objective, Stockholm will be exploring the relationships between the notion of semantic level and the notion of the 'knowledge level' suggested by Newell and later expounded within the field of knowledge representations.

5. The identification of salient dimensions of articulation work can be taken as a tentative set of categories of primitives for the γ notation. However, in order to develop a γ notation that can be tried out and evaluated empirically, additional empirical studies are required. First, field studies that systematically investigate the use of symbolic artifacts in articulation work are required in order to refine the conception of these dimensions and their relationships and thereby provide a proper empirical basis for the development of the γ notation. Second, experimental development and evaluation of mechanisms of interaction for different domains is required in order to explore the problems of interfacing and linking different mechanisms of interaction and the α and β notations they embody. The planned studies at Milano and Risø are intended to do just that. That is, the strategy for developing and evaluating the γ notation is empirical and experimental.

6. The issues of linking mechanisms of interaction will also be explored at UPC from a somewhat different but compatible approach. The UPC group plans to study and produce demonstrators on the computing support for embedding mechanisms of interaction and appropriate notations, with emphasis on large scale, pluri-organizational cooperative arrangements. On one hand, UPC will explore the conceptual commonalties and differences between the current work on Open Distributed Processing (ODP) and the concepts of mechanisms of interaction and notations. The goal is to provide support for the salient dimensions of articulation work in large scale distributed systems. This is going to be demonstrated in a prototype distributed computing platform. On the other hand, UPC will investigate how cooperation occurs across organizational boundaries and how mechanisms of interaction deal with boundaries. This objective is to provide support for articulation work across boundaries. This issue is related to the work on ODP modelling and Enterprise Integration.

Finally, there are important issues to be investigated with respect to the relationships between the concept of mechanisms of interaction and the conceptual work in the other strands.

In conjunction with the research in Strand 2, it is obviously worth exploring whether and how the conceptual framework being developed in Strand 3 can be applied in ethnographic and other empirical studies of cooperative work with a view to designing CSCW systems, especially as a means for sensitizing analysts

to aspects of primary importance to the design of CSCW systems and as a means for conceptualizing findings.

Mechanisms of interaction are always managed under the constraints of the policies dictated by the organizational context. In providing actors with means of accessing and manipulating mechanisms of interaction, policy issues clearly need to be taken into account in the design of the various facilities of the mechanisms. Thus, mechanisms of interaction need to interface with the facilities for supporting the representation of organizational context that are being investigated in Strand 1. On the other hand, the concept of mechanisms of interaction seem necessary as a way of handling the distributed cooperative management of the large-scale common repositories that characterize organizational life. Steps have been taken to explore these commonalties in a joint workshop.

The issue of visualizing the constraints and affordances offered by mechanisms of interaction are closely related to important aspects of the work on metaphors in Strand 4, especially the notion of 'boundaries'. It is, for instance, conceivably possible to visualize the stipulations of a mechanism spatially as paths of different kinds and similarly that the various aspects of control of execution can be visualized as different kinds of 'permeable boundaries'. The prospects are quite interesting.

Publications

During year 1 of the work in Strand 3, the following papers have been published:

Divitini, M., G. Omodei Salé, A. Pozzoli, C. Simone: "Supporting the dynamics of knowledge sharing within organizations", *COOCS - ACM Conference on Organizational Computing Systems, Milpitas, California, November 1-4, 1993*.

Schmidt, Kjeld: "Mechanisms in cooperative work: Preliminary observations," in *MOHAWC Separate Papers*, ed. by B. Brehmer, vol. 2, Risø National Laboratory, 1992, pp. 7-40.

Schmidt, Kjeld: "The Articulation of Cooperative Work — Requirements for Computer Support," in *Developing CSCW Systems: Design Concepts. Report of CoTech WG4, February 1993*, ed. by K. Schmidt, Risø National Laboratory, Roskilde, Denmark, 1993, pp. 37-104.

Schmidt, Kjeld: "Cooperative work and its articulation: Support requirements," *Travail Humain*, 1993. - Forthcoming.

Schmidt, Kjeld, and Tom Rodden: "Putting it all together: Requirements for a CSCW platform," *Design of Computer Supported Cooperative Work and Groupware Systems. 12th International Workshop on "Informatics and Psychology", Schärding, Austria, June 1-3, 1993*, 1993. - Forthcoming.

Simone, C.: "Supporting collaborative dialogues in distance learning", *NATO Workshop on Dialogues in Distance Learning, Segovia, 1993* (to appear in NATO/ASI Series, Springer Verlag).

Simone, C.: "Evaluation of work processes by modular Petri Nets", *Workshop on CSCW, Petri Nets and related formalisms, Chicago, 1993*.

References

COMIC: *The COMIC Project (Computer-based Mechanisms of Interaction in Cooperative Work). Esprit Basic Research Project No 6225. Technical Annexe*, Lancaster University, 27 April, 1992.

Flores, Fernando, Michael Graves, Brad Hartfield, and Terry Winograd: "Computer Systems and the Design of Organizational Interaction," *ACM Transactions on Office Information Systems*, vol. 6, no. 2, April 1988, pp. 153-172.

Gerson, Elihu M., and Susan Leigh Star: "Analyzing Due Process in the Workplace," *ACM Transactions on Office Information Systems*, vol. 4, no. 3, July 1986, pp. 257-270.

Harper, Richard H. R., and John A. Hughes: "What a f-ing system! Send 'em all to the same place and then expect us to stop 'em hitting. Managing technology work in air traffic control," in *Technology in Working Order. Studies of work, interaction, and technology*, ed. by G. Button, Routledge, London and New York, 1993, pp. 127-144.

Harper, Richard R., John A. Hughes, and Dan Z. Shapiro: *The Functionality of Flight Strips in ATC Work. The report for the Civil Aviation Authority*, Lancaster Sociotechnics Group, Department of Sociology, Lancaster University, January, 1989.

Heath, Christian, Marina Jirotka, Paul Luff, and Jon Hindmarsh: "Unpacking Collaboration: the Interactional Organisation of Trading in a City Dealing Room," in *ECSCW '93. Proceedings of the Third European Conference on Computer-Supported Cooperative Work, 13-17 September 1993, Milan, Italy*, ed. by G. De Michelis, C. Simone and K. Schmidt, Kluwer Academic Publishers, Dordrecht, 1993, pp. 155-170.

Heath, Christian, and Paul Luff: "Collaboration and Control. Crisis Management and Multimedia Technology in London Underground Control Rooms," *Computer Supported Cooperative Work (CSCW). An International Journal*, vol. 1, no. 1-2, 1992, pp. 69-94.

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*, ed. by J. Turner and R. Kraut, ACM Press, New York, 1992, pp. 298-305.

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*, ed. by J. Turner and R. Kraut, ACM Press, New York, 1992, pp. 378-385.

Kreifelts, Thomas, Elke Hinrichs, Karl-Heinz Klein, Peter Seuffert, and Gerd Woetzel: "Experiences with the DOMINO Office Procedure System," in *ECSCW '91. Proceedings of the Second European Conference on Computer-Supported Cooperative Work*, ed. by L. Bannon, M. Robinson and K. Schmidt, Kluwer Academic Publishers, Amsterdam, 1991a, pp. 117-130.

Kreifelts, Thomas, Frank Victor, Gerd Woetzel, and Michael Woitass: "Supporting the design of office procedures in the DOMINO system," in *Studies in Computer Supported Cooperative Work. Theory, Practice and Design*, ed. by J. M. Bowers and S. D. Benford, North-Holland, Amsterdam etc., 1991b, pp. 131-144.

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*, ed. by J. Turner and R. Kraut, ACM Press, New York, 1992, pp. 289-297.

Milner, R., J. Parrow, and D. Walker: "A Calculus of Mobile Processes, I-II," *Information and Computation*, vol. 100, no. 1, 1992, pp. 1-40, 41-77.

Schmidt, Kjeld: "Riding a Tiger, or Computer Supported Cooperative Work," in *ECSCW '91. Proceedings of the Second European Conference on Computer-Supported Cooperative Work*, ed. by L. Bannon, M. Robinson and K. Schmidt, Kluwer Academic Publishers, Amsterdam, 1991, pp. 1-16.

Schmidt, Kjeld: "Cooperative work and its articulation: Support requirements," *Travail Humain*, 1993a. - Forthcoming.

Schmidt, Kjeld: *Modes and Mechanisms of Interaction in Cooperative Work: Outline of a Conceptual Framework*, COMIC Internal Report, Risø National Laboratory, Roskilde, Denmark, (version 1.1), February, 1993b. [COMIC-Risø-3-3].

Schmidt, Kjeld, Peter Carstensen, and Betty Hewitt: *Evaluating Computational Notations of Mechanisms of Interaction: Notations, Dimensions, and Features*, Risø National Laboratory, Roskilde, Denmark, May, 1993a. [COMIC-Risø-3-8].

Schmidt, Kjeld, Carla Simone, Peter Carstensen, and Betty Hewitt: *Computational Notations of Mechanisms of Interaction: Dimensions, Features and Requirements*, Dipartimento di Scienze dell'Informazione, Milano, Italy, (Version 1.0), July 7, 1993b. [COMIC-MILAN-3-3].

Simone, C., A. Grasso, and A. Pozzoli: *Mechanisms of interaction based on conversations*, Dipartimento di Scienze dell'Informazione, (Version 1.0), 8 February, 1993. [COMIC-MILAN-3-1].

Winograd, Terry, and Fernando Flores: *Understanding Computers and Cognition: A New Foundation for Design*, Ablex Publishing Corp., Norwood, New Jersey, 1986.

Part 1

Social mechanisms of interaction

Table of Contents, Part 1

Modes and mechanisms of interaction in cooperative work*

Kjeld Schmidt

Risø National Laboratory

1.1. Introduction

In the design of conventional computer-based systems for work settings the core issues have been to develop effective computational models of pertinent structures and processes in the field of work (data flows, conceptual schemes, knowledge representations) and adequate modes of presenting and accessing these structures and processes (user interface, functionality). While these systems, more often than not, were used in cooperative work settings and even, as in the case of systems that are part of the organizational infrastructure, were used by multiple users (e.g., database systems), the issue of *supporting the articulation of cooperative work by means of such systems* has not been addressed directly and systematically, as an issue in its own right. If the underlying model of the structures and processes in the field of work was 'valid', it was assumed that the articulation of the distributed activities was managed 'somehow'. It was certainly not a problem for the designer or the analyst.[1]

In so far as CSCW can be conceived of as an endeavor to understand the nature and support requirements of cooperative work arrangements with the objective of designing computer-based technologies for such arrangements (Schmidt and Bannon, 1992), CSCW can be taken as a complete overturn of this paradigm (Hughes et al., 1991).

The general objective of the present report is to outline a conceptual framework for the analysis of cooperative work that will assist analysts and designers in several ways, e.g.:

- provide heuristic guidance to analysts so as to assist them in conducting cost-efficient empirical field studies with a view to identifying system requirements;

* The framework outlined in this paper has been developed in recurrent debates with a number of colleagues in the COMIC project: Liam Bannon, John A. Hughes, Tom Rodden, Dan Shapiro, and Carla Simone as well as my colleagues at Risø: Hans Andersen, Peter Carstensen, Betty Hewitt, Bjarne Kaavé, and Carsten Sørensen. In addition, I would like to thank Christian Heath for constructive comments to an early draft and Dave Randall for many, always rambling but always highly stimulating and encouraging, discussions during his stay as a visiting researcher at Risø in the Spring of 1993. — Useful initial discussions of the framework were facilitated by the CoTech network, in particular Working Group 4; the formation of this network was supported by ESPRIT Basic Research.

1 A similar point was made very early in CSCW by Anatol Holt: "Whatever has to do with task interdependence — *coordination* — is left to the users to manage as best they can, by means of shared databases, telephone calls, electronic mail, files to which multiple users have access, or whatever ad hoc means will serve." (Holt, 1985).

- support the comparison and transfer and thus the generalization of findings from different empirical field studies;
- provide a frame of reference for conceptualizing, comparing, and transferring user experience with particular CSCW systems;
- assist analysts in conceptualizing empirical findings in terms of requirements, i.e., in a format that is compatible with the issues and concerns of software engineering;
- identify distinct classes of support requirements so as to provide a rational basis for developing architectures for CSCW systems.

More specifically, the objective is to investigate the different roles of *modes of interaction* and *mechanisms of interaction* in the articulation of cooperative work so as to identify the different support requirements of modes and mechanisms of interaction. In order to do this, the report collates, discusses, and compares a number of empirical field studies which have been reported in the literature or in which the author has been personally involved.

The bulk of the report is therefore devoted to a number of case studies of cooperative work in different work domains.

1.2. Cooperative work

The term ‘cooperation’ has a wide variety of connotations in everyday usage, ranging from notions of joining alliances (as in the ‘cooperative’ movement) and being amicable and altruistic (‘You should be more cooperative’) to actually working together in producing a product or service irrespective of whether those working together are allies or friends.

In some areas of social research, in particular political science, institutional economics, and organizational theory, the term ‘cooperation’ has been used broadly to designate the formation of coalitions between actors with partially divergent interests and motives. For instance, in his influential investigation of institutional economics, John Commons uses the term in the strong sense of subjection of the centrifugal forces of conflicting individual interests to a putative common cause and collective action:

> “coöperation […] arises from the necessity of *creating a new harmony* of interests — or at least order, if harmony is impossible — out of the conflict of interests among the hoped-for coöperators. It is the negotiational psychology of persuasion, coercion, or duress. The greatest American piece of actual coöperation, latterly under ill repute [anno 1934], is the holding companies which suppress conflicts, if persuasion proves inadequate. A more universal coöperation, suppressing conflict in behalf of order, is proposed by Communism, Fascism, or Nazism. These have found their own ways of submerging conflicts of interest.” (Commons, 1934, pp. 6 f.)

The conception of ‘cooperation’ as a governance structure for curbing opportunistic behavior among actors does not provide an adequate approach to CSCW. Of course, opportunistic behavior is part and parcel of working life, under the

auspices of "common ownership" as well as on the "open market". In designing CSCW system this fact of life must certainly be taken into account (Kling, 1980; Grudin, 1989; Orlikowski, 1992). But if this conception is taken to be provide the general conceptual framework for CSCW, essential aspects of the multi-faceted phenomenon of cooperative work is marginalized or simply lost: the work itself, the complex material interdependencies between actors, the role of artifacts in mediating interactions and the different affordances and constraints of different artifacts in that respect, the multifarious technical and social skills required, the continuous effort of maintaining mutual awareness and making one's own activities publicly visible, the mutual help.

In other words, the concept of 'cooperation' does not enable us to grasp the rich multiplicity of interdependency and reciprocity among actors in cooperative work arrangements. It only allows us to conceive of a world of partially conflicting and mutually repellent actors whose only interactions take the abstract form of allocations of resources.

On the other hand, however, the term 'cooperative work', chosen by Greif and Cashman to designate the object domain of the new R&D area of CSCW, also happens to be a term with a long history in the social sciences. It was used as early as the first half of the 19th century by economists such as Ure (1835) and Wakefield (1849) as the general and neutral designation of work involving multiple actors and was further developed by Marx (1867) who defined it as "multiple individuals working together in a conscious way [planmässig] in the same production process or in different but connected production processes." In this century, the term has been used extensively with the same general meaning by various authors, especially in the German tradition of the sociology of work (Popitz et al., 1957; Bahrdt, 1958; Dahrendorf, 1959; Kern and Schumann, 1970; Mickler et al., 1976, for example).

This concept of 'cooperative work' is, surely, the appropriate starting point for developing a conceptual framework of cooperative work for CSCW systems design (Bannon and Schmidt, 1989).

At the core of this conception of cooperative work is the notion of *interdependence in work*. in the sense that *cooperative work occurs when multiple actors are required to do the work and therefore are mutually dependent in their work and must coordinate and integrate their individual activities to get the work done* (Schmidt, 1991). We will discuss the notion of interdependence at length below. First, however, we need to discuss why we need to distinguish cooperative work from work in general — in view of the fact that all work is essentially social.

According to Montesquieu, "Man is born in society and there he remains."[1] In the same vein, Marx (1857) posited that

---

1 Actually, Montesquieu does not quite put it this way: "Je n'ai jamais ouï parler du droit public qu'on n'ait commencé par rechercher soigneusement quelle est l'origine des sociétés, ce qui me paraît ridicule. Si les hommes n'en formaient point, s'ils se quittaient et se fuyaint les uns les autres, il faudrait en demender la raison et chercher pourquoi ils se tiennent séparés. Mails ils naissent tous liés les uns aux autres; uns fils est né auprès de son père, et il s'y tient: voila la societé et la cause de la

> "Individuals producing in society — hence socially determined individual production — is, of course, the point of departure. The individual and isolated hunter and fisherman, with whom Smith and Ricardo begin, belong among the unimaginative conceits of the eighteenth-century Robinsonades."

Marx' critique of the Robinson Crusoe metaphor is rooted in a conception of work as an intrinsically social phenomenon:

> "Production by an isolated individual outside society — a rare exception which may well occur when a civilized person in whom the social forces are already dynamically present is cast by accident into the wilderness — is as much of an absurdity as is the development of language without individuals living *together* and talking to each other." (Marx, 1857)

In work, that is, the social setting is ubiquitous. Work is always immediately social in that the object and the subject, the end and the means, the motives and the needs, the implements and the competencies, are socially mediated. The social nature of work is not a static property, however; it develops historically. With the ever deeper and increasingly comprehensive social division of labor, the subject and object of work, etc. become increasingly social in character. Hunter-gatherers, for instance, work in an environment that is appropriated socially and yet to a large extent naturally given, whereas, in the case of operators in modern chemical plants, every aspect of work is socially mediated — to the extent that it is conducted in an 'artificial reality'.

While work is always socially situated and socially organized, the very work process is not always cooperative in the sense that it requires and involves multiple actors who are thus interdependent in their work.

Now, in cooperative work settings, cooperative and individual activities are inextricably interwoven. Cooperative work is always conducted by individuals (albeit interdependently and hence concertedly), and yet, individual activities are always penetrated and saturated by cooperative work as by a social 'ether' — so that, in any given case, it may be impossible to determine whether a given activity is part of a cooperative activity (Hughes et al., 1991; Heath and Luff, 1992).

So, why make the distinction? Because, if actors are interdependent in their work, then they objectively need to coordinate and integrated their individual activities to get the work done.

Work does not always involve multiple people that are mutually dependent in their work and therefore required to coordinate and integrate their individual activities. We are social animals, but we are not *all* of us *always* and in *every* respect mutually dependent in our work. Thus, in spite of its intrinsically social nature, work is not intrinsically cooperative in the sense that actors are mutually dependent in their work. As observed by Popitz and associates in their classic work:

> "It is not sufficient to remark that the individual work activities are embedded within a larger work context. One must be more concrete and with each individual work activity demonstrate *how* and *to what extent* cooperation with other work activities is a requirement. In doing so, the following issues seem to be important: Is a work activity determined by other work activities

---

société." (Montesquieu, 1721, Lettre xciv, p. 153)— The wording of the quote is Ferguson's apt rendition (Ferguson, 1767, p. 16).

> and does it, on its part, determine others? What kind of dependency? How does it show? Furthermore: Does a certain work activity require assistance or not? Is mutual assistance possible or even necessary? Or is each worker so preoccupied with his own work that a mutual support is not possible?"(Popitz et al., 1957, p. 41)

Consequently, if actors are not mutually dependent in their work and therefore required to coordinate and integrate their individual activities, they may not need the support of CSCW systems. In that case, a 'shared' information system is merely a pooled resource in the sense that it is provided under the auspices of the 'common ownership' of a firm. The actors may — *or may not* — find it in their individual interests to actually 'share' this resource, for instance by providing information to others via the system (Orlikowski, 1992).

That is, as soon as we abandon the specific notion of cooperative work as constituted by interdependent activities for the notion of the social nature of all work, we are back with the issues of designing information systems in general.

Let us therefore explore the concept of cooperative work a little further.

1.2.1. The emergent nature of cooperative work

Generally speaking, cooperative work relations are formed because of the limited capabilities of single human individuals, that is, because the work could not be accomplished otherwise, or at least could not be accomplished as quickly, as efficiently, as well, etc., if it was to be done on an individual basis:

> "If we eliminate from consideration personal satisfaction […], their coöperation has no reason for being except as it can de what the individual cannot do. Coöperation justifies itself, then, as a means of overcoming the limitations restricting what individuals can do." (Barnard, 1938, p. 23)

More specifically, cooperative work arrangements emerge in response to different requirements and may thus serve different generic functions (Schmidt, 1990):

Augmentation of capacity: A cooperative work arrangement may simply augment the mechanical and information processing capacities of human individuals and thus enable a cooperating ensemble to accomplish a task that would have been infeasible for the actors individually. As an ensemble they may, for instance, be able to remove a stone that one individual could not move one iota. In the words of John Bellers: "As one man cannot, and 10 men must strain, to lift a tun of weight, yet one hundred men can do it only by the strength of a finger of each of them." (Bellers, 1696, p. 21). This is cooperative work in its most simple form. By cooperating, they simply augment their capacity: "With simple cooperation it is only the mass of human power that has an effect. A monster with multiple eyes, multiple arms etc. replaces one with two eyes etc." (Marx, 1861-63, p. 233□)

Differentiation and combination of specialties: A cooperative work arrangement may combine multiple *technique-based specialties*. In augmentative cooperation the allocation of different tasks to different actors is incidental and temporary; the participants may change the differential allocation at will. By contrast,

technique-based specialization requires an "exclusive devotion" to a set of techniques (de Tracy, 1826, p. 79). That is, as opposed to the contingent and reversible differentiation of tasks that may accompany augmentative cooperation, the *technique-based specialization is based on an exclusive devotion to a repertoire of techniques*. In the words of the eulogist of technique-based specialization, Adam Smith: "the division of labour, by reducing every man's business to some one simple operation, and by making this operation the sole employment of his life, necessarily increases very much the dexterity of the workman" (Smith, 1776, p. 7). The different techniques must be combined, however, and the higher the degree of technique-based specialization, the larger the network of cooperative relations required to combine the specialties (Babbage, 1832, §§ 263-268, pp. 211-216). That is, *technique-based specialization requires combinative cooperation*. This combinative cooperation is defined by Marx as "cooperation in the division of labor that no longer appears as an aggregation or a temporary distribution of the same functions, but as a decomposition of a totality of functions in its component parts and unification of these different components" (Marx, 1861-63, p. 253). Hence, the combination of multiple technique-based specialties assumes the character of a mechanical totality in which the human actors are assigned the role of a component. In the words of Ferguson's classic denunciation of this kind of division of labor: "Manufactures […] prosper most, where the mind is least consulted, and where the workshop may, without any great effort of imagination, be considered as an engine, the parts of which are men." (Ferguson, 1767, p. 183)

Mutual critical assessment: A cooperative work arrangement may facilitate the application of multiple problem-solving *strategies and heuristics* to a given problem and may thus ensure relatively balanced and objective decisions in complex environments. Under conditions of uncertainty decision making will require the exercise of discretion. In discretionary decision making, however, different individual decision makers will typically have preferences for different heuristics (approaches, strategies, stop rules, etc.). Phrased negatively, they will exhibit different characteristic 'biases'. By involving different individuals, cooperative work arrangements in complex environments become arenas for different decision making strategies and propensities where different decision makers subject the reliability and trustworthiness of the contributions of their colleagues to critical evaluation (Schmidt, 1990). As an ensemble they are thus able to arrive at more robust and balanced decisions. For example, take the case of an "experienced and skeptical oncologist," cited by Strauss and associates:

> "I think you just learn to know who you can trust. Who overreads, who underreads. I have got X rays all over town, so I've the chance to do it. I know that when Schmidt at Palm Hospital says, 'There's a suspicion of a tumor in this chest,' it doesn't mean much because she, like I, sees tumors everywhere. She looks under her bed at night to make sure there's not some cancer there. When Jones at the same institution reads it and says, 'There's a suspicion of a tumor there,' I take it damn seriously because if he thinks it's there, by God it probably is. And you do this all over town. Who do you have confidence in and who none." (Strauss et al., 1985)

The point is, as observed by Cicourel (1990, p. 222), that "the source of a medical opinion remains a powerful determinant of its influence." That is, "physicians typically assess the adequacy of medical information on the basis of the perceived credibility of the source, whether the source is the patient or another physician." Thus "advice from physicians who are perceived as 'good doctors' is highly valued, whereas advice from sources perceived as less credible may be discounted." This process of mutual critical evaluation was described by Cyert and March (1963) who aptly dubbed it 'bias discount.' Even though dubious assessments and erroneous decisions due to characteristic individual biases are transmitted to other decision makers, this does not necessarily entail a diffusion or accumulation of mistakes, misrepresentations, and misconceptions within the decision-making ensemble. The cooperating ensemble establishes a negotiated order.

Confrontation and combination of perspectives: A cooperative work arrangement may finally facilitate the application of multiple *perspectives* on a given problem so as to match the multifarious nature of the field of work. A perspective, in this context, is a particular — local and temporary — conceptualization of the field of work, that is, a conceptual reproduction of a limited set of salient structural and functional properties of the field of work, such as, for instance, generative mechanisms, causal laws, and taxonomies, and a concomitant body of representations (models, notations, etc.).

To grasp the diverse and contradictory aspects of the field of work as a whole, the multifarious nature of the field of work must be matched by a concomitant multiplicity of perspectives on the part of the cooperating ensemble (Schmidt, 1990). The application of multiple perspectives will typically require the joint effort of multiple agents, each attending to one particular perspective and therefore engulfed in a particular and parochial small world.

The cooperative ensemble must articulate (interrelate and compile) the partial and parochial perspectives by transforming and translating information from one level of conceptualization to another and from one object domain to another (Schmidt, 1990).

An interesting issue, raised by Charles Savage in a 'round table discussion' on Computer Integrated Manufacturing (Savage, 1987) illustrates this issue quite well:

> "In the traditional manual manufacturing approach, human translation takes place at each step of the way. As information is passed from one function to the next, it is often changed and adapted. For example, Manufacturing Engineering takes engineering drawings and red-pencils them, knowing they can never be produced as drawn. The experience and collective wisdom of each functional group, usually undocumented, is an invisible yet extremely valuable company resource."

This fact is ignored by the prevailing approach to CIM, however:

> "Part of the problem is that each functional department has its own set of meanings for key terms. It is not uncommon to find companies with four different parts lists and nine bills of

material. Key terms such as *part*, *project*, *subassembly*, *tolerance* are understood differently in different parts of the company."

The problem is not merely terminological. It is the problem of multiple incommensurate perspectives. The issue raised by Savage is rooted in the multiplicity of the domain and the contradictory functional requirements. In Savage's words: "Most business challenges require the insights and experience of a multitude of resources which need to work together in both temporary and permanent teams to get the job done".

In sum, a cooperative work arrangement arises simply because there is no omniscient and omnipotent agent.

Because of the underlying and constitutive interdependence, any cooperative effort involves a number of secondary activities of coordinating and integrating these cooperative relationships. In other words, the cooperating actors have to *articulate* (divide, allocate, coordinate, schedule, mesh, interrelate, etc.) their individual activities (Strauss, 1985; Gerson and Star, 1986; Strauss, 1988). Tasks have to be allocated to different members of the cooperative work arrangement: which actor is to do what, where, when?

By entering into cooperative work relations, the participants must engage in activities that are, in a sense, extraneous to the activities that contribute directly to fashioning the product or service and meeting requirements. That is, compared with individual work, cooperative work implies an overhead cost in terms of labor, resources, time, etc. This point is clearly illustrated by the following observation from the study of air traffic control by Hughes and associates:

"The limit to the existing [Air Traffic Control] system is the human controller and the capacity he/she can cope with safely […]. In other words, it is the workload limit of controllers which determine the capacity of a sector. An apparent solution to capacity problems is to subdivide the airspace into a larger number of smaller sectors. However, this problem is exacerbated by the fact that as the number of sectors increases, so too do the coordination and handover elements of the workload, so that the potential gain is negated." (Hughes et al., 1988, pp. 33 f.).

The obvious justification for incurring this overhead cost and thus the reason for the emergence of cooperative work formations is, of course, that the actors in question could not accomplish the given task if they were to do it individually (Schmidt, 1990).

Conceived of in this way, cooperative work arrangements are transient formations, emerging contingently to handle specific requirements — in response to the requirements of the current situation and the technical and human resources at hand — merely to dissolve again when there is no need for multiple actors and their coordinated effort to handle situations.

That is, cooperative work arrangements arise from and dissolve into individual work. More than that, the boundary between individual and cooperative work is dynamic in the sense that people enter into cooperative work arrangements and leave them according to the requirements of the current situation and the technical and human resources at hand. Cooperative activities are punctuated by individual activities and vice versa. Over time, people shift between individual and coopera-

tive activities and, while engaged in cooperative work activities, they may be simultaneously involved in parallel streams of activity conducted individually.

1.2.2. The evolutionary character of cooperative work

Since cooperative work arrangements require an overhead cost of coordination and articulation work, they should be conceived of as emergent formations.

For example, in a study of the impact of technology on cooperative work among the Orokaiva in New Guinea, Newton (Newton, 1985) observes that technological innovations for hunting and fishing such as shotguns, iron, torches, rubber-propelled spears, and goggles have made individual hunting and fishing more successful compared to cooperative arrangements. As a result, large-scale cooperative hunting and fishing ventures are no longer more economical or more efficient and they are therefore vanishing. Likewise, the traditional cooperative work arrangements in horticulture for purposes such as land clearing and establishment of gardens have been reduced in scope or obliterated by the influence of the steel ax. A similar shift from cooperative to individual work can be observed wherever and whenever new technologies augment the capabilities of individual actors to accomplish the task individually: harvesters, bulldozers, pocket calculators, word processors, etc.

Cooperative work relations emerge in response to the requirements and constraints of the transformation process and the social environment on one hand and the limitations of the technical and human resources available on the other. Accordingly, cooperative work arrangements adapt dynamically to the requirements of the work domain and the characteristics and capabilities of the technical and human resources at hand. Different requirements and constraints and different technical and human resources engender different cooperative work arrangements.

As befits an emergent phenomenon, cooperative work develops historically. For example, agricultural work and craft work of pre-industrial society was only sporadically cooperative. Due to the low level of division of labor at the point of production, the bulk of human labor was exerted individually or within very loosely coupled arrangements. There were, of course, notable exceptions to this picture such as harvest and large building projects (e.g., pyramids, irrigation systems, roads, cathedrals), but these examples should not be mistaken for the overall picture.

Cooperative work as a systematic arrangement of the bulk of work at the point of production emerges in response to the radical division of labor in manufactories that inaugurated the Industrial Revolution. In fact, systematic cooperation in production can be seen as the ‘base line’ of the capitalist mode of production. However, cooperative work based on the division of labor in manufactories is essentially amputated: the interdependencies between the specialized operators in their work are mediated and coordinated by means of a hierarchical systems of social control (foremen, planners etc.) and by the constraints embodied in the layout

and mode of operation of the technical system (conveyer belt etc.). In Marx' words:

> "To the workers themselves, no combination of activities occurs. Rather, the combination is a combination of narrow functions to which every worker or set of workers as a group is subordinated. His function is narrow, abstracted, partial. The totality emerging from this is based on this *utterly partial existence* and isolation in the particular function. It is thus a combination of which he constitutes a part, based on his work not being combined. *The workers are the building blocks of this combination*. The combination is not their relationship and it is not subordinated to them as an association." (Marx, 1861-63, p 253)

The societal precondition for the prevalence of this 'fetishistic' form of cooperative work is that manufacturing and administrative organizations are in control of their environment to the extent that they can curtail its complex and dynamic character. By severely limiting the range of products and services offered and by imposing strict schedules and procedures on their customers and clientele, organizations in branches of mass production and mass-transactions processing were able to contrive synthetic work settings where activities, for all practical purposes, could be assumed to be subsumed under preconceived plans.

In view of the fundamental trends in the political economy of contemporary industrial society, the 'fetishistic' form of cooperative work is probably merely a transient form in the history of work. Comprehensive changes of the societal environment permeate the realm of work with a whole new regime of demands and constraints. The business environment of modern manufacturing, for instance, is becoming rigorously demanding as enterprises are faced with shorter product life cycles, roaring product diversification, minimal inventories and buffer stocks, extremely short lead times, shrinking batch sizes, concurrent processing of multiple different products and orders, etc. (cf. Gunn, (1987)). The turbulent character of modern business environments and the demands of an educated and critical populace, compel industrial enterprises, administrative agencies, health and service organizations, etc. to drastically improve their innovative capability, operational flexibility, and product quality. To meet these demands, work organizations must be able to adapt rapidly and diligently and to coordinate their distributed activities in a comprehensive and integrated way. And this requires horizontal and direct cooperation across functions and professional boundaries within the organization or within a network of organizations.

In short, the full resources of cooperative work must be unleashed: horizontal coordination, local control, mutual adjustment, critique and debate, self-organization. Enter CSCW.

In order to support and facilitate the articulation of distributed and dispersed work activities, modern work organizations need support in the form of advanced information systems. This is illustrated by the efforts in the area of Computer Integrated Manufacturing to integrate formerly separated functions such as design and process planning, marketing and production planning, etc., and by the efforts in the area of Office Information Systems to facilitate and enhance the exchange of information across geographical distance and organizational and professional

boundaries. Common to the efforts in these very different areas are the issues explored by CSCW: How can computer systems assist cooperating ensembles in developing and exercising horizontal coordination, local control, mutual adjustment, critique and debate, and self-organization?

1.2.3. Interdependence in general and interdependence in work

Whatever the specific requirement (or combination of requirements) engendering the emergence of a particular cooperative work arrangement, actors engaged in cooperative work are *mutually dependent in their* work.

The notion of mutual dependence *in work* does not refer to the interdependence that arises from simply having to share the same resource. Actors using the same resource certainly have to coordinate their activities but to each of them the existence of the others is a mere nuisance and the less their own work is affected by others the better. Time-sharing facilities cater for just that by making the presence of other users imperceptible. Being mutually dependent *in work* means that A relies positively on the quality and timeliness of B's work and vice versa. B may be 'down stream' in relation to A but in that case A nonetheless will depend on B for feedback on requirements, possibilities, quality problems, schedules etc. In short, mutual dependence in work should primarily be conceived of as a positive, though by no means necessarily harmonious, interdependence.

This conception of interdependence in work as constitutive of cooperative work is somewhat related to Thompson's concept of "internal interdependence" (Thompson, 1967, pp. 54-55). There a some significant differences, however, that need to be explored. In his classic study Thompson makes a distinction between three "types of interdependence":

Pooled interdependence:

> "The Tuscaloosa branch of an organization may not interact at all with the Oshkosh branch, and neither may have contact with the Kokomo branch. Yet they may be interdependent in the sense that unless each performs adequately, the total organization is jeopardized; failure of any one can threaten the whole and thus the other parts. We can describe this situation as one in which each part renders a discrete contribution to the whole and each is supported by the whole. We will call this *pooled interdependence*." (Thompson, 1967, p. 54)

Sequential interdependence:

> "Interdependence may also take a serial form, with the Keokuk plant producing parts which becomes inputs for the Tucumcari assembly operation. Here both make contributions to and are sustained by the whole organization, and so there is a pooled aspect to their interdependence. But, in addition, direct interdependence can be pinpointed between them, and the order of that interdependence can be specified. Keokuk must act properly before Tucumcari can act; and unless Tucumcari acts, Keokuk cannot solve its output problems. We will refer to this as *sequential interdependence*, and note that it is not symmetrical." (Thompson, 1967, p. 54)

Reciprocal interdependence:

> "A third form of interdependence can be labeled reciprocal, referring to the situation in which the outputs of each become inputs for the others. This is illustrated bye the airline which con-

tains both operations and maintenance units. The production of the maintenance unit is an input for operations, in the form of a serviceable aircraft; and the product (or by-product) of operations is an input for maintenance, in the form of an aircraft needing maintenance. Under conditions of reciprocal interdependence, each unit involved is penetrated buy the other. There is of course, a pooled aspect to this, and there is also a serial aspect since the aircraft in question is used by, then by the other, and again by the first. But the distinguishing aspect is the reciprocity of the interdependence, with each unit posing contingency for the other." (Thompson, 1967, pp. 54-55)

Summing up, Thompson observes:

"In the order introduced, the three types of interdependence are increasingly difficult to coordinate because they contain increasing degrees of contingency. With pooled interdependence, action in each position can proceed without regard to action in the other positions so long as the overall organization remains viable. With sequential interdependence, however, each position in the set must be readjusted if any one of them acts improperly or fails to meet expectations. There is always an element of potential contingency with sequential interdependence. With reciprocal interdependence, contingency is not merely potential, for the actions of each position in the set must be adjusted to the actions of one or more others in the set." (Thompson, 1967, p. 55)

In the context of understanding cooperative work, Thompson's notion of interdependence is problematic. The reason being that the issue pursued by Thompson is that of 'the theory of the business firm', not that of actual cooperative work arrangements.

Thus, the concept of pooled interdependence refers to the interdependence of units owned by the same firm, conglomerate, corporation, holding company, municipality, state, or whatever. The units do not interact in doing their work — they contribute financially to 'the whole', the firm etc.. Thus, the fate of any one unit is certainly dependent on the *financial* performance of the other units and, consequently, the *financial* performance of the collection of units as a whole and they are thus quite interdependent, but only financially.

The sequential interdependence, on the other hand, is obviously an interdependence constituted by the productive activities of the units: The output of A's activities becomes the input for B's activities. The same applies to pooled interdependence defined as "the situation in which the outputs of each become inputs for the others".

The distinction between sequential and reciprocal interdependence is not quite clear, however. In so far as the terms 'input' and 'output' refer to the flow of materials, components, products, and services, there is a clear difference between sequential and reciprocal interdependence. For example, if the outcome of A's activities provide the input (material, components etc.) for B's activities, B obviously depends on A in terms of quality, quantity, schedules etc. But the interdependence is not strictly one-directional. A also depends on B, albeit in other ways — not only to "solve its output problem" but also to provide feedback on quality problems and the like. A and B depend on each other to do their respective work but they depend on each other in different ways. In reciprocal interdependence, the outputs of each become inputs for the others. They are reciprocally interde-

pendent in the sense that each of the units in its work depends of the performance of the others in terms of quality, quantity, schedules etc. as well as feedback.

It is worth noting that Thompson's example of reciprocal interdependence in an airline is quite misleading. While Maintenance certainly provides an output for Operations in the form of a serviceable aircraft and Operations thus, in its work, clearly depends on Maintenance, Operations does not *produce* "aircraft needing maintenance". In so far as the difference between sequential and reciprocal interdependence is defined in terms of the direction of input-output relations, the interdependence between Maintenance and Operations of an airline is just another example of sequential interdependence. In discussing the difference between sequential and reciprocal interdependence, however, Thompson introduces a new definition of reciprocal interdependence: "the distinguishing aspect is the reciprocity of the interdependence, with each unit posing contingency for the other." And, in fact, he eventually seems to define the different types of interdependence and interdependence as such by means of the concept of contingency. Whether that is a reasonable way of conceptualizing the interdependence of the parts of the organization in the context of the theory of the business firm and its structure is beyond the scope of this report, but it is far too general and inclusive to conceptualize actual cooperative relations of work. The various units of an organization (or a market for that matter) poses contingency for each other in innumerable ways: not only by means of productive activities whose outcome others take as input for their productive activities, but also by demanding a product or service, by being owned by the same firm and thus partaking in the same financial and administrative arrangements and sharing the same corporate image in public, by being involved in the internal politics of the firm, by competing with others, and so forth. What we need to study, however, is the cooperative work relations that emerge between people that mutually depend on each other's productive activities in order to do their work.

In order to understand these interdependencies and how they determine cooperative practices, we need to introduce the concept of the 'field of work', that is, the part of the world that is being transformed or otherwise controlled by the cooperative work arrangement.

1.2.4. The common field of work

Having entered into a cooperative work arrangement, the actors cooperatively and interactively transform and control a conglomerate of *mutually interacting* objects and processes, the field of work. That is, their interdependence in their work is constituted by the interdependencies between the objects and processes constituting the field of work. Thus, all cooperative work is based upon interactions mediated through the changing state of a common field of work.

The field of work is not a thing — it is a conceptual construct that shall help us in analyzing and conceptualizing the formation and articulation of cooperative work arrangements:

First, the field of work and the cooperative work arrangement mutually constitute and delimit each other. The field of work is always the field of work for a particular cooperative work arrangement and the cooperative work arrangement is itself bounded and constituted by the interdependence of its activities as determined by the field of work. Second, the field of work itself is manifold. It comprises, of course, the objects and processes but also the sensors and effectors as well as the more complex tools and control mechanisms that has been inserted between the actors and the objects and processes. In addition, the field of work may comprise the repertoire of material resources and technical artifacts (data bases, inventories and other repositories, buildings, infrastructures) by means of which the production process is performed. Third, the production process will be carried out in a wider work environment with specific constraining characteristics and operational demands (commercial, economic, environmental, legal requirements and constraints). Fourth, in the contemporary system of social division of labor, the actor-object relationship is a recursive phenomenon in the sense that some actor-object relationships are objects for other work processes (e.g., training, ergonomic intervention) and in the sense that some cooperative work arrangements are objects of work of other cooperative work arrangements (e.g., administration, ethnographic studies, and systems engineering). In the social division of labor, the field of work for some ensembles may even be collections of data: the archives of administrative agencies, for example, may be the common field of work for the staff of the registry; the global collection of data on causes of deaths is the common field of work of doctors, authorities, and other parties maintaining and using the International Classification of Diseases.[1] And fifth, the boundary and character of the field of work changes dynamically. For example, when a ship meets another ship during its voyage, the field of work of the crew — basically, the ship and the water — "suddenly expands to include another ship" (Perrow, 1984, p. 178). Similarly, crews face fields of work that are basically different from the one they are faced with on the open ocean

> "when ships a city block long go into port with only two feet under their keel, with highly unpredictable suction effects and a virtual complete loss of maneuverability. [This] increases the time-dependent nature of the system and reduces the slack available (tighter coupling), and through increased proximity, brings into play poorly understood processes (the suction and bank effects), which rely upon indirect and inferential information sources (thus, more complex interactions are fostered)." (Perrow, 1984, p. 182).

The concept of a field of work has been analyzed — under different labels — by a number of researchers in empirical and theoretical work analysis.

First and foremost, Charles Perrow (1984) uses the term 'system' rather consistently, in his seminal comparative study of high-risk and high-tech work settings, in much the same sense as the term 'field of work' as defined above. The only significant difference is that Perrow — whose focus is on the etiology of system accidents, not on the formation and articulation of cooperative work ar-

1 See the discussion of the ICD below.

rangements — includes aspects of the work organization in the definition of 'the system'.

In order to be able to compare (the systemic risk-potentials of) different 'systems', Perrow suggests a two-dimensional framework: On one hand "complex and linear interactions" and on the other hand "tight and loose coupling", cf. the tables in Figure 1-1 and Figure 1-2:

Complex versus Linear Systems

Complex Systems	*Linear Systems*
Tight spacing of equipment	Equipment spread out
Proximate production steps	Segregated production steps
Many common-mode connections of components not in production sequence	Common-mode connections limited to power supply and environment
Limited isolation of failed components	Easy isolation of failed components
Personnel specialization limits awareness of interdependencies	Less personnel specialization
Limited substitution of supplies and materials	Extensive substitution of supplies and materials
Unfamiliar or unintended feedback loops	Few unfamiliar or unintended feedback loops
Many control parameters with potential interactions	Control parameters few, direct, and segregated
Indirect or inferential information sources	Direct, on-line information sources
Limited understanding of some processes (associated with transformation processes)	Extensive understanding of all processes (typically fabrication or assembly processes)

Figure 1-1. Aspects of complex versus linear systems (Perrow, 1984, p. 88).

Tight and Loose Coupling

Tight Coupling	*Loose Coupling*
Delays in processing not possible	Processing delays possible
Invariant sequences	Order of sequences can be changed
Only one method to achieve goal	Alternative methods available
Little slack possible in supplies, equipment, personnel	Slack in resources possible
Buffers and redundancies are designed-in, deliberate	Buffers and redundancies fortuitously available
Substitutions of supplies, equipment, personnel limited and designed-in	Substitutions fortuitously available

Figure 1-2. Aspects of tight and loose coupling (Perrow, 1984, p. 96).

Similarly, in the Cognitive Engineering approach to the design of decision support systems, Woods (1988) distinguishes different complexity factors for

problem solving with respect to three basic elements (the Agent, the Representation, and the World). In his analysis the dimensions pertaining to the complexity posed by "the World", Woods divides the two dimensions suggested by Perrow into four, namely Dynamism, Many Highly Interacting Parts, Uncertainty, and Risk. The emphasis on Risk as a separate dimension again reflects the class of domains Cognitive Engineering research primarily addresses. For our purposes, Risk can be seen as one among many demands and constraints posed by the work environment in general (along with, say, flexibility, resource economy).[1] In the context of cooperative work, the effect of Risk is to make the field of work tighter coupled (because actions are irreversible under constraints of Risk). Uncertainty, on the other hand, is of significant importance to the formation and articulation of cooperative work arrangements.

The field of work of a particular cooperative work arrangement can thus be characterized along the following dimensions.

Structural complexity: The members of a cooperative work arrangement may interact through and in relation to a field of work characterized by different degrees of interactional complexity:

> "When a world is made up of a large number of highly interconnected parts, one failure can have multiple consequences (produce multiple disturbances); a disturbance could be due to multiple potential causes and can have multiple potential fixes; there can be multiple relevant goals which can compete with each other; there can be multiple on-going tasks having different time spans. In addition, the parts of the world can be complex objects in their own right." (Woods, 1988, p. 130)

When the field of work encompasses subsystems that are complex objects in their own right, cooperative work will involve multiple representations and conceptualizations of the domain. Thus, the decision-making process requires employment of, and transformations between, different representations and conceptualizations (Mintzberg, 1979, p. 268; Rasmussen, 1988, pp. 176 f.; Star, 1989).

Temporal complexity: The members of a cooperative work arrangement may interact through and in relation to a field of work characterized by more or less dynamic behavior or by being more or less tightly coupled and hence time-critical.

> "When a world is dynamic, problem-solving incidents unfold in time and are event-driven, that is, events can happen at indeterminate times. This element means that there can be time pressure, tasks can overlap, sustained performance is required, the nature of the problem to be solved can change, and monitoring requirements can be continuous or semi-continuous and change over time."(Woods, 1988, p. 130)

Apperceptive complexity: The members of a cooperative work arrangement may face a vast variety of problems in apperceiving (perceiving, making sense of, interpreting) the state of affairs in the field of work due to, for example, noise, un-

---

1 The work environment in the wider sense, that is, the demands and constraints that are not directly posed by the field of work but rather characterize the conditions under which the demands and constraints directly pertaining to the control of the field of work must be met. We will simply refer to this 'second order' environment, the work environment in the wider sense, as *the work environment*.

reliable sensors, indirect or inferential evidence, or from ambiguous, misleading etc. information.

> "When there is high uncertainty, available data can be ambiguous, incomplete, erroneous, low signal to noise ration, or imprecise with respect to the state of the world; the inferential value of data can vary with context; future states and events are not completely predictable. Uncertainty can be due to external occurrences, noise, changes in noise parameters over time, nonlinearities, time delays or the influence of previous events and inaccurate measurements can arise through sensor failures, miscalibrations or misentries." (Woods, 1988, p. 130)

In order to be able to conceptualize and specify the support requirements of cooperative work we need to make a fundamental analytical distinction between (a) cooperative work activities in relation to the state of the field of work and mediated by changes to the state of the field of work and (b) activities that arise from the fact that the work requires and involves multiple agents whose individual activities need to be coordinated, scheduled, meshed, integrated etc. — in short: *articulated*. This distinction is fundamental to CSCW. As noted in the introduction, the core issues in the design of conventional computer-based systems for work settings have been to develop effective computational models of pertinent structures and processes in the field of work (data flows, conceptual schemes, knowledge representations) and adequate modes of presenting and accessing these structures and processes (user interface, functionality).

The booking system of an airline, for example, is a computer-based system for the cooperative task of handling reservations. The database of the booking system embodies a model of the seating arrangements of the different flights. The seating arrangements of the different flights and the database model of it constitutes the field of work of the booking agents. Thus, the operators of the booking agents cooperate by changing the state of the field of work, in casu, by blocking (or releasing) seats. The system does not in any way support articulation work, apart from providing a simple access control facility. In this case, the field of work can be handled as a system of discrete and extremely simple (binary) state changes. Apart from the fact that a seat can only be assigned to one person at a time, there are no interactions between processes. Accordingly, even though a booking system does not support articulation work, it is probably quite sufficient for the job.

Thus, in many cases the articulation of the different activities of different actors in relation to the field of work (or in relation to the model of the field of work as incorporated in the computer-system) is not a problem for the designer. In order to develop computer-based systems that support the articulation of cooperative work, however, this issue comes to the fore. The distinction between doing the work and articulating the work is therefore essential to CSCW.

1.2.5. The articulation of cooperative work

The concept of articulation work was developed by Strauss, Gerson, Star and others (Strauss, 1985; Strauss et al., 1985; Gerson and Star, 1986; Strauss, 1988) in order to handle the fact that cooperating actors, being mutually dependent in

their work, have to *articulate* (divide, allocate, coordinate, schedule, mesh, interrelate, etc.) their individual activities: Who is doing what, where, when, how, by means of which, under which constraints?

In the words of Strauss (1985, p. 8), articulation work is "a kind of supra-type of work in any division of labor, done by the various actors":

> "Articulation work amounts to the following: First, the meshing of the often numerous tasks, clusters of tasks, and segments of the total arc. Second, the meshing of efforts of various unit-workers (individuals, departments, etc.). Third, the meshing of actors with their various types of work and implicated tasks."

Articulation of cooperative work is thus required with respect to multiple "salient dimensions" (Strauss, 1985): who, what, where, when, how, etc.? A tentative inventory of the salient dimensions of articulation work could look as follows:

(1) Articulation in terms of **actors**, that is, the actual or potential participants in the cooperative effort whose cooperative activities are being articulated (in different capacities such as roles, jobs, individuals, collectives): Which partners are potentially relevant for a particular project in terms of skills, competing commitments etc.? Who are available when?

(2) Articulation in terms of **responsibilities**, that is, in terms of general accountability (obligation, commitment).

(2) Articulation in terms of **tasks**, that is, in terms of an operational intention (goals to attain, obligations and commitments to meet): What is the problem? What is to be done? Who should do it? Should I do it? Which task is (normally, advisably, or according to statute) to be undertaken in which circumstances, by which actor, based on what information and which criteria, creating what information? What is the (normal, advisable, or statutory) relation between tasks (procedure, workflow)?

(3) Articulation in terms of **activities**, that is, in terms of an unfolding course of purposive action. What are the others doing, and why? What have they done, what will they be doing, etc.? Do they cope?[1]

(4) Articulation in terms of **conceptual structures**, that is, in terms of the relationship between categories used within a specific community as ordering devices: definition, classification, prototypical, causal, genetic, historical, means/-end relations.

(5) Articulation in terms of common **resources**:

---

1 The terminology used here comes from the Scandinavian tradition: An *activity* is used to denote a work process as an unfolding course of action, but only those aspects of a work process that are relevant to doing the work with the currently available resources, not all other incidents that may occur in the same course of action but which are of no consequence for getting the work done (like spilling coffee). — The concept of a task, on the other hand, is used to denote an operational intent, irrespective of how it is implemented. A task is expressed in terms of *what*, an activity in terms of *how*. A task can be *accomplished*, an activity can *cease.* (Andersen et al., 1990)

- **information** resources (documents, letters, applications, notes, files, memos, reports, drawings): Which actor can access, change, delete, copy which information resources? To which actor is the object to be displayed, in which format? Which actor can see who doing what to which objects?
- **material** resources (materials, components, assemblies). Which materials, components, assemblies are available where, when, how, in which quantity? What are their characteristics?
- **technical** resources (tools, fixtures, machinery, software applications). What are their operational characteristics (machining tolerance, suitability for different kinds of materials and material dimensions, processing time and cost)?
- **infrastructural** resources (rooms, buildings, communication facilities, transportation facilities). What are their operational characteristics (capacity, location, compatibility, turnaround time, bandwith)?

Furthermore, articulation work is never done in the abstract, it is always done in relation to and in terms of a wider context:

- the state of **field of work**;
- the demands and constraints as posed by the **work environment**;
- the wider **organizational setting**.

And, finally, articulation work is, of course, done with reference to abstract systems of reference: **time** and **space**.

It goes without saying that these dimensions of articulation work are interdependent. For example, articulation with respect to tasks may refer to actor, various resources, the field of work etc. and the tasks, actors, resources etc. may themselves be defined, classified, etc.

Compared to such terms as ‘coordination’ or ‘conversation for action’, the concept of articulation work provides a number of benefits in a CSCW context: First, the concept of articulation work is more flexible than the connotations usually implied by the term ‘coordination’. Articulation work connotes far more than mere scheduling and allocation of resources. In connotes, for instance, monitoring, handing over, resolving inconsistencies, reconciling incommensurate assumptions, opinions, and beliefs, and so forth. Second, articulation work specifically refer to the articulation of distributed interdependent activities required when multiple actors are engaged in cooperative work. That is, is does not necessarily encompass the coordination of “multiple, interdependent activities” performed by only one actor.[1] Third, articulation work is conceived of

[1] Malone and Crowston define “coordination” as “the act of managing interdependencies between activities performed to achieve a goal” and add: “we have become convinced that the essential elements of coordination listed above arise whenever multiple, interdependent activities are performed to achieve goals — even if only one actor performs all of them” (Malone and Crowston, 1990, p. 361 f.). — A major problem with that approach is that it does not seem to provide a criterion for

with respect to the specific context, that is, in terms of the state of affairs in the field of work. And fourth, articulation work is conceived of as on-going articulation of cooperative work in face of unforeseen contingencies.

Thus, for example, the major problem with the 'conversation for action' metaphor (Winograd and Flores, 1986) is what is leaves out. The 'speech act' conception of articulation work ignores the articulation of meanings (concepts, categories, assumptions, beliefs). Moreover, in the 'conversation for action' approach articulation work is conceived of as an abstract, domain-independent generic activity — that is, the fact that work is articulated with reference to and in terms of the state of the field of work is not accounted for. In other words, the 'conversation for action' metaphor provides a strangely disembodied account of articulation work that it is hard to recognize in real-world settings, except, perhaps, in management committees and the like.

1.2.6. The complexity of articulation work

The very fact that multiple actors are involved in doing the work introduces an essential element of distributed decision making.

The contingencies encountered in any human action may defeat the very best plans and designs. As pointed out by Suchman (1987, p. 38), "the relation of the intent to accomplish some goal to the actual course of situated action is enormously contingent." Plans may of course be conceived by actors prior to action but they are not simply executed in the actions. Action is infinitely rich compared to the plan and cannot be exhausted by it.

Accordingly, even in the most mundane cooperative work arrangements, each individual encounters contingencies that may not have been anticipated by his or her colleagues and that, possibly, will remain unknown to them. Each participant in the cooperative effort is faced with a, *to some extent*, unique local situation that is, in principle, partially opaque to the others and each participant has to deal with this local situation individually. For example: misplaced documents, shortage of materials, delays, faulty parts, erroneous data, variations in component properties, design ambiguities and inconsistencies, design changes, changes in orders, cancellation of orders, rush orders, defective tools, software incompatibility and bugs, machinery breakdown, changes in personnel, illness, etc. That is, due to the fundamentally 'situated' nature of human action, cooperative work arrangements take on an inexorably distributed character.

Furthermore, the fact that the cooperative arrangement involves — and has emerged to facilitate — a combination of different specialties, incongruent heuristics, and incommensurate perspectives introduces a systematic element of distributed decision making in cooperative work.

And finally, work is an individual phenomenon in so far as labor power happens to be tied to individuals and cannot be separated from the individuals. That

determining the level at which coordination is considered; in principle, all processes of the field of work at all semantic levels are interdependent and have to be coordinated.

is, a cooperative work process, is performed by individuals with individual interests and motives. Because of that, cooperative ensembles are coalitions of partially incongruent and even conflicting interests rather than perfectly collaborative systems. Thus, in the words of Ciborra (1985), the use of information for "misrepresentation purposes" is a daily occurrence in organizational settings. The Russian proverb saying that 'Man was given the ability of speech so that he could conceal his thoughts' applies perfectly to the use of information in organizations.

In sum, then, cooperative work is, in principle, distributed in the sense that actors are semi-autonomous in their work in terms of contingencies, criteria, methods, specialties, perspectives, heuristics, interests, motives, and so forth.

Now, due to the very interdependence in work that gave rise to the cooperative work arrangement in the first place, the distributed nature of the arrangement must be kept in check, managed. As an integral part of cooperative work, articulation work arises as a set of activities required to manage the distributed nature of cooperative work.

In order to account for the distributed nature of cooperative work, Gerson and Star (1986, p. 266) develops the concept of articulation work further by emphasizing its contingent and dynamic nature:

> "Reconciling incommensurate assumptions and procedures in the absence of enforceable standards is the essence of articulation. Articulation consists of all the tasks involved in assembling, scheduling, monitoring, and coordinating all of the steps necessary to complete a production task. This means carrying through a course of action despite local contingencies, unanticipated glitches, incommensurate opinions and beliefs, or inadequate knowledge of local circumstances.
>
> Every real world system is an open system: It is impossible, both in practice and in theory, to anticipate and provide for every contingency which might arise in carrying out a series of tasks. No formal description of a system (or plan for its work) can thus be complete. Moreover, there is no way of guaranteeing that some contingency arising in the world will not be inconsistent with a formal description or plan for the system. […] Every real world system thus requires articulation to deal with the unanticipated contingencies that arise. Articulation resolves these inconsistencies by packaging a compromise that 'gets the job done,' that is closes the system locally and temporarily so that work can go on."

Now, although it is crucial for CSCW systems design to maintain that cooperative work is distributed *in principle*, this general statement is insufficient for analyzing cooperative work for CSCW systems design. We need to understand the specific sources engendering the specific aspects of distributed decision making under specific conditions — only then is it possible to improve the ability of a cooperative ensemble to manage and curb the distributed nature of its cooperative effort.

For example, when Suchman (1987) posits that "the relation of the intent to accomplish some goal to the actual course of situated action is enormously contingent", this statement is overly general and exaggerated. Action to accomplish some goal is not *always* "enormously contingent"! Of course, any action may be construed as enormously contingent in Herakleitos' sense that every situation is

unique. But the action is not necessarily “enormously contingent” to the actors themselves. Contingencies may be more or less complex to deal with, more or less serious in terms of effect, scope etc., more or less frequent, and so forth, and different contingencies may affect the outcome of action and the validity of plans differently.

In fact, the distributed character of cooperative work varies depending on a number of factors, e.g., the specific structural and temporal complexities of their interdependence as determined by the field of work (complex interactions, massive concurrency, intermittent interactions), the distribution of activities in time and space and the concomitant constraints on communications, the apperceptive complexity posed by the field of work and hence the discretionary nature of contributions of participants, the number of participants in the cooperative ensemble, the degree and scope of specialization required by the manifold nature of the field of work and hence the structural complexity of the cooperative work arrangement itself, the apperceptive complexity of assessing the state of affairs within the cooperative ensemble, and so on.

The more distributed the activities of the cooperative work arrangement, the more complex the articulation of the activities of that arrangement.

1.3. Articulation of cooperative work: Cases

A cooperative work relationship is constituted by the fact that multiple workers are interdependent in their work. In other words, they are working on the same ‘field of work’, that is, they are transforming and controlling a conglomerate of *mutually interacting* objects and processes. Thus, all cooperative work involves and, indeed, is based upon interaction through changing the state of a common field of work. What one worker — A — is doing is of import to B and C and they can to some extent infer what A is doing from the changing state of the field of work.

While articulation of cooperative work via the field of work is basic to all cooperative work, it is rarely adequate. In fact, articulation of cooperative work involves and, indeed, requires a vast variety of *modes and mechanisms of interaction* that are combined and meshed dynamically in accord with the specific requirements of the unfolding work situation and the means of communication available.

In the following sections we will present a set of cases that represent different pertinent modes and mechanisms of articulation of cooperative work.

1.3.1. Manual control: The hot rolling mill

In modern industrial plants, cooperative work interaction is typically mediated by complex machine systems and sometimes the setting even prevents direct communication between agents. Is such settings cooperative work can, so to

speak, be observed in its basic form, in the nude: cooperative work solely mediated by the changes to the common field of work. A case of such settings is therefore the best place to start our analysis.

The classic study by Popitz and others (1957) on cooperative work in the German steel industry provides an eloquent example of "structurally mediated cooperation" or cooperative work mediated by a technical system, in this case cooperative manual control of a rolling mill that shapes hot steel ingots into strips of different forms and dimensions.[1]

The basic transformation process of rolling metal is to reduce or change the cross-sectional area of a work piece — typically a hot ingot — by the compressive forces exerted by rotating rolls. The mill in question has a reversible double-roll of six different calibers (se Figure 1-3). The work piece — a 'strip' or 'slab' (Block) — is formed as it passes through the rolls and is shifted from caliber to caliber. Depending on the alloy of the particular strip and its temperature, the strip may require multiple passes at each caliber. The proper reduction per pass depends on the type of material and other factors. For example, Thomas steel requires fewer passes than the harder Siemens-Martin steel and hot strips require fewer passes than colder ones. After each pass the distance between the rolls is adjusted and after some passes the strip is tilted 90°.

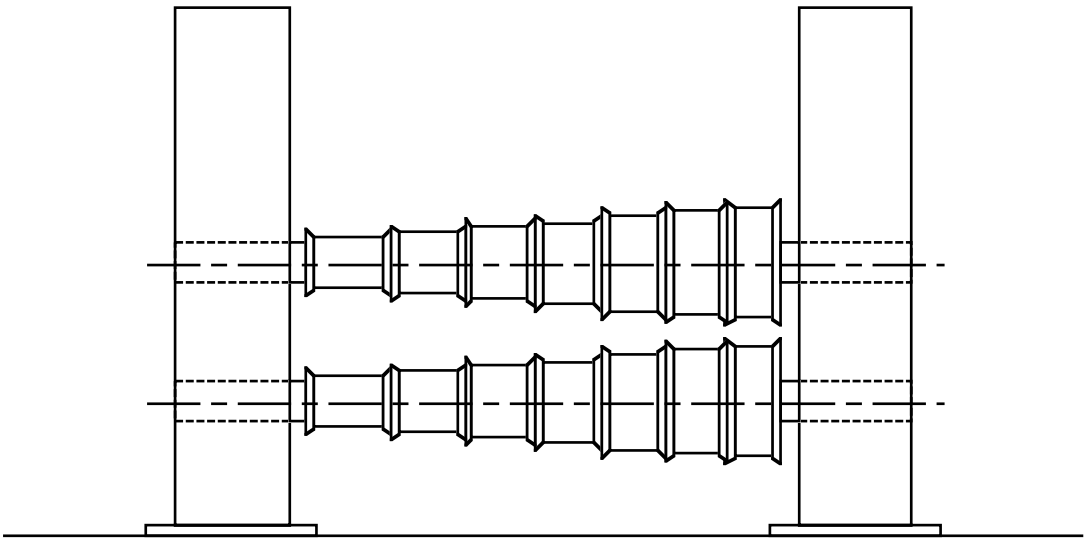


Figure 1-3. Cross section of the rolling mill stand showing the two six-caliber rolls. When being shaped, the strip is subjected to multiple passes at each caliber whereupon it is shifted to the next caliber, from left to right. The distance between the rolls is adjusted between each pass by the driver. (Popitz et al., 1957)]

1 One should bear in mind that the study was conducted in the early 1950's. Modern rolling mills are quite different from the one described here. For example, the mill in the Popitz study is driven by steam! The study is used here, nonetheless, not only because it is a brilliant sociological study, but because it is an exceptionally clear and telling case of cooperative work mediated by the field of work.

The rolling mill is operated by four workers: a ‘driver’ (Steuermann), a ‘tip cart operator’ (Kippwagenführer), an ‘engineer’ (Walzenzugmaschinist), and a ‘foreman’ (Walzenmeister) or his deputy.

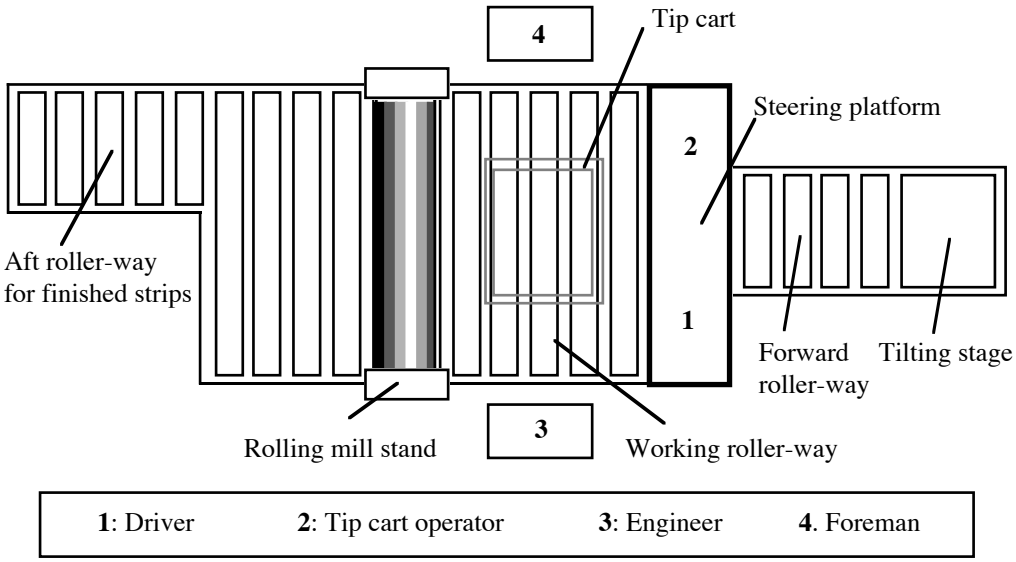


Figure 1-4. The rolling mill seen from above. (Popitz et al., 1957)

The driver and the tip cart operator are stationed on a bridge-like platform in a cabin from where they can see the process in front of and in the rolls but not the processes in the rolling-way area after the stand.

The driver controls the tilting stage and the forward rolling-way that moves the incoming ingot forward towards the rolls. His main task, however, is to control the rolling-way in front of the stand that makes the strip move to or from the rolls and to adjust the distance between the rolls for each pass at the same caliber.

The tip cart operator, on the other hand, tilts the strip between passes at the same caliber and, when a change of caliber is required, shifts the strip to another caliber. In addition, the tip cart operator controls the rolling-way in the area after the rolls, and moves the strip back to the rolls when a new pass is required. If the strip is not finished, he should not let it move too far to the rear; it has to be caught and moved back into the rolls at the same caliber.

When the strip has to be tilted for another pass at the same caliber, it must be done while the it is in motion and while the driver adjusts the setting of the rolls. These operations must be carried out quickly in a fluently integrated way.

The engineer controls the speed and direction of rotation of the rolls. He is stationed next to the rolling mill, before the roll stand, but he is also unable to see the area behind the roll stand. Apart from occasional signs from the foreman, he has to rely on indirect indicators such as changing light intensity, glow from the hot strips, and vibrations. Thus, when the strip is moving forward, the engineer cannot see it coming out behind the rolls but the vibrations of the machine are conveyed by his control levers and he senses immediately when the friction be-

tween strip and rolls has ceased, the machine is running more smoothly, and the speed of rotation increases. He then stops and reverses the rotation of the rolls. The tip cart operator lets the strip move to the rolls again and as soon as it touches the rolls, the engineer starts the rolls again. The tip cart operator assists in making the rolls grip by letting the rolling-way run backwards (pp. 58 f.).

The foreman supervises the work of the three others. He provides the specifications for the strips to the driver and the tip cart operator. The foreman's deputy monitors the rolling process and, when required, intervenes by giving instructions with a signal whistle.

As pointed out by Popitz and associates, the field of work in this case — the complex technical system and the transformation processes — is subjected to a "rigorous temporal order" (p. 60).

First, each operation requires a certain time whose minimum is determined and cannot be exceeded. Popitz and associates aptly refer to this as the "Eigenzeit" (intrinsic timing) of the technical system.

Second, for the strip to be rolled requires a certain temperature (for alloy steels, 930°-1260°C). An inadequate temperature will deteriorate the quality of the finished product and it may have to be discarded. In the worst case, the rolls may break which amounts to an enormous loss (p. 63). Since the temperature of the strip falls continuously, the whole operation is time critical. Any delay will reduce the temperature because of which the strip may need additional passes at each caliber which prolongs the process even further and so forth.

Third, the continuity of the various stages of the rolling process is a prerequisite for its success. For instance, the rolls will only receive the strip if it is in motion but the moving strip will only be received if the rolls starts moving in the very moment when the strip arrives. Thus, in the words of Popitz and his colleagues, "the work activities are subjected to a temporal order determined by the 'Eigenzeit' of the technical system, the prescribed succession of operations and the necessity of maintaining the continuity of the process" (p. 61).

In fact, the functional decomposition and allocation of work at the rolling mill is designed to ensure continuity. For example, when the strip has passed the rolls, the tip cart operator has to catch it with the aft rolling-way and move it back to the rolls. The engineer then has to reverse the rolls in order to receive the strip. If it rolls out in front of the rolls, the driver has to catch it and move it to the tip cart so that it can be tilted or shifted to another caliber. "The work activities of one man depends on the activities of the others: Each does what he has to do so that another can do what he has to do. Consequently, the activities are mutually temporally determined" (p. 61).

The rigorous determinacy of the processes does not prevent variations and disturbances. A typical variation may arise when a particular ingot does not have the required temperature. The driver can see that from the color of the glowing ingot — it is too dark red. The driver may reject it and it is then picked up by the crane again and taken back to the furnace. However, if the driver deems it just hot

enough and accepts it, he has to adjust the distance between the rolls to decrease the reduction per pass and allow for more passes. The tip cart operator observes the roll adjustment and apperceives immediately that the driver is planning for additional passes. From that he can infer that it will require more passes until the strip has to be tilted and that it will take longer until it can be shifted to the next caliber. The engineer also apperceives the situation immediately and operates the rolls more carefully than usual (pp. 62 f.).

The cooperating workers are — for all practical purposes — unable to coordinate their individual activities by talking to each other. The noise level prevents them from talking during work, and some of them cannot even see each other. It thus often happens that operators do not talk to each other during an eight-hour day. Furthermore, the operators are so intensely preoccupied with controlling a process that has its own intrinsic temporal order, that they do not have the time to talk or to watch the hand movements of each other (p. 185).

Thus, cooperative work mediated by the changing state of the field of work does not require participants to socialize beyond the immediate interaction.[1] It is, for instance, completely irrelevant to the tip cart operator whether the engineer is a good comrade or not. But his capability to drive a rolling mill is of utmost importance to him.

An operator only operates the system rationally and effectively if each operation is carried out with a view to the necessary cooperation with the others (p. 185). That is, he has to take into account the preceding, concurrent, and immediately ensuing operations.

Each operator is on his own in doing his work — but in such a way that his activity at any time fits closely into and continues the technical transformation process. Thus, every variation in the work of another of import for this technical process must immediately be countered by him by a variation in his own work. For example, if the driver notices that a relatively cold strip needs more passes he will adjust the setting of the rolls differently than normal. That signifies that the tip cart operator has to tip that strip two or four passes later. Similarly, if the strip bends — that can happen at the two last calibers — then the tip cart operator, the driver and the engineer have to modify their operations simultaneously.

The crew nevertheless manages to act in a concerted may without verbal communication and without watching the operations of each other. Each of them know what the other is doing by apperceiving the behavior of the mill: the movement of the rolling-way and the tip cart as well as the setting and direction of rotation of the rolls. In a normal rolling process this mediation is even reduced to one object: the glowing strip. Its motions indicates what the others are doing at any time. “The activities of each worker are thus apperceived on the basis of the behavior of the object that these activities assist in moving and transforming” (p.

1 It goes without saying that technically mediated cooperative work does not prevent a supplementary socialization. Close personal contacts may occur, for example fostered by guild traditions (Popitz et al., 1957, p. 185).

187). The activities of any of them are being observed continuously and intensely by the others — *by way of* the behavior of the strip and the mill.

In sum, then, the cooperative operation of the hot rolling mill is a telling example of cooperative work mediated by state changes in the common field of work:

(1) The field of work is strictly causally coupled; all sub-processes are subjected to a rigorous temporal order and have to be carried out in a continuous way.

(2) The operators are engaged in 'manual process control' (like a driver of a car); they are directly in control of processes and thus have to perform control actions continuously. Because they are in the first order control loop, their distributed activities are subjected to the same rigorous temporal order as the processes themselves and have to be articulated in the same continuous way.

(3) The operators cannot leave their stations during work. None of them can oversee the process as a whole from their respective stations. To a large extent, they have to rely on indirect evidence (vibrations, light intensity, glow). The operators are thus highly interdependent.

(4) The cooperating workers are — for all practical purposes — unable to coordinate their individual activities by talking to or watching each other.

(5) The crew nevertheless manages to act in a concerted may without verbal communication. They succeed in doing so because what they have to coordinate between them is — first and foremost — the exact timing of their individual operations.

(6) The awareness of the intentions and actions of the other members of the crew — that is, the dynamic and mutual awareness that is a prerequisite for the articulation of their cooperative effort — is developed and maintained on the basis of intense observation of the behavior of the strip, the rolls, the roller-way, the tip cart etc.

(7) However, the changing state of the field of work is a perilous channel of articulation work because state changes in the field of work are rudimentary as a means of communication: The bandwith is quite restricted; the turn-around time of the interaction is determined by the frequency of state changes in the field of work, and — most importantly — the message is completely embodied in the state of the field of work. A state change is never undertaken to send a message. In fact, the behavior of the strip and the mill does not carry any message. The strip is just a strip, and whatever is done to it is done with the single purpose of transforming it the proper way. The operators simply take their cues from the state of the field of work and infer the actions and intentions of their colleagues from that.

1.3.2. Supervisory process control: The nuclear power plant

In the control of highly complex and automated processes, the cognitive control functions require more sophisticated articulation work than interaction via the field of work allows.

A recent study by Kasbi and Montmollin (1991) explores the impact on cooperative work practice of the planned radical computerization of control room design for the French 1500 MW power plants of the N4 PWR series. In order to study the impact of this putative "technological leap", it was decided to connect a prototype of the advanced computer-based control room to a 1300 MW PWR process simulator called S3C. Operators running the S3C were observed and their performance was compared with field study findings from conventional control rooms.

In the power plants in question, two operators manage the control function. In some control situations (start-up or shut-down of plant units, incidents, etc.), there are prescribed, detailed procedures which organize the allocation of tasks between the two operators. The procedures are based on a subdivision of the process into a Primary side (nuclear reactor) and a Secondary side (water and steam). However, in most control situations, the operators are left free to decide how to allocate tasks between themselves. The organization of work — in particular, the allocation of the Primary and Secondary systems between the operators — is far more flexible.

Since the plant is a highly integrated technical complex, the activities of the two operators are complementary and interdependent. In fact, "two operators are really needed to control the process" (p. 281). Therefore, in order to carry out their work, they must act in a highly coordinated fashion, and to do so each them needs access to reliable information on the state of the plant.

In traditional control rooms in nuclear power plants, information on the state of the plant is displayed on a panel that is several meters long; it is located in a room in which the two operators both work. By contrast, in the S3C control room design each operator has a computer workstation. While these workstations provide access to all relevant control data, S3C have some disruptive effects on the articulation work required to control the plant. At the beginning of the session, the operators normally agree on the allocation of the Primary and Secondary systems. However, during their work, they often have to handle tasks concerning the side of the system initially assigned to the other operator. In a conventional control room this poses no problem. Each operator is continuously informed of the part of the process monitored by the other operator from the position of the other in relation to the instrument panel. From the changing positions of his colleague in the room, each of them can effortlessly infer what the other is up to. Furthermore, he only has to take a few steps to get a clearer idea of what is happening and in doing so he does not need to disturb the activities being carried out (p. 281).

> "Interactions between operators (oral exchanges, glances, movements to and fro), on the one hand, and the information sources available in the control room (alarms, pictures, mimic panel)

are the means the operators working in pairs use to monitor the overall process and/or the other's activity." (p. 282)

That is, the specific characteristics of the conventional interface to the control system of the plant provide cues for operators to develop and maintain the required reciprocal awareness without forcing them to resort to verbal communication.[1]

In S3C, however, the formation of this reciprocal awareness is not supported by the design of the control room. Thus, in S3C articulation work requires precise verbalization and conscious and perhaps disruptive workstation management activities (p. 281).

In sum, then:

(1) As in the case of the rolling mill, the field of work of the operators — the power plant — is strictly causally coupled. However, the control of the transformation of energy is mediated by a complex control system. Thus, whereas the manual process control of rolling mill imposes a rigorous temporal order on the activities of the operators, the operators of the power plant are engaged in 'supervisory process control' in the sense that they do not need to exercise control actions continuously (Sheridan and Ferrell, 1974; Rouse, 1980, p. 57)

(a) Because they are engaged in supervisory process control and their activities are not subjected to any rigorous temporal order, they can reallocate tasks between themselves.

(3) Also because they are engaged in supervisory process control, operators have no direct evidence of the state of the plant in the sense that they have no direct perceptual evidence of the energy transformation process. They rely on representations in the form of sensors and effectors to monitor and regulate the transformation process. What each of the operators is doing at any given time is therefore essentially invisible to the other. This impedes the formation of the reciprocal awareness required for articulation work.

(4) Because the design of the conventional control room forces operators to move around in space in order to monitor and regulate the plant, it also supports the formation of reciprocal awareness. Again, like the operators of the hot rolling mill, they do not move to a particular panel in order to indicate to the other what they intend to do; their movements in the room are motivated by the requirements of the control function at any given time. However, unlike the operators of the hot rolling mill, they are able to develop and maintain a reciprocal awareness by observing each other.

---

1 A similar observation is given by Perrow: "I am told that one of the advantages of old-style steam valves, where the steam of the valve will rise when opened, is that a quick glance by an operator or supervisor over a huge room will show which valves are open, and which are shut — they just stick up when open. In an employee heads for the wrong valve after misunderstanding an order, that will be quite visible too. In complex systems, where not even a tip of an iceberg is visible, the communication must be exact, the dial correct, the switch position obvious, the reading direct and 'on-line'" (Perrow, 1984, p. 84).

(5) Because the control of the energy transformation process is mediated by the automatic control system of the plant, the operators are able to articulate their activities in a flexible way: In addition to being able to develop and maintain reciprocal awareness by observing each others movements, they are able to communicate verbally and visually. They are thus able to negotiate in ambiguous situations.

1.3.3. Supervisory process control: The London Underground control room

The studies by Heath and Luff of cooperative work in Line Control Rooms on London Underground (Heath and Luff, 1991; Heath and Luff, 1992) provide detailed insight into the workings of the formation of reciprocal awareness among the operators of the line.

The operators in the control room coordinate train traffic and movement of passengers on a particular line, in this case the Bakerloo Line. The control room can house several staff, but concern here is with two main actors: the Line Controller who coordinates the day-to-day running of the railway and the Divisional Information Assistant (DIA) who, amongst other things, provides information to passengers and to Station Managers.

Both operators are able to monitor the state of the Bakerloo line traffic on a real-time display, a 'fixed line diagram', which runs the length of the room. In addition, a paper timetable specifies train numbers, times, and routes; crew allocations, shifts, and travel; vehicle storage and maintenance; etc. The Controller can contact train drivers via a radio system. The DIA, on the other hand, can monitor platforms via a closed circuit television (CCTV) and provide information to passengers via a Public Address system. In addition the DIA can establish contact with Station Managers by touch-screen phone.

Coordination of train traffic and passenger movement is a domain specific characteristic of rapid urban transport:

> "unlike others forms of transport, rapid urban transport systems do not provide a timetable to the public. Instead, passengers organise their travel arrangements on the assumption that trains will pass through particular stations every few minutes. When such expectations are broken, or travellers are unable to change at certain stations, or have to leave a train because the line is blocked, then the DIA needs to provide information and advice. The nature of such announcements varies with the circumstances of, and reasons for their production." (Heath and Luff, 1992, p. 74)

Because the two controllers have to coordinate the movements of trains and passengers speedily and with minimal discomfort to the public, the activities of the Controller and DIA require extremely close collaboration. Accordingly, the operators have developed "a subtle and complex body of practices for monitoring each other's conduct and coordinating a varied collection of tasks and activities" (Heath and Luff, 1992, p. 73). One element of this informal, implicit and yet systematic articulation of responsibilities and tasks is "an emergent and flexible di-

vision of labour which allows the personnel to lend support to the accomplishment of each others' tasks and activities and thereby manage difficulties and crises" (pp. 73 f.).

As in the case of the operators of the nuclear power plant, the operators of the Bakerloo Line need to be able to articulate their activities tacitly:

> "It is relatively unusual for the Controller or the DIA to tell each other what tasks they are undertaking or explicitly to provide information concerning: the changes they have made to the service, the instructions they have provided to other personnel, or the announcements they have made to passengers. Indeed, given the demands on the Controller(s) and the DIA, especially when dealing with emergencies or difficulties. it would be impossible to abandon the tasks in which they were engaged explicitly to provide information to each other as to what they were doing and why. And yet it is essential that both Controller and DIA remain sensitive to each other's conduct, not only to allow them to coordinate specific tasks and activities, but also enable them to gather the appropriate information to grasp the details of the current operation of the service." (Heath and Luff, 1992, p. 74).

Heath and Luff (p. 75) provides a striking example of tacit development of reciprocal awareness:

	… Controller calls Driver …
Controller:	Control to the train at Charing Cross South Bound, do you receive?
	… Controller switches monitor to the platform …
Controller:	Control to the train at Charing Cross South Bound, do you receive?
Driver:	Two Four O Charing Cross South Bound
Controller:	Yeah, Two Four O. We've got a little bit of an interval behind you. Could you take a couple of minutes in the platform for me please?
Driver:	(( )) Over
Controller:	Thank you very much Two Four O.
DIA:	Hello and good afternoon Ladies an Gentlemen. Bakerloo Line Information…

> "The announcement emerges in the light of the DIA overhearing the Controller's conversation with the driver and assessing its implications for the expectations and experience of travellers using the service. He transforms the Controller's request into a relevant announcement by determining who the decision will effect and its consequences. In this case, this is particularly the passengers at Charing Cross whose train is delayed as a consequence of a problem emerging on the Southbound service. […] The DIA does not wait until the completion of the Controller's call before preparing to take action. Indeed, in many cases, it is critical that announcements are delivered to passengers as Controllers are making adjustments to the service. In the case at hand, as the call is initiated, we find the DIA progressively monitoring its production and assessing the implications of the Controller's request for his own conduct. The technology, and in particular the fixed line diagram, provides resources through which the DIA can make sense of the Controller's actions and draw the necessary inferences. At the onset of the call he scans the fixed line diagram to search for an explanation, or provide an account for, why the Controller is contacting a driver and potentially intervening in the running of the service. By the Controller's second attempt to contact the driver, the DIA is moving into a position at the console where he will be able to reach the operating panel for the Public Address

> system and if necessary make an announcement. On the word 'couple', at which point he can infer the potential delay that passengers might incur, he grabs the microphone and headset in preparation for the announcement. In consequence, even before the Controller's call to the driver is brought to completion, the DIA has set the Public Address system to speak to the passengers on a particular platform and is ready to deliver the announcement." (Heath and Luff, 1992, pp. 75 f.)

In the example given above, the DIA's very looking for evidence is motivated and driven by virtue of the Controller's attempt to call a driver:

> "Activities such as telephone conversations with personnel outside the room, tracking a particular train with the CCTV, or discussions with Line Management concerning the state of the service, are, at least in part, publicly visible within the local milieux and ordinarily the bits and pieces available can be used to draw the relevant inferences." (Heath and Luff, 1992, p. 79)

Having noticed the Controller's attempt to call a driver, the DIA scans the fixed line diagram in order to provide an account for the upcoming intervention. That is, the DIA is not only able to overhear the Controller and assume that they have mutual access to the same information displays, but is also able to discern, through "peripherally monitoring the actions of his colleague", where the Controller might be looking and what he might have seen. "The various information displays, and their use by particular individuals, is publicly visible and can be used as a resource in determining courses of action and for the mutual coordination of conduct." (p. 76)

For the operators to make sense of what each other is doing, the activities of the other must be interpreted in relation to the state of the field of work. Thus, the formation of the reciprocal awareness requires access to (much of) the same evidence regarding the current state of the field of work (the movement of trains, passengers etc.): "The fixed line diagram and the station monitors, provide an invaluable resource for the DIA in producing an account for his colleagues' interventions in the running of the service" (p. 76). In particular, the common availability of various sources of information in the Line Control Room allows the DIA to assume that the current problems in the operation of the service noticed by the Controller are similarly available to the himself if he scans the various displays.

> "The 'public' availability of the technology within the Control Room, whether it is a fixed line diagram, a CCTV screen, a screen-based line diagram or an information display, and the visibility of its use, provide critical resources in the collaboration between Controller and DIA. […] More importantly perhaps, the DIA and Controller can use the common sources of information as a reliable means of accounting for a broad range of actions and tasks undertaken by the other. […] Moreover, their use of the fixed line diagram and the surrounding monitors of the console is publicly visible, and can be used to determine a particular activity in which the DIA or Controller is engaged, or, […] to display a potential problem which is emerging within the operation of the service. The mutual availability of the various information displays, and the visibility of their use, are important resources for making sense of the actions of a colleague and developing a coordinated response to a particular incident or problem." (Heath and Luff, 1992, p. 76)

Now, the formation of reciprocal awareness is not only the product of a — more or less — passive (visual and auditory) monitoring of what other is doing

but involves complementary pro-active process of conveying cues of one's own activities and concerns. Thus, where activities (such as reading the timetable or entering the details of incidents on the various logs) are less visible, the details of the activity may not be available to a co-participant. Making such 'less visible' activities accessible to colleagues may for example involve reading or thinking aloud, humming, and so forth. The London Underground case provides an excellent example of how one operator actively directs the attention of another to some particular feature of the state of the field of work in a way that is more direct and effective than merely marking certain objects but still unobtrusive and inconspicuous:

> "On occasions, it may be necessary for the Controller to draw the DIA's attention to particular events or activities, even as they emerge within the management of a certain task or problem. For example, as he is speaking to an operator or signalman, the Controller may laugh or produce an exclamation and thereby encourage the DIA to monitor the call more carefully. Or, as he turns to his timetable or glances at the fixed line diagram, the Controller will swear, feign momentary illness or even sing a couple of bars of a song to draw the DIA's attention to an emergent problem within the operation of the service. The various objects used by the Controller and DIA to gain a more explicit orientation from the other(s) towards a particular event or activity, are carefully designed to encourage a particular form of co-participation from a colleague, but rarely demand the other's attention. They allow the individual to continue with an activity in which they might be engaged, whilst simultaneously inviting them to carefully monitor a concurrent event." (Heath and Luff, 1992, p. 81)

Now, in spite of the enormous flexibility, efficiency, and effectiveness of these informal and implicit modes of interaction, the coordination of the myriad activities of the Bakerloo Line *at large* is far too complex, far too distributed in space and time, and involves far too many actors and specialties to be managed by means of these modes of interaction. These large scale cooperative activities are basically managed by means of a timetable:

> "The Underground service is coordinated through a paper timetable which specifies: the number, running time and route of trains, crew allocation and shift arrangements, information concerning staff travel facilities, stock transfers, vehicle storage and maintenance etc. Each underground line has a particular timetable, though in some cases the timing of trains will be closely tied to the service on a related line. The timetable is not simply an abstract description of the operation of the service, but is used by various personnel including the Controller, DIA, Signalmen, Duty Crew Managers, to coordinate traffic flow and passenger movement. Both Controller and DIA use the timetable, in conjunction with their understanding of the current operation of the service, to determine the adequacy of the service and if necessary initiate remedial action. Indeed, a significant part of the responsibility of the Controller is to serve as a 'guardian of the timetable' and even if he is unable to shape the service according to its specific details, he should, as far as possible, attempt to achieve its underlying principle: a regular service of trains with relatively brief intervening gaps." (Heath and Luff, 1992, pp. 72 f.)

As a 'mechanism of interaction', the timetable requires continuous management by the operators:

> "The timetable is not only a resource for identifying difficulties within the operation of the service but also for their management. For example the Controller will make small adjustments to the running times of various trains to cure gaps which are emerging between a number of trains during the operation of the service. More severe problems such as absentees, vehicle

> breakdowns or the discovery of 'suspect packages' on trains or platforms, which can lead to severe disruption of the service, are often successfully managed by reforming the service. These adjustments are marked in felt pen on the relevant cellophane coated pages of the timetable both by the Controller and the DIA, and communicated to Operators (Drivers), Signalmen, Duty Crew Managers and others when necessary." (Heath and Luff, 1992, p. 73)
>
> "Perhaps the most critical activity within the Line Control Room […], is rewriting the timetable; a process known as 'reforming' the service. Almost all problems which arise in the operation of the service necessitate 'reformations', where the Controller, actually within the developing course of an event, reschedules particular trains, their crews, and even their destination, so as to maintain, for the practical purposes at hand, a relatively even distribution of traffic along the line." (Heath and Luff, 1992, p. 79).

However, as opposed to changes in the state of the field of work as represented by the fixed line diagram or the platform monitors, changes made to the timetable are not immediately and automatically conveyed to the other operators. Thus the distributed management of the timetable may give rise to inconsistencies in the cooperative operation of the line. It is thus "critical that the DIA and others receive information concerning changes to the timetable, otherwise they will not only misunderstand the current operation of the service, but take the wrong courses of action" (p. 73).

The operators — in casu the Controller — handle this by thinking aloud when his is making changes to the timetable:

> "It is essential that both colleagues within the Line Control Room, and personnel outside such as Duty Crew Managers, drivers and even Station Managers, are aware of these changes. Otherwise, these staff will not only fail to enact a range of necessary tasks, but will misunderstand the state of the service and make the wrong decisions. Reforming the service however, is an extremely complex task, which is often undertaken during emergencies, and it is not unusual for the Controller to have little time explicitly to keep his relevant colleagues informed.
>
> One solution to this potential difficulty is to render features of their individual reasoning and actions 'publicly' visible by talking though the reformations whilst they are being accomplished. The Controllers talk 'out loud', but this talk is not specifically directed towards a colleague within the Control Room. Rather, by continuing to look at, and sketch changes on the timetable, whilst producing talk which is often addressed to oneself, the Controller precludes establishing a 'recipient' and the interactional consequences it would entail. Talking through the timetable, whilst rendering 'private' activities 'publicly' visible, avoids establishing mutual engagement with colleagues which would undermine the ongoing accomplishment of the task in question. Consider the following fragment in which the Controller finishes one reformation and then begins another.

	*…Controller **reads** his timetable …*
Controller:	It's ten seventeen to ( ) hhhhhhh
	(4.3)
Controller:	Right (.) that's that one done.
Controller:	hhh hhh (.) hhh
Controller:	Two O Six ( ) Forty Six
	(0.7)
Controller:	Two Two Five
	… the DIA begins to tap on his chair and the trainee begin a separate conversation. As they begin to talk the Controller ceases talking our loud …

Whilst looking at the timetable, the Controller announces the completion of one reformation and begins another. The Controller talks numbers, train numbers, and lists the various changes that he could make to the 206 to deal with the problems he is facing, namely reform the train to ~46 or to 225. As the Controller mentions the second possibility, the DIA begins to tap the side of his chair, and a moment or so later, discusses the current problems and their possible solutions with a trainee DIA who is sitting by the DIA's side. As soon as the DIA begins to tap his chair and display, perhaps, that he is no longer attentive to his colleague's actions, the Controller, whilst continuing to sketch possible changes on the timetable, ceases to talk out loud. Despite therefore, the Controller's apparent sole commitment to dealing with specific changes to the service, he is sensitive to the conduct of his colleague, designing the activity so that, at least initially, it is available to the DIA and then transforming the way the task is being accomplished so that it ceases to be 'publicly' accessible.

Whilst 'self talk' may primarily be concerned with providing co-present colleagues with the necessary details of changes made by the Controller to the running order of the service, it is interesting to observe that a great deal more information is made available in this way than simply the actual reformations. […]□[T]he Controller renders visible to his colleagues the course of reasoning involved in making particular changes. The natural history of a decision, the Controller's reasoning through various alternative courses of action, are rendered visible within the local milieu, and provides colleagues with the resources through which they can assess the grounds for and consequences of 'this particular decision' in the light of possible alternatives. While the Controller is talking out loud, it is not unusual to find the DIA following the course of reasoning by looking at his own timetable, and where necessary sketching in the various changes which are made. In this way, DIA and Controller, and if present, trainees and reliefs, assemble the resources for comprehending and managing the service, and preserve a mutually compatible orientation to the 'here and now', and the operation of the service on some particular day. The information provided through the various tools and technologies, including the CCTV monitors, the fixed line diagram, and information displays, is intelligible and reliable by virtue of this collaborative activity." (Heath and Luff, 1992, pp. 79-81)

In sum, then:

(1) The field of work of the operators in the Bakerloo line control room, i.e., the trains and the infrastructure of the line on one hand and the passengers on the other, is not causally coupled in the same strict sense as the hot rolling mill and the nuclear power plant. Rather, the general function of the line operators is to establish a very close coupling of the movement of trains and passengers so as to provide the required quality of service to the passengers.

(2) The operators have a variety of means of monitoring the state of the field of work: automatic (the fixed line display), semi-automatic (CCTV) and

manual (time tables). Due to their adjacent positioning, they are thus to some extent able to develop a reciprocal awareness by visually monitoring each other's activities.

(3) The various information displays, and their use by particular individuals, are publicly visible and can therefore be used as a resource in determining courses of action and for the mutual coordination of conduct. The operators can use the common sources of information as a reliable means of accounting for a broad range of actions and tasks undertaken by the other. The mutual availability of the various information displays, and the visibility of their use, are important resources for making sense of the actions of a colleague and developing a coordinated response to a particular incident or problem.

(4) The operators do not regulate the state of the field of work by means of effectors or other control mechanisms. Rather, they regulate the state of the field of work by means of talking with train drivers, station managers, and passengers via radio and telephone. Accordingly, the two operators can develop and maintain a more rich and accurate reciprocal awareness by overhearing to each other's conversations over telephone or radio.

(5) The operators direct the attention of their colleague to certain features or events in myriad ways: by modulating their conversations with third parties, by humming or singing, by gazing etc.

(6) The large-scale cooperative operation of the Bakerloo Line as a whole is basically managed by means of a timetable. As a 'mechanism of interaction', the timetable requires continuous management by the operators. This management of the timetable is itself a cooperative activity whose articulation may require the application of the whole repertoire of modes of interaction.

1.3.4. Air traffic control: The flight strips

A comprehensive study on cooperative work in air traffic control conducted over several years by researchers at the Department of Sociology at the University of Lancaster has provided interesting insights into the delicate and multifaceted handling of an artifact — the flight strip — to facilitate fluent and dynamic articulation work (Hughes et al., 1988; Harper et al., 1989a; Harper et al., 1989b; Harper et al., 1991; Hughes et al., 1992; Harper and Hughes, 1993; Hughes et al., 1993; Harper et al., Forthcoming).

The whole of the airspace of England and Wales is controlled from a center near London.

In order to handle the complexity of ensuring orderly and safe air traffic, the airspace is divided into sectors. Thereby, a moderate degree of coupling of the system has been achieved (Perrow, 1984, pp. 159 f.).

The capacity of a given sector is determined by the workload limit of controllers. On the other hand, a further subdivision of the airspace into a larger number of smaller sectors is not feasible, however, due to the increased overhead of coordination activities that would entail. Operators controlling different sectors are interdependent in their work in that the state of one sector will have consequences for the state of an adjoining sector and vice versa (Harper et al., 1989a, p. 5). As the number of sectors increases, so too do the coordination and handover elements of the workload, so that the potential gain is negated (Hughes et al., 1988, pp. 33 f.).

Two types of sectors are distinguished. The *en route* sectors are the 'lanes' between major junctions of the airways. Thus, the bulk of traffic is *en route* to somewhere beyond the sector. By contract, Terminal Maneuvering Areas (TMAs) are sectors at the confluence of major airways and over busy airports.

The majority of aircraft travel through a sector only for an average of 20 minutes at a time. On busy *en route* sectors there can be up to 30-40 planes an hour. The speed of modern jet aircraft means that air traffic controllers have to make decisions quickly to maintain adequate separations; decisions that, regularly and routinely, have to deal with various departures from standard profiles, such as failure to achieve expected levels and pilots mishearing instructions. However, even under ordinary controlling conditions, air traffic control has a *discretionary character* (Hughes et al., 1988, pp. 19 ff.):

The objective of air traffic control is to direct flights safely and efficiently to their destinations. Safety is basically attained by keeping aircraft separated from each other, and the standard distances for this are five miles laterally and 1,000 feet vertically. Efficiency, on the other hand, is attained by directing aircraft to follow the shortest available route. Unavoidably, there is an occasional conflict between these two aims and it is up to the controllers to maintain the first by sometimes compromising the latter using the procedural framework of the Air Traffic Control system. Hence the discretionary character of ATC work: The "implementation of the rules of control consist in working within them with respect to a particular configuration of aircraft at any one time; a configuration that may be, as it were, 'typical' but which may also display 'untypical' and unpredictable characteristics." (Hughes et al., 1988, p. 20). The specific complexity of air traffic control as a work domain, then, is a function of the potential risk of midair collisions, the time-constraints under which decisions must be taken, and the number and configuration of aircraft in the air.[1]

The complexity of controlling a sector is not primarily related to the number of aircraft in the sector but rather to the specific "configuration" of the sector:

> "Although there is a limit to the total number of aircraft a controller can handle at once, it is the configuration of those aircraft which is the more important feature attended to in controlling. Roughly speaking, the more complicated that configuration — with cross-overs,

1 "In ATC processing delays are possible; aircraft are highly maneuverable and in three-dimensional space, so an airplane can be told to hold a pattern, to change course, slow down, speed up, or whatever" (Perrow, 1984, p. 160).

planes ascending and descending, changing direction, and so on — the more difficult is coordination." (Hughes et al., 1988, p. 20).

Controlling a sector requires the cooperative effort of several operators. Thus, each sector is normally, that is when traffic is relatively light, controlled by a team consisting of two air traffic controllers, two assistants, and one sector chief. During busier periods, sectors may be split according to the intensity and the complexity of the traffic. On the other hand, when traffic is very light, usually at night, the two sides of a sector may be controlled by only one controller under the supervision of a chief and with the help of another controller or 'wingman' (Hughes et al., 1988, pp. 104 f.). Air traffic control thus requires a complex cooperative work arrangement with an elaborate and dynamically changing system of division of labour. .

The control of any one sector is executed at a particular "suite", i.e., a workstation equipped with radar screens, telephone and radiotelephone communication facilities, maps, computer input and printout stations, and racks for flight progress strips. (See Figure 1-5).

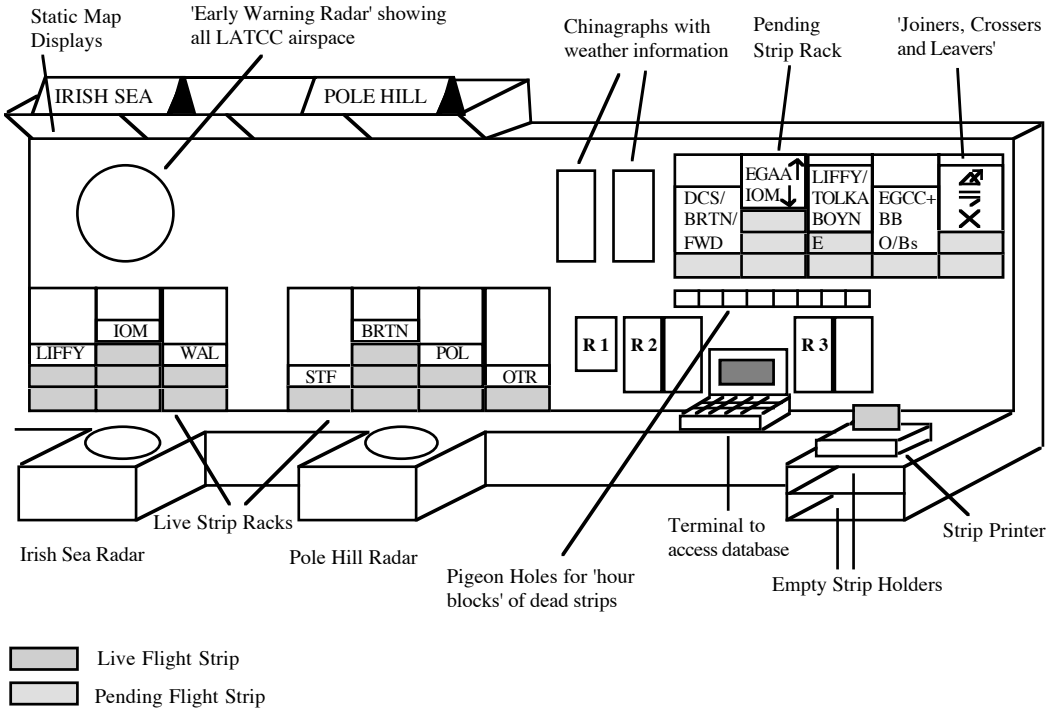


Figure 1-5. Layout of an en-route suite: Pole Hill and Irish Sea sectors. (Hughes et al., 1992)

The radar system, in addition to representing radar contacts as 'blips', displays a 'data block' alongside each blip showing the flight number and flight level of the aircraft (based on data provided by the responder of the aircraft). Furthermore, some screens display a trail of 'blips' representing the heading of each particular flight. The radar system is used to provide "an accurate representation of movement in airspace", that is, a representation of the current state of the field of work

at any moment. “Cognitively, the screen is a technological representation of a slice of sky and the relevant events occurring within it. The orderliness of the screen stands proxy for the orderliness of the sky ” (Hughes et al., 1988, p. 40).

However, the radar screen is uninterpretable without the further information for each flight embodied in the flight progress “strips”, especially “the goals, intentions and plans of pilots and controllers and their recent actions” (Harper et al., 1989a, p. 10□). The strips are made of card, approximately 200 by 25 mm and divided into fields (see Figure 1-6). The information in the fields comes from a database holding the flight plan filed by the pilot prior to departure, sometimes modified by inputs keyed in by controllers or assistants. It includes the aircraft’s callsign, its flight level, its heading, its planned flight path, the navigation points on its route and its estimated time of arrival, the departure and destination airports, and the aircraft’s type. Strips are arranged in racks immediately above the radar screens. The racks are in turn divided into bays themselves separated by fixed markers representing particular navigation points in the sector.

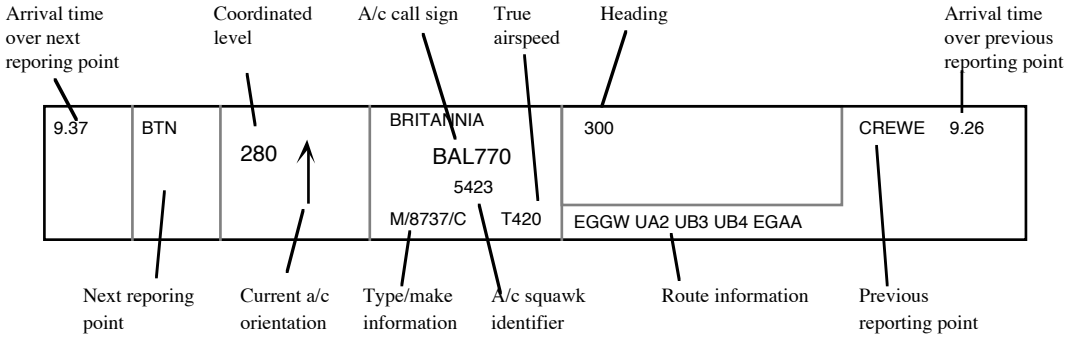


Figure 1-6. A flight progress strip. (Graphics made by the author on the basis of a photocopy of a specimen provided by the Lancaster group).

A strip first comes to the attention of the controller when it is placed on the strip bay. New strips are initially treated as ‘pending strips’ and placed above a special strip denoting the navigation point marking the boundary of the sector. Once the aircraft in question calls the controller, the relevant ‘pending’ strip is placed beneath the marker strip. It is now ‘live’. The collection of pending strips enables a controller to gauge at a glance what traffic is due in the sector and, consequently, how busy he or she is likely to be, either because of the number of strips stacked or the complexities indicated. For example, “noting among the ‘pending strips’ a fast aircraft curving along a route but not yet entering the sector in which there is already a slower aircraft, and therefore the possibility of an overtaking problem, is to use the ‘pending’ strips to *see* ‘the state of the sector’ *now* in terms of the likely problems ‘in a few minutes time’.” (Harper et al., 1989a, p. 13□). Pending strips are thus part of the controller’s “horizon of attention” in the sense that it “incorporates the future in relation to the present ‘state of the sector’” (Harper et al., 1989a, p. 40). The controller will mark the pending strips for any features deemed appropriate to note, such as aircraft speed, route and airway,

destination, or whatever. The strips may be ordered according to their pending times and strips indicating a procedural conflict may be lifted slightly out of position (‘cocked out’).

Because of the potential risks involved, the time-critical nature of the control task, and the exigencies of traffic flow, the overriding concern of air traffic controllers is to organize their work in such a way that they provide themselves with ‘thinking time’ so that when there is least ‘thinking time’ there will be ‘least to think about’ (Harper et al., 1989a, p. 11□). Thus, pending strips are organized in such a way that they are “made relevant” to what the controller anticipates will happen and plans to do in what order and with what priority:

> “Ordering the strips is […] a way of shaping attention in terms of the particularities that make up the ‘current state of the sector’ including such matters as traffic densities, standard traffic patterns, and any special problems such as VIP flights that need to be anticipated. This preliminary ordering of the strips is not determinatively related to real time events ‘in the sky’ as represented on the radar screen since, at this stage, the aircraft referred to by the strips are ‘not yet the business of this sector controller’. This is very much a preparatory, but vital, ordering.” (Harper et al., 1989a, p. 17).

By contrast, the revision of live strips is seen as organizing what the controller is immediately about. The revisions on live strips are not noticings of what is likely to be the case but records of what has been done. When a controller gives an instruction to a pilot, for example to ascend to flight level 220, he or she marks this on the strip. In this case, the mark is an upwards arrow and the number 220. When the pilot acknowledges the instruction, the controller crosses through the old flight level on the strip. When the new flight level is attained, the controller marks a check beside it. Changes in heading, estimated time of arrival, route, call sign, etc. are dealt with in similar ways. The information contained on the strip is thus systematically altered as instructions given and acknowledged or as up-dates on aircraft now being dealt with. “In this manner not only is the ‘state of the sector’ displayed and seeable ‘at a glance’ by those equipped to read the strips, but a written, and reproducible, record of actions taken is created” (Harper et al., 1989a, pp. 14 f.). Or in the words of one of the controllers interviewed by Harper and associates, strips “are like your memory, everything is there” (p. 49).

As opposed to radar, strips are not an automatically updated representation of the state of the sector. Nor do strips “determine the sequence of tasks controllers perform” in the same way as the work of the operators of the hot rolling mill is subjected to a rigorous temporal order:

> “Although the strips are produced according to the order in which designated aircraft will reach defined markers, the controller has to organize the strips so that they can become an instrument that serves, and makes possible, controlling work. Strips are ‘manipulated’, ‘taken note of’, ‘ignored for the moment’, ‘revised’, and more, continuously throughout the time that they are in use.” (Harper et al., 1989a, p. 16).

Strips are organized and annotated according to a standardized format:

- The categories of information on the strip and its general typographical layout follow a standard format.

- The color of the strip holder is used to effect a two-fold categorization into east and west bound traffic.
- The color of the strip paper is used to effect a two-fold categorization into 'standard' and 'crossing' or military traffic.
- The color of hand notations on the strip is used to effect a two-fold categorization distinguishing annotations made by the controller from annotations made by the sector chief.
- In general, annotations follow an elaborate set of conventions specified in the *Manual of Air Traffic Services* (e.g., upward or downward arrows, check marks, crosses through numbers).

Because of their "formatted character", "strips provide a *template* for noting and recording what is happening and will happen in the sector." (Harper et al., 1989a, pp. 15 f.). More than that, also because of their "formatted character", strips serve as a resource for articulating the activities of the different members of the sector team. In maintaining a constantly up-dated representation of the 'state of the sector' in terms of the standard categories of information on the strip itself and in the standard format and notation, the controller is not just providing information relevant to his or her own work but is also providing what Harper and associates calls "team relevant information":

> "The conventionality of the strips, their standard format, the known-in-common notations used, make the work the strips represent visible, in *multi-functionalities*, to other members of the sector team, other controllers, chiefs, assistants, etc.; in fact, to anyone enculturated in the work of controlling. In making manifest 'actions so far' in ways that are 'seeable at a glance' by those equipped and needing to know, the strip plays a key role in the team features of the social organisation of controlling work." (Harper et al., 1989a, pp. 15 f.)

The strip thus serves as a 'note pad' for all the team members, any of whom may write on it. "The standard format of data, colour coding of holders, role distinctive markings, the placement of the racks, among other features, means that, again 'at a glance', what the strip signifies in terms of controlling tasks is available to other members of the team" (Harper et al., 1989a, p. 45).

Thus, while being an important means of anticipating and controlling the state of the field of work as a whole, the strips in their racks simultaneously play a crucial role in the articulation of the work of members of the sector team. Controllers do not just attend to their 'own' strips but regularly read the strips, both pending and live, that 'belong' to their suite colleagues. Rather than being read closely, strips of colleagues will be read 'at a glance' to see if anything is indicated which might be of concern. This activity is related to 'buying time' by expanding the horizon of attention for a moment to obtain wider coverage. This tacit monitoring also enables the controller to asses how busy the colleague is and whether coordination can be achieved easily.

When necessary, overt coordination can be achieved by pointing to particular strips where there may be a problem, perhaps two flights due at the same navigation point at a similar time and at the same height. If the controller cannot attend to the request at that time, anyone who notices the problem can "cock out" the

strips, i.e., move them noticeably out of alignment in the racks. In a very inconspicuous and non-intrusive way, this makes it immediately obvious that, when it is time to deal with those flights, a problem will need to be considered, and to the practiced eye it will be obvious from a glance at the strips what the problem is (Hughes et al., 1992). "Such *noticings* do not need to specify just what specifically the problem might be; simply marking out the strip as 'something to take note of' is normally sufficient." (Harper et al., 1989a, p. 24).

The facility with which controllers can tacitly monitor each other's strips and each other's work with strips is, of course, "a function of the adjacent positioning" of the controllers on the suites. "This means that alterations or reference to strips 'on the other side' can be done with a minimum of explanation. […] Drawing the strip slightly out of position, pointing to it, making a notation, are 'sufficient' to draw attention to *that* strip, and its aircraft, and any problem it may represent and, as such, are actions manifesting the interdependence of one controller's activities with those of others." (Harper et al., 1989a, p. 24).

In providing means for representing the division of labour on the suite ("the conventional notations used denoting not only actions taken but by whom"), the 'formatted character' of flight strips provides yet another mechanism for articulating the cooperative effort of controlling a sector of airspace.

> "Alterations, whether by controllers or by the chief, are 'instantly' recognisable as alterations by 'such-and-such' and, therefore, informative about the kind of problem the 'noticing' might represent. So, the distinctiveness of markings on the strips, by calling into play the division of labour of controlling, also guides the attentiveness that makes the information as information to be 'noted and dealt with accordingly'." (Harper et al., 1989a, p. 27)

In sum, then:

(1) The regulation of the field of work of a particular sector team — i.e., the regulation of the sector and its changing configuration of aircraft — is done by means of radiotelephone communications with pilots. As in the case of the London Underground control room, the different operators at the suite can thus develop and maintain a reciprocal awareness of each other's activities by overhearing each other's conversations with actors 'in the field'.

(2) Each member of the sector team is able to interpret the activities of the others because the state of the field of work as represented by the radar displays and the ordered collection of annotated flight strips is publicly available to all members of the team.

(3) The flight strip is a vehicle for making activities persistently and publicly accessible by recording pertinent actions in the course of a flight through the sector. The flight strip supports and mediates the articulation of the work of the sector team, not only because of its public visibility, but also and most importantly because of its formatted character.

(4) Embedding cues in the objects belonging to or representing certain objects in the field of work, e.g. by highlighting certain flight strips (cocking out) is an non-intrusive and tacit way of directing attention. On

the other hand, the expressive power of materially embedded cues is limited. In the case of the flight strips, the 'categorical distinction' supported by highlighting strips is "effectively limited to two categories", namely relatively routine versus relatively problematic (Harper et al., 1989a, p. 30).

1.3.5. Production control in manufacturing: The MRP system

In 1990, Bjarne Kaavé conducted a study of cooperative production control in a manufacturing company we can call Alpha.[1] The company manufactures specialized optical appliances, and it covers about 50% of the world market for this category of equipment. The company currently produces about 6,000 units a year in 15 different models, each in 7 different variants.

The products manufactured by Alpha are fairly large units, between 1 and 2 cubic meters each; a unit typically weighs about 300 kilograms. Simply put, the product consists of a cabinet housing a complex rigging of electrical equipment, electronic circuits, and optical instruments. The metal cabinet is produced in-house, typically by cutting, bending, and welding sheet metal. Most metal parts go through more than five processes before entering the assembly process.

The operation of the plant involves a large number of different kinds of interdependent activities. The overall organization of the plant reflects this by being decomposed into a number of functional departments:

Shaping is a plant of it own. There are about 150 workers handling seven different types of processes.

Paint shop is working as a conveyer, where all parts are led through the paint cabin. They paint with five different colors, two colors covers about 85% of the parts, and a change of color takes two hours. Effectiveness depends only of the numbers of shifts from one color to another.

Sub-assembly employs about 25 workers, each handling a number of different assembly tasks. They work in series less than 50, and set up time for each task is negligible, but effectiveness rises with longer series.

Assembly is making the units according to the customers orders. In final assembly, each worker produces a complete unit from all the different parts and sub-assemblies. All fitters can handle more than one type of model, and the problem is not the set-up time, which again is negligible, but the fact that the number of errors increases if the fitters have to change model frequently.

Testing of a unit can be seen as a time delay, and the only management problem here is to choose the right sequence for testing the units.

Packing and Shipping. The units are shipped from the packing department, and the hard problem here is which customers orders to fulfill first.

---

1 This section is based on Bjarne Kaavé's findings as reported in his thesis (Kaavé, 1990) as well as in several joint analysis sessions with the present author.

A strategic decision made by Alpha in 1974 resulted in a very rapid change from production of one single model to production of several models (see Figure 1-7). The total number of models and variants are approximately 100. This shift in strategy dramatically changed the conditions of production control from that of a forecast-driven, production-for-inventory oriented strategy to an order-driven, 'just in time' strategy. Thus, the reference point in the planning process changed from number of units on stock to the time necessary to fulfill current customer orders. The company is now committed to a delivery time of three weeks.

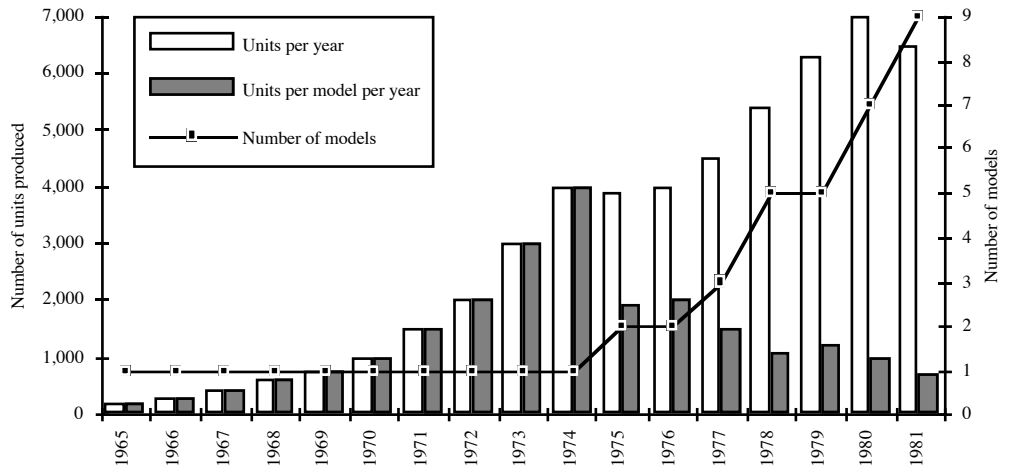


Figure 1-7. In 1974, company Alpha made a strategic decision to shift from production of one single model to production of several models. (Adopted from Barfoed (1982)).

Kaavé's investigations in Alpha provided an interesting case of how a conventional production management system — an MRP II or Manufacturing Resource Planning system[1] — is appropriated and used cooperatively in controlling a manufacturing operation with characteristics far beyond the valid application domain of the system.

Manufacturing involves multitudes of discrete parts and processes that are interdependent in multitude and enormously complex ways:

- each product consists of a many component parts, in some cases tens or hundreds of thousands of components;
- the production of each part may require a number of different processes in a specific sequence, and the production of different parts thus require different routings;
- different processes may require specialized tools and skills, and different parts thus compete for the same workstations;

1 In casu, the COPICS system from IBM (IBM, 1972).

- many products are being manufactured simultaneously, and at any given time a large number of products and their components coexist at different stages of completion;
- in modern manufacturing with a large number of different models and variants to be manufactured in small volumes at short notice, different models and variants are being manufactured simultaneously.

Thus, in the words of Harrington (1984), manufacturing can be conceived as "an indivisible, monolithic activity, incredibly diverse and complex in its fine detail. The many parts are inextricably interdependent and interconnected." Thus, as observed by Susman and Chase (1986), the various categories of workers — product designers, process planners, programmers, supervisors, operators, etc. — "will be highly interdependent with one another because of the need to exchange information to keep the factory operating." Accordingly, for a manufacturing enterprise to be able to adapt diligently and dynamically to changing conditions, the entire enterprise must react "simultaneously and cooperatively" (Harrington, 1979). Rapid and concerted adaptation of all the specialized functions of a diversified manufacturing operation, from Marketing to Shipping, to the vicissitudes of a volatile and complex environment is indeed the very essence of advanced manufacturing.

In conventional manufacturing, i.e., manufacturing of standardized products for fairly stable markets, the basic coordination mechanism of the diversified manufacturing operation — "the vital control center for the company's manufacturing planning and control system" (Gunn, 1981) — is computer-generated a "master production schedule" based on forecasts and standard lead times for the various parts. The planning algorithm decomposes the material requirements and computes the schedules for the production of each part, sub-assembly, and assembly. The production control problem is thus, in principle, reduced to executing the plan and adhering strictly to schedules. That is, in so far as the underlying forecasts are accurate, coordination across functions and departments can be mediated by plans and other organizational procedures. Direct cooperative interaction across functions only takes place to handle exceptional contingencies such as shortages or to expedite a high priority order.

At the heart of an MRP II system is a set of related models of essential aspects of the manufacturing operation. In order to understand the cooperative appropriation and management of the □MRP II system, it is essential to understand the nature of underlying models — even if it is industrial engineering textbook stuff:

Bills of Materials: A bill of materials is an ordered list of every part that makes up the finished product (see Figure 1-8). The bill of material thus states that a particular assembly is made up of several parts that, in turn, can be components or sub-assemblies.

Item #	Description	Quantity
79111	Mounting Kit, Deluxe	1
47342	Mounting Kit, Basic	1
76504	Bracket, cast	1
64333	Bolt, $\frac{5"}{16}$ □× □24⊠□1”	2
30751	Nut, $\frac{5"}{16}$ □× □24	2
22479	Washer, flat, $\frac{5"}{16}$	2
22842	Washer, lock, $\frac{5"}{16}$	2
16935	Clamp assembly	1
88327	U bolt	1
30750	Nut, self-lock, $\frac{5"}{16}$ □× □24	2

Figure 1-8. Indented bill of material for part #79111. (Gunn, 1981, p. 15)

That is, the bill of materials is a hierarchical or tree structure denoting all parts and their relationships (see Figure 1-9). Based on the bill of materials it thus becomes a simple calculation to foresee which parts are required in which quantities in order to produce a certain number of a given product.

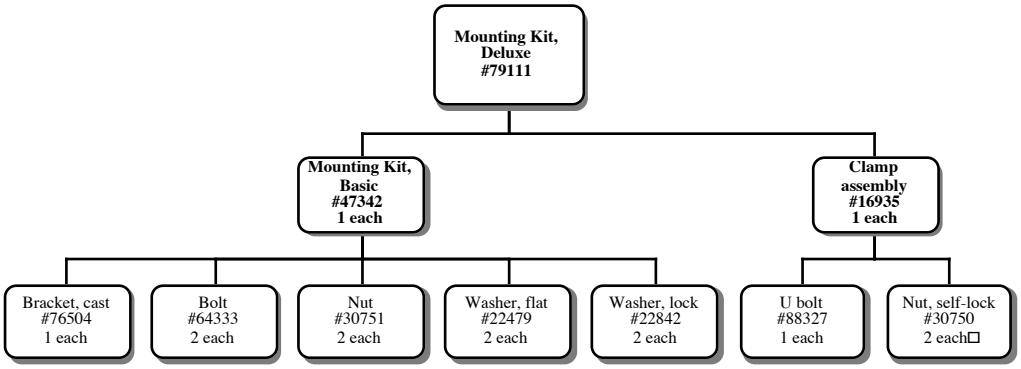


Figure 1-9. Product structure for part #79111 (Gunn, 1981, p. 15) showing the two-level hierarchy of parent parts (assemblies) and dependent parts (parts and sub-assemblies).

Routings or Process Sheets: The process sheet specifies and lists, in sequence, each manufacturing operation a part or sub-assembly must go through to be manufactured (see Figure 1-10). For each operation the average processing time is listed, based on a statistical study “performed over some production period, or from standard time study figures, or just the foremen’s best estimate” (Gunn, 1981, p. 25). However, the production cycle time does not indicate the number of labor hours required to produce the part. The number of labor hours required depends on the number of people required at each operation. Thus, the process sheets will often specify the average production cycle time as well as the average labor hours required.

Dept.	Description	Operation	Production cycle time	Labor hours required	Total labor hours required
100	Machining				
		Machine center 100		6	
		Work station 140		4	
		Machine center 150		2	
		Subtotal, dept. 100	2		12
200	Assembly				
		Work station 210		4	
		Work station 270		2	
		Subtotal, dept. 200	3		6
300	Crating				
		Work station 350		2	
		Subtotal, dept. 300	1		2
Total			6		20 hours

Figure 1-10. A routing or bill of labor for part #1050 (Gunn, 1981, p. 26).

Labor hours and calendar time is not necessarily the same, of course. That is, the time from a part enters a particular work station and until it arrives at the next work station is normally larger than the production cycle time as a direct function of labor hours required. In traditional manufacturing, a part was normally only processed 5% of time it spends underway from start to finish, and queue time alone could take 80-90% of the entire production cycle. In fact, the production cycle for each part involves several operational stages: apart from the time required for processing which involves set-up time and run time (e.g. the various machining or assembly operations), parts spend time in queues before and after processing and while being moved between work centers (see Figure 1-11).

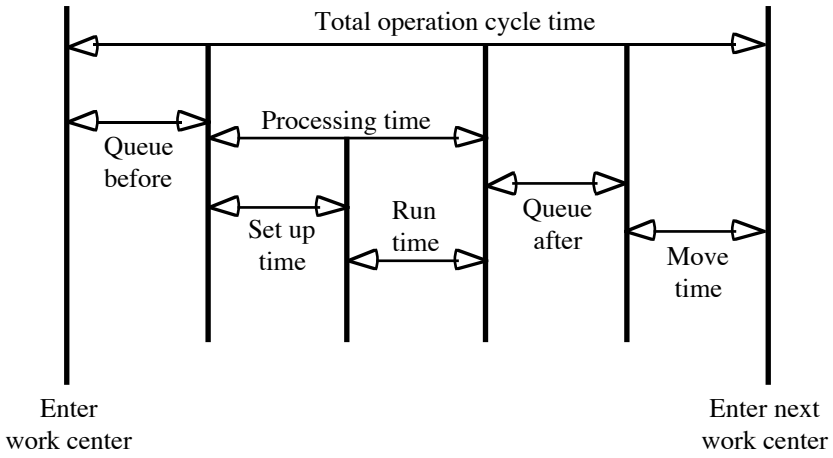


Figure 1-11. Elements of total operation cycle time (Gunn, 1981, p. 26).

The time required to produce different parts varies, of course. Thus, in order to produce a particular end item at a particular due date, the production of the different dependent parts needs to be initialized at different points in time. By coupling the model of the structure of parts and assemblies and the models of the routing and process characteristics of each part and assembly, it is then possible to calcu-

late the due date of the various components or dependent parts based on the due date of the orders of the end or parent item as well.

Since the operation cycle times for each part are specified in the routing sheet alongside the labor hours required, the total work time required for a batch of a particular part can be calculated as the queue time plus setup time plus the lot size multiplied by the run time per part. Thus, it is also possible to calculate the capacity requirements at each point in the production of a given end item.

The master production schedule is a time-phased plan identifying the aggregate capacity requirements at a company level. It is thus "the vital control center for the company's manufacturing planning and control system" (Gunn, 1981, p. 66). In conventional manufacturing, where production — to a large extent — has been planned on the basis of forecasts of expected sales, the master production schedule is computed on the basis of the sales forecasts for each product or family of products (plus, of course, actual sales demands) coupled with the models of product structure and the models of the routing and process characteristics of each part and assembly. Given a master production schedule based on reliable forecasts, the production control function is — for all practical purposes — reduced to ensuring that the detailed plan is executed properly.

A major problem with MRP II-based production control, however, is that is involves a significant element of guesswork:

> "a weakness of MRP is that there is some guesswork involved. You need to guess what customer demand will be in order to prepare the schedule, and you need to guess how long it will take your production department to make the needed parts. The system allows corrections to be made daily (called shop-floor control). Nevertheless, bad guesses results in excess inventories of some parts" (Schonberger, 1982, pp. 220 f.).

That is, MRP II-based production control is only feasible under certain conditions. The most important precondition is that the enterprise manufactures a limited number of products for a predictable market, that is, under conditions where deviations of sales from expected demand can be parried by inventories of final products.

These preconditions ceased to exist in Alpha as a consequence of the strategic shift in 1974. Nevertheless, the MRP II-based production planning and control system is still used intensively. Why?

The COPICS system at Alpha is used to generate production orders. In addition, it is used in order to record production orders that have been accomplished and to ensure that operators are paid when a production order has been carried out.[1]

The system still generates production orders based on projected demand (a simple monthly average) but actual customer orders are no longer assumed to match projected demand. To the contrary, the totality of COPICS-generated production orders are treated as a hypothetical plan that ensures that appropriate material resources in the form of parts and sub-assemblies can be made available in

1 The blue collar workers at Alpha are paid on the basis of a negotiated piece rate tariff.

order to meet actual demand as derived from the order book and propagated down the line.

Each decision maker then compares the set of COPICS-generated production orders with the actual demand. If customer orders fall within what has been planned by COPICS on the basis of average demand, the operators simply stick to the master schedule and executes the COPICS-generated production orders.

However, if customer orders are beyond the plan, operators override the plan and 'grab' a production order and redirect and reschedule it. Thus, when the actual customer order situation is not matched by the resource envelope established by COPICS, the network of local decision makers have to find out if and how the desired production can be made. This may result in a change in the sequence in which orders are produced or in the generation of new production orders and thereby in the creation of a different resource envelope.

In addition to the absence of reliable forecasts and the large number of product variants, there are other reasons for overriding the master schedule generated by COPICS.

First, some parts — the components of the cabinet — are extremely bulky. "They take up space like hell". These parts are not produced for buffer stocks. Their production is therefore always controlled manually. However, the equipment produced by Alpha only comes in three sizes, which means that the fluctuations in demand for the different models sizes are fairly small and cabinet parts can be produced in a flow-like manner.

Second, if suddenly some parts are missing — a fault may have occurred during painting — then that production order is 'carried by hand' all the way through the factory, that is, its production is controlled manually.

And third, changing from one color to another in the paint cabin requires that the paint cabin is thoroughly cleaned. So, when an order for, say, white products have to be made (only some 10 % of all products are painted white), it is important to make sure that all parts for that order are ready for being painted at the same time. This creates a wave throughout the plant: "We are painting white Friday — we need all parts to be painted white by then."

Anyway, whatever the reason for overriding the master schedule, the COPICS system is invaluable as a support to the operators in this cooperative manual control of production. As reported by Kaavé:

> "When he takes an order that only were to be carried out two weeks later, that resource withdrawal is recorded in the complex continuously controlled by COPICS. During the night, COPICS then tries to correct the accident — as it sees it — that suddenly 25 more right legs have been withdrawn than it had expected. It then computes — based on the MRP model — how to produce these parts anyway so that the plan is maintained. So, even if the system ought to break down if you do things like that, you are not severely punished.
>
> The MRP model in COPICS saves the planner every time he does a stunt like that. Because it tries to correct — it tries to maintain a kind of balance. Every time they steal something in one end it tries to resurrect balance.

> They exploit the fact that the MRP model generates new production orders when you withdraw parts. If you use some parts, the model enters and plan to produce some new ones. It does not know that the demand is fluctuating. It assumes that you need the same number every month.
>
> It pours water into the jar again. The jar is always refilled." (Kaavé, 1992)

In other words, what enables and motivates the operators at Alpha to use the MRP II system in a manufacturing operation that is basically order driven — as opposed to inventory driven — is the models of pertinent interdependencies of the manufacturing operation at Alpha: the vast array of Bills of Materials of all assemblies and sub-assemblies; the average set-up times and processing times of all manufactured parts and assemblies; the average lead time of purchased items etc. By incorporating such a set of related models of essential interdependencies, the MRP II system reduces the immense complexity of scheduling the production of a vast number of parts in a distributed setting by trying to maintain the fictitious master schedule based on projected average demand. It thus provides an essential service to an order-driven manufacturing operation.

However, in order for the MRP II system to be used in the way described above some important changes have been made to the design:

First, because COPICS-generated production orders are not treated as operational orders but rather as hypothetical orders, the production orders printed by COPICS are placed in "the box", that is, an archive of pending production orders. The archive is under the jurisdiction of the production planner. When the pending orders are to be executed, they are retrieved and finally placed in "the rack" where they are publicly visible and accessible. Operators are now allowed and expected to pick them up and produce the parts.

In other words, the *direct coupling* between (a) the state of the field of work for the cooperative production control activities (i.e., the manufacturing operations in its totality) and (b) the representation of the field of work in the MRP II system has been severed. The coupling is now merely hypothetical, pending human intervention (approval, modification, cancellation). Control of the production process is thus squarely in the hands of the cooperative ensemble.

Second, the COPICS system has itself been modified accordingly. It now has three check-boxes providing information about the status of each particular production order:

- o Has been printed
- o In the rack
- o Taken by engineer

The first check-box is marked automatically by COPICS when the production order is printed, whereas the production planner marks the second box when the

order is retrieved from the archive box and placed in the rack for execution. The latter box is marked by the foreman when the order is taken by an engineer.[1]

The MRP II system thus provides a means of perceiving the state of affairs in various parts of the plant.

The foreman in Assembly may for instance notice, while walking around, that 'his' buffer stock of some type of parts is low. He may think, "Next week we'll probably be out of these fuse sockets for Asia." He will then turn to COPICS to see if the system has planned an order that can be fulfilled in time for him to get them. He simply looks up in the system: "Does it look like I'll be getting a problem next week?" He can see that quite precisely. He can for instance see whether the sub-parts of that particular part are being machined or not or whether the production order has been placed on the board for the engineers to take it.

This way of cooperative production control points out new demands to production management systems, where the existing planning system must be supplemented with facilities that support local decision makers in their planning and evaluation tasks. The role of the planning system is simply changing as local actors are deciding which orders should be put into production, and new systems must enable local actors to generate orders when the set of expected production orders is not creating the needed resource envelope.

In sum, then:

(1) In the company we have called Alpha, the MRP II-based production planning system is used in a control mode based on actual customer order as opposed to a mode of control based on reliable forecasts of demand. The MRP II system is thus used in a context far beyond its valid application domain.

(2) The MRP II-based system used nevertheless to (a) to coordinate the mass of activities that need not be handled manually; (b) to coordinate the execution of customer orders well within the projected average demand; (c) to compute and counteract the potential adverse effects of manually overriding the hypothetical master schedule; and (d) to provide all those involved in the control of production at large with a representation of the state of the field of work.

(3) The MRP II-based system is able to provide this support because it incorporates a set of interrelated models of pertinent interactions and interdependencies in the field of work.

(4) The MRP II-based system as used in Alpha company has been redesigned so as to support its use as a mechanism of interaction: By introducing a manually controlled 'buffer' archive of printed production orders between COPICS and the production process and by recording

---

1 The engineers are allowed to chose which order to pick. This allows them, to a certain extent, to avoid changing the set-up at their individual workstations for each order and thus enable them to make a handsome wage. Their choice of job is to be approved by the foreman in order to ensure that the engineer does not skip rush orders for the sake of maximizing his income.

and displaying the state of each COPICS-generated production order, the coupling between the state of the field of work for the cooperative production control activities and the representation of the state of the field of work in the MRP II system has been made visible and accessible to those concerned.

1.3.6. Production control in manufacturing: The kanban system

The Alpha company studied by Kaavé provides another example of a mechanism of interaction in cooperative work: The kanban system.

In company Alpha a *kanban* system is used in certain processes the Shaping department. As noted above, the shaping department manufactures cabinets from sheet metal. Altogether seven different processes such as cutting, bending, welding etc. are involved. The processes of cutting, bending, welding etc., demand distinctly different machinery and tools, and this means that there are considerable set-up times, from about 15 minutes to less than hour, and each part are therefore produced in lots whose size is defined according to the overall flow and the set-up time on the specific machine. Lot sizes vary from process to process from up to 1,000 in the cutting processes to less than 50 in the welding and the sub-assembly processes. These processes are de-coupled by buffer stocks. Most metal parts are going through more than five processes before the assembly process.

Kanban is a Japanese word meaning ‘card’ or more literally ‘visible record’ (Schonberger, 1982, p. 219) and is now used to denote a production control system where a set of cards acts as the coordination mechanism, both as carrier of information about the state of affairs *and* as a production order conveying an instruction to initiate certain activities.

The basic idea is that loosely coupled but interdependent production processes can be coordinated by means of exchanging cards between processes (see Figure 1-12). A particular card is attached to a container used for the transportation of a lot of parts or sub-assemblies between work stations. When the operator has processed a given lot of parts and thus has emptied the container, the accompanying card is sent back to the operator who produces these parts. Having received the card he has now been issued a production order.

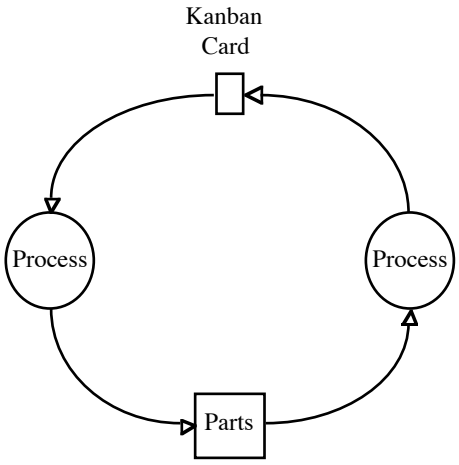


Figure 1-12. The basic 'syntax' of a kanban system.

The basic set of rules of a kanban system are (Schonberger, 1982, p. 224):

1. No part may be made unless there is a kanban authorizing it.
2. There is precisely one card for each container.
3. The number of containers per part number in the system is carefully calculated.
4. Only standard containers may be used.
5. Containers are always filled with the prescribed quantity — no more, no less.

Aoki (1988) conceives of the *kanban* or just-in-time system as a "semi-horizontal operational coordination mechanism" and argues that this mechanism of coordination is an effective way to adapt a distributed cooperative effort to changing market circumstances quickly without accumulating costly buffer inventories when many varieties comprising a large number of parts are involved. He adds the important observation that the semi-horizontal mode of coordination "crucially depends on the skills, judgment, and cooperation of [a] versatile and autonomous work force on the shop floor", and "a certain degree of blurring of job territoriality between workers on the one hand and foremen, engineers, programmers, etc., on the other".

However, in a *kanban* system, information only propagates 'up-stream' as parts are used down the line. The speed and pattern of propagation of information is severely restricted and the information ultimately conveyed has been filtered and distorted by the successive translations along the line up-stream. The *kanban* system does not provide facilities allowing decision makers to anticipate disturbances and to obtain an overview of the situation. They are enveloped by an overwhelming and inscrutable automatic coordination mechanism.

The *kanban* system is not adequate for coordinating manufacturing operations faced with severe demands on flexibility of volume. The *kanban* system can only handle small deviations in the demand for the end product (Schonberger, 1982, p. 227; Monden, 1983).

Accordingly, since Alpha is faced with extreme fluctuations in demand, operators constantly experience that the configuration of the *kanban* system (the number of containers per part number and the quantity per container) is inadequate. In stead of abandoning the kanban system altogether or at least temporarily, they are *change the configuration* in various ways, for example by pocketing a card for time, by leaving card on the fork-lift truck, by ordering new lots before a container has been emptied, by handing cards over directly, by changing lot sizes, etc.

This is possible because an informal network of clerks, planners, operators, fork-lift drivers, and foremen in various functions such as purchasing, sales, production, shipping etc. cooperate directly in controlling the flow of parts. A member of this network will for example explore the state of affairs 'up-stream' so as to be able to anticipate contingencies and, in case of disturbances that might have repercussions 'down-stream,' issue warnings. That is, the indirect, dumb, and formal *kanban* mechanism is subsumed under a very direct, intelligent, and informal cooperative coordination. The cooperative ensemble has 'appropriated' the *kanban* system in order to increase its flexibility. They have thus taking over control of the system, and are controlling the production far more closely and effectively than warranted by the *kanban* system. This is possibly because of their deep knowledge about lead times and inventories in the shaping processes, and the flow of information through the network of what Assembly needs.

It is important to notice, that the system might not work like this if it had been designed otherwise, for instance with sensors at the bottom of each container in order to detect and notify automatically when a new production order should be issued. This would make it impossible for the operators to change the set-up contingently.

In sum, then:

(1) The kanban system is a mechanism of interaction in the sense that it is a symbolic artifact that it used to reduce the complexity of articulating a large number of different cooperative work activities.

(2) As the MRP II-based system discussed in the previous section, the kanban system only serves as a mechanism of interaction because it incorporates (implicitly, in the configuration of the system) a model of pertinent features of the manufacturing operation.

(3) Thus, in spite of the fact that the kanban system often is used in situations where it is 'beyond its bounds', it is not discarded but merely modified locally and temporarily according to the requirements of the situation. When an operator pockets a kanban, he is *changing* the configuration of the system, not switching the system off. When the card is put back in circulation, the default configuration is in force again.

(4) In order to be usable in a setting like Alpha, the kanban system must to be managed (monitored, adapted, modified) continuously. This is

facilitated by the formation of a network of operators who keep each other informed about the state of affairs.

(5) The kanban system is tied to the field of work in such a way that state changes to the kanban system are strongly coupled to state changes in the field of work. The speed and pattern of propagation of information is severely restricted by the successive nodes along the line up-stream.

(6) The design of the kanban system as a mechanism of interaction allows operators to manage the system in so far as the control of the execution of the mechanism is in the hands of the operators: it is the down-stream operator who takes the card and sends it up-stream; it is the truck driver who delivers it; it is the up-stream operator who receives the card and decides to act on it or to deviate from the procedure in some way. That is, due to the operator's control of the execution of the kanban system and the fact that they exercise that control, the direct coupling of the system to the field of work has been severed.

(7) The information ultimately conveyed by the propagation of card up-stream has been filtered and distorted by the successive translations along the line. The *kanban* system does not provide facilities allowing decision makers to anticipate disturbances and to obtain an overview of the state of affairs up- and down-stream. They are enveloped by an overwhelming and inscrutable automatic coordination mechanism.

1.3.7. Portfolio management: The list

A portfolio management agency investigated by the author in 1987 (Schmidt, 1987) provides an example of the use of simple artifacts as vehicles of articulation work and of the cooperative effort required to manage the artifact.

Investors are faced with an extremely complex and turbulent object domain — the world market, or rather the complex of interrelated national and international markets for bonds, shares, options, futures, currencies, commodities, etc. Portfolio management agencies assist clients in making sound investment decisions, monitoring the state of their portfolios, and taking corrective actions as need be.

Portfolio management can thus be conceived of as a mediating function between a number of clients with different individual propensities and concerns on one hand and on the other hand an extremely complex and turbulent object domain.

Figure 1-13.

That is, on one hand portfolio management is faced with the monitoring and navigating the vast space of potential investment objects and their multi-dimensional interdependencies:

- Kinds of markets: shares, bonds, futures, options, currency, etc.
- National economies with different currency rates, interest rates, inflation rates, political prospects etc.
- Branches of production with different technical developments and market trends.
- Relations of ownership and control.

On the other hand, the different characteristics of clients add to the complexity of portfolio management:

- Clients have specific propensities and apprehensions.
- Clients have different 'risk profiles'.
- Clients have different needs for liquid assets.

In order to handle this complexity, the portfolio management agency studied by the author has adopted a fairly clear division of work and responsibility between 'analysts' and 'consultants' that reflects the polarity of the field of work. The analysts monitor the development of the markets in order to identify new investment opportunities as well as to monitor the changing degree of risk exposure of current investments, whereas the consultants monitor and control the individual portfolios in accordance with the propensities and concerns of their clients: they will sell and buy in order to keep the profile of the portfolio in line with that of the client.

Also in order to handle the compelling complexity, the portfolio management agency in question has developed effective strategies for reducing operational complexity.

As far as the complexity posed by the multiplicity of clients' propensities is concerned, the multiplicity is reduced drastically by classifying clients according to a very simple classification scheme:

- Risk profile, with only three categories: 'speculative', 'medium', 'conservative'.
- Needs for liquid assets, again with only three categories: 'high', 'medium', 'low').
- Specific commandments (e.g., 'No South African shares')

The complexity of controlling that the composition of each portfolio at any time is in accordance with the client's profile can thus be reduced to a relatively simple daily routine:

- The different needs for liquid assets can be attained by establishing and maintaining a certain mixture of high-volume bonds, low-volume bonds, and shares.

- The risk profile of the client can be translated directly into a corresponding mixture of shares of different risk categories ('speculative', 'medium', 'conservative').
- In addition, accordance with the client's risk profile can be ensured by distributing the investments of each portfolio over a number of countries and branches.

As far as the immense complexity posed by the markets is concerned, the portfolio management agency in question has drastically reduced the number of potential investment objects to be taken into account on a daily basis by maintaining a 'positive list' of approved shares. This list comprises some 200 shares which are deemed 'interesting' and 'attractive'.

In the daily routine of portfolio management, i.e., in monitoring and regulating the composition of each portfolio, the consultants only buy shares that have been approved and added to the list. Because the list — as a written text — is persistently accessible, a lot of time consuming and disruptive discourse between analysts and consultants is avoided. Though a very simple artifact, the list substantially reduces the overhead cost of conveying the recommendations of the analysts to the consultants.

However, taking investment decisions on behalf of a client is a delicate affair. A certain anxiety prevails. This issue is exacerbated when, as is the case in the particular portfolio management agency in question, clients typically are institutional investors and other large investors who often possess considerable insight of their own in the workings and behavior of the markets. The overriding concern of the consultants, therefore, is to be able to 'explain' and 'justify' their actions to their clients. Accordingly, a continuous exchange of views and background information between analysts and consultants is a prerequisite for the consultants' ability to instill confidence in their clients. For instance, the consultants need to be able to explain to clients how the markets in which they have invested behave. They must be able to tell whether there are causes for concern. They must be able to inform the client of his position. And they must be able to discuss alternative options. The situation in the markets is discussed regularly across the room. That is, a close contact between analysts and consultants is necessary.

This need for access to background information applies to the list as well. If the consultants simply had to execute a list handed down from analysts at the headquarters of the bank,[1] they would not be able to instill much confidence in their clients. The list is therefore managed cooperatively by the analysts and consultants in the sense that the analysts has to convince the consultants of the rationality of any change to the list whereas the consultants, reflecting the concerns of their clients, so to speak play stubborn and recalcitrant. The recommendations of the analysts are only accepted after intensive and thorough discussion. In fact, quite often the cooperative management of the this simple coordination mechanism will turn into protracted and heated arguments.

---

1 Which is what the headquarters of the bank planned to do at the time.

In sum, then:

(1) Day-to-day decisions concerning the control of portfolios are made on the basis of a list of ca. 200 'approved' shares. As an artifact, the list is quite unassuming. It does not provide any structure to the listed shares. The list merely records and conveys the fact that the listed shares have been agreed upon as 'interesting' securities.

(2) The list mediates and coordinates cooperative work between specialists in the sense that the daily portfolio control work of the consultants can be carried out without ongoing discourse with the analysts.

(3) However, the management of the list itself is a cooperative activity. Since the consultants need to be able to 'explain' their actions to their relatively sophisticated clients, approval of the composition of the list requires — sometimes heated — debates between analysts and consultants. In other words, the cooperative management of the list involves negotiation.

(4) Continuous exchange of views and background information between analysts and consultants is a prerequisite for the consultants' ability to instill confidence in their clients.

(5) That is, given the fact that the clients of the portfolio management agency in question are relatively sophisticated, the division of labour between analysts and consultants is feasible if and only if the consultants have access to general background information about the state of affairs in the markets and the rationale for selecting particular securities as investment objects for certain clients.

(6) The discretionary nature of investment decision making combined with the general anxiety of managing other people's fortunes and the exacerbated stress of serving relatively sophisticated and demanding clients all conspire to require intensive daily contact and, intermittently, spontaneous debates and negotiations between analysts and consultants.

1.3.8. Classification scheme: The ICD

Like the list in the portfolio management agency, the International Classification of Diseases (ICD) is an example of a list that mediates and coordinates cooperative work between specialists (Bowker and Star, 1991). As opposed to the unassuming list of shares, it is not — as an artifact — simple nor is its cooperative management merely a matter of deciding whether a certain entity should be included or not. Rather, the ICD is a classification scheme that has been evolved over very long time, it is being developed by a large number of people with incongruent objectives and perspectives, and it serves to coordinate a vast number of widely distributed activities in quite different circumstances.

The ICD is a list of causes of death and diseases. It is about one hundred years old, and has been revised nearly every ten years since the 19th century. The ICD

is typically distributed as a book to public health offices, hospitals, and bureaus of vital statistics throughout the world. It contains numbers which correspond to causes of death or illness, and algorithms for arriving at those numbers in complex cases involving more than one disease or cause.

> "One of the values of a list like the ICD is that it can be used in trans-national comparisons. This is useful epidemiologically, in that it enables one to trace specific environmental and nutritional factors that might be involved in the occurrence or spread of particular diseases." (Bowker and Star, 1991, p. 75)

However, these advantages can only be fully exploited if the various states agree on the way information is collected and coded. Different states submit their data more or less promptly; they have different administrative structures with different units of aggregation; and they have different policies concerning death certificates that have made an appreciable difference to the results of the ICD.

In addition, different categories of users of the ICD (such as doctors, epidemiologists, and statisticians) have different needs and pose conflicting demands on the design of the ICD. And, finally, similar conflicting demands are posed by various industrial actors: insurance companies, industrial firms, and pharmaceutical companies.

The point is, in the words of Bowker and Star, that "the list cannot be homogeneous, neutral and appeal to all parties" (Bowker and Star, 1991, p. 77).

As established by Bowker and Star, the design and use of the ICD has been adapted to these requirements in many ways, most importantly by means of the following features:

Residual categories. These categories can indicate uncertainty at the level of data collection or interpretation. Forcing a more precise designation could give a false impression of positive data. The major disadvantage is that a lazy or rushed doctor will be tempted to overuse 'other'.

Heterogeneous Lists. Throughout the history of the ICD, there has been a great deal of debate about whether it constituted a nomenclature or a classification. The difference is that a nomenclature is merely a list which does not give any indication of cause whereas a classification gives causes. The advantage of a nomenclature is that it can remain more stable over time whereas classifications are more convenient immediately, but change frequently. In the words of Bowker and Star, "the solution that has emerged over time has been rather to find the appropriate level of ambiguity — to keep the list as heterogeneous as possible for the different actors to find their own concerns represented." (Bowker and Star, 1991, p. 77)

Parallel Different Lists. Different user groups have found that the ICD did not serve their purposes, and so they have modified it. This could for instance happen in a country with a different range of medical problems. Similarly, some specialists might be interested in a finer breakdown than that permitted by the ICD. The strategy of the ICD committee in these cases has been to issue rules for how the list is to be modified.

Bowker and Star draw four lessons from their study of the ICD that are worth quoting:

> "- first, there is a permanent tension between attempts at universal standardization of lists, and the local circumstances of those using them;
>
> - second, this tension should not, and cannot, be resolved by imposed standardization, because the problem recurses;
>
> - third, rather, from the point of view of coordination, ad hoc responses to standardized lists can be mined for their rich information about the heterogeneous knowledge domain […];
>
> - fourth, making this sort of list is an example of the creation of the sort of object which must satisfy members of worlds or organizations with conflicting requirements. In its creation, and later in its use, the complex list is a kind of knowledge representation particularly useful for coordinating distributed work, which often contains requirements of this sort. Some, ourselves among them, would argue necessarily conflicting." (Bowker and Star, 1991, p. 74)

In sum, then:

(1) The collection, organization, and use of data pertaining to causes of death at a global scale is a cooperative effort on a vast scale and involving multitude of interested parties. The whole enterprise hinges on a globally accepted, standardized classification scheme.

(2) The adoption of a standard format does not alleviate the conflicting demands and needs. The problem recurses. The list is subjected to cooperative management — by the committee as well as by the data providers and users.

(3) The design and management of the list adapts to the conflicting demands by carefully finding "the appropriate level of ambiguity".

1.3.9. Administrative procedures

Suchman gives a detailed account of the role of administrative procedures in an accounting office. The accounting office studied by Suchman (Suchman, 1983) is responsible for the orderly payment of money due to outside organizations supplying goods and services to the organizational units in its charge. Orderly payment is documented through the office's record-keeping, and accuracy is monitored by the auditing of invoices against records of requisition and receipt.

According to Suchman, the "smooth flow" of paper on a given purchase the following sequence "occurs":[1]

(1) The facility's procurement office issues a purchase order (P.O.). Three copies are distributed: one each to the supplier, the shipping/receiving department of the facility, and the Accounting Office.

(2) The Accounting Office copy is filed in a temporary file.

(3) As the items ordered arrive at the receiving department they are marked off on the receiving department's copy of the purchase order (the receiver), a copy of which is in turn sent to Accounting.

---

1 The description of the case does not indicate whether this sequence is explicitly stipulated in a written text. In most accounting offices, however, this would be the case.

(4) Invoices issued by the vendor arrive in the Accounting Office via the U.S. mail. On arrival they are matched with the waiting purchase order and receiver.

(5) With the purchase order, receiver, and invoice in hand, the audit of price, quantity, sales tax, account numbers, part numbers, and so forth, can be done.

(6) On completion of the audit, with no discrepancies encountered, the work necessary to a generation of payment begins.

(7) When the payment is issued, the invoice, purchase order, and receiver are attached behind a copy of the check and filed away in the paid file.

Suchman highlights one specific routine complication. It may be the case that the items on a given purchase order are received and billed in separate installments over an extended period of time. Again, if all goes smoothly, the items marked off on the receiving report from Shipping/Receiving correspond to those on the invoice from the vendor. The purchase order, receiver, and invoice are matched and audited. The payment for the items received is recorded by margin notes on the purchase order, which is then returned to the temporary file to wait for the next shipment and billing. Only after all bills have been received and paid is the completed purchase order filed permanently in the paid file.

According to Suchman, the data on this case are constituted principally by a lengthy session of collaborative work between the accounts payable auditing clerk, K, and the accounting supervisor, R. K's work on the case begins with the arrival of a past due invoice in the mail. As a claim of money owed by the facility, the arrival of any invoice from an outside supplier initiates action. As a claim of payment overdue, a past due invoice is a formalized notice of trouble.

If a past due invoice were taken at face value, payment could simply be issued without delay. But before making payment the Accounting Office must establish the legitimacy of the vendor's claim. A review of past actions taken on the order, as recorded in Accounting Office files, is the primary resource for that task. In this case, however, the record of what happened presents its own troubles.

Up to the point in the work where the following the transcript starts, a search of the files has produced the following discrepancies:

(1) The original purchase order is missing.

(2) A completed receiving document is found. There are eight items listed on it, all of which have been marked as received. But the two invoices found in the paid file show only items 3 and 8 as paid. There is no invoice or record of payment for items 1, 2, 4, 5, or 6 and 7, yet the vendor reports that the transaction will be completed with payment of the past due invoice for items 6 and 7.

(3) Two packing slips and a receiving document show items 1, 2, 4, and 5 received with item 8, but the invoice to which they are all attached shows item 8 only.

It appears, then, that there are in fact *six* items whose payments are due (1, 2, 4, 5, *and* 6 and 7.) At the same time, the vendor reports that the past due invoice (for items 6 and 7) is the final payment, and the receiving department reports all the items received. At L256-57 of Suchman's transcript, on the basis of their work together to this point, K and R agree that there must be, somewhere, another record of payment for items 1, 2, 4, and 5.

Sequence 1

L255	*R:*	There's another purchase order some-
L256		I mean there's another payment somewhere?
L257	*K:*	Yea, there's got to be another.
L258	*R:*	There's another payment somewhere,
L259		now where is that? Is the question.

R and K agree here that there is a missing record of payment on items 1, 2, 4, and 5; the location of that record becomes, accordingly, the question to be answered and the direction for a search.

Sequence 2

L260	*K:*	The only thing I can think of is that it's:
L261	*R:*	Where's their (paid) folder.
L262	*K:*	
		Let me go get the whole folder,
L263	*R:*	Why don't you.
L264	*K:*	and maybe if I: go through the control numbers *(inaudible)*
L265		*((she goes to paid invoice files))*
L266	*R:*	Yea, that's what we're gonna have to do
L267		we have to look at that whole folder.
L268	*K:*	*((returns with folder))* The only thing I really was goin' on
L269		was the P.O. number, cause I didn't have any invoice numbers,
L270		or really) any dates, to go-
L271		to find out when it would have been paid
L272	*R:*	*((looking through folder contents))*
L273		What purchase order are we dealing with?
L274	*K:*	36905.
L275		*((pause while R leafs through folder))*
L276	K:	What's weird is, though, the girl was telling me
L277		this number that comes after this, *((number on the past due invoice))*.
L278		That tells you this is the third invoice,
L279		for this like P.O.
L280		[
L281	*R:*	billed on that P.O.
L282	*K:*	But if that's true, this is one *((invoice for item 3))*, that's two *((invoice for item* 8)), and this is three: *((past due invoice for items 6 and 7))*
L283		Then there might not *be* another bill.
L284	*R:*	*(inaudible)* Is that *((the missing P.O.))* in that problem pile up there anywhere?
L285	*K:*	No, I don't recall seeing it, but I'll double check on it
L286		*((getting stack from upper shelf)*

At L272, the accounting supervisor R begins what proves to be an extensive search of the record of past payments to the vendor, while the auditing clerk K re-examines the set of documents already in hand. At L276-83, informed by her talk with the vendor, K pulls out one detail that seems somehow related to the question of the record's completeness. R's completion of K's remark at L281 demonstrates that he is listening, but at L284 R leaves K's comment without remark and continues with a new question which temporarily brings K away from the invoices and back to the missing purchase order.

Their work is proceeding along two more or less independent lines, with R searching the record of past payments while K continues to study the documents already pulled from the file, when K makes a discovery:

Sequence 3

L333	*K:*	Huuuh.
L334	*R:*	Hmm?
L335	*K:*	Look at ((*invoice for item 8*)) missing page 2.
L336	*R:*	Where do you get that at?
L337	*K:*	Page 2.
L338	*R:*	Page 2? It's a two page pur- two page, uh invoice?
L339	*K:*	Oh no, oh no. (You're not gonna like this.)
L340	*R:*	know I'm not gonna like it. I already don't like it,
L341	*K:*	Okay,
L342	*R:*	I'm having to look at it, that's making me not like it.

In spite of the split in their attention, which begins with R's search of the paid folder, K and R are each "continually producing comments that, as assessments of what each is finding, allow the other to monitor their joint work" (p. 325). In other words, in exactly the same way as the operators of the Bakerloo Line control room and the air traffic control operators, they are developing and maintaining a reciprocal awareness by making their individual activities publicly visible and at the same time monitoring each other's activities. K and R's continual monitoring of each other provides for their respective searches, at any point, to develop into concerted attention to any of a number of findings.

A little later, K and R turn back to K's finding of the missing page:

Sequence 4

L414	*K:*	Okay,
L415	*R:*	*Now* then tell me what you see there.
L416		Now I've got that in order *((the paid file)),* then we don't have to look.
L417	*K:*	This is page two *((invoice for item 8)),*
L418	*R:*	Mmhrn.
L419	*K:*	Okay. We got three of these items *((8))* for $156,
L420		but all of the tax on them does not equal $117,
L421		so the page one items: *((items 1, 2, 4, and 5))* go with this invoice *((for item 8)).*
L422		That's why *((the vendor))* says this *((invoice for 6 and 7)) is* the last item.
L423	*R:*	But you don't have page one.
L424	*K:*	No. *((pause))* Page one isn't there.

L425	*R:*	Thi-this one *((for item 8)) is* already paid?
L426	*K:*	Yea, this one's paid.
L427	*R:*	And that's the check for it?
L428	*K:*	Yea, that- these two: packing slips *((for items 1, 2, 4, 5, and 8))* were attached to the receiver.
L429		So that was- according to them, we've paid the full amount *((for items 1, 2, 4, 5, and 8)),*
L430	*R:*	We've paid the full amount,
L431	*K:*	but we don't: *((laugh))*
L432		[
L433	*R:*	But we don't know where page one is.
L434	*K:*	Cuz this, times tax, just don't: equal up.
L435	*R:*	Mm hm, mm hm.

At L428-9, K shows how the missing invoice page works to explain why there are two packing slips with the invoice for item 8, one being for items 1, 2, 4, and 5, and R agrees at L430. R immediately offers a next action:

Sequence 5

L435	*R:*	Mm hm, mm hm. I want you to call that lady
L436		and tell her you want page one. Of this invoice.
L437	*K:*	Now at least we have a number to go on,
L438		[
L439	*R:*	Tell her that you're gonna do her something *((pay for items 6 and 7))-*
L440		we're gonna do her something, we want her to do us something *((provide the missing invoice page for items 1, 2, 4, and 5)).*
L441		We need page one for this invoice. All right? And then that-
L442		that explains why all those other things ((1,2,4 *and 5))* are not there.
L443	*K:*	Okay.

Suchman's interpretation of the case is thought-provoking and somewhat contentious:

> "Standard procedure is constituted by the generation of orderly records. This does not necessarily mean," Suchman posits, "that orderly records are the result, or outcome, of some prescribed sequence of steps. Workers in the Accounting Office are concerned that (1) money due should be paid, and (2) that the record should make available both the warrant for payment and the orderly process by which it was made. In this case, once the legitimate history of the past due invoice is established, payment is made by acting as though the record were complete and then filling in the documentation where necessary. The practice of completing a record or pieces of it after the fact of actions taken is central to the work of record-keeping." (p. 326).

To be sure, the case shows convincingly that orderly records are not necessarily the result of some prescribed sequence of steps. But the case analyzed by Suchman is a case of *recovery from error* in an administrative agency and provides little, if any, insight into how standard procedures, defined as pre-defined written stipulations, are applied in routine daily work.

It is therefore difficult, if not impossible, to conclude that orderly records are *not* — in normal circumstances where records are complete — the result of some prescribed sequence of steps:

> "It is the assembly of orderly records out of the practical contingencies of actual cases that produces evidence of action in accordance with routine procedure. This is not to say that workers 'fake' the appearance of orderliness in the records. Rather, it is the orderliness that they construct in the record that constitutes accountability to the office procedures." (Suchman, 1983, p. 327)

Suchman's analysis is demonstrably true as far as the error recovery case is concerned, but there is no empirical basis for the quite general claim: "It is the assembly of orderly records out of the practical contingencies of actual cases that produces evidence of action in accordance with routine procedure."

Anyway, precisely because it is a case of recovery from error, the case also provides a graphic impression of the massive heuristic use of standard procedures even in a seemingly abnormal situation. The two actors solves the abnormal problem because of their "knowledge of the accounts payable procedure" (p. 322).

In sum, then:

(1) Standard procedures have a heuristic function in the sense that they "are formulated in the interest of what things should come to, and not necessarily how they should arrive there." (p. 326)

"The operational significance of a given procedure or policy is not self-evident, but is determined by workers with respect to the particulars of the case in hand. Their determinations are made through inquiries for which both the social and material make-up of the office setting serve as central resources. This view recommends an understanding of office work that attends to the judgmental practices embedded in the accomplishment of procedural tasks." (Suchman, 1983, p. 327).

(2) Suchman's case study describes a case of recovery from error in an administrative agency. As such, the case study does not tell us very much about the use of prescribed procedures in routine daily work. What the case does give us, however, is an insight into the crucial role of prescribed procedures even in handling contingencies. The case shows that prescribed procedures convey important heuristic information for the handling of routine tasks as well as errors.

1.4. Articulation of cooperative work: Modes and mechanisms

The discussion of the different cases of cooperative work in the preceding chapter allows us to outline some salient features of cooperative work and its articulation.

In cooperative work, multiple persons cooperate in the sense that they are interdependent in their work or in other words work 'on' a common field of work,

the part of the world that is affected (controlled or transformed) by the work of the actors, e.g.:

- the hot rolling mill as controlled manually by a small number of operators;
- the nuclear power plant and its control system as supervised by the operators in the control room;
- the air traffic in a particular sector of airspace and the concomitant monitoring and communication system at the disposal of the controllers in charge of that sector;
- the manufacturing plant and its myriad parts and processes, orders and workstations;
- the collection of portfolios to be protected and nurtured by the consultants and analysts in a turbulent and complex securities market;
- the global mass of vital statistics to be ordered by a vast and distributed ensemble of doctors and bureaucrats.

The distributed activities of the members of the cooperative ensemble are interdependent in the sense that they — in different ways — contribute to the overall process of control of the common field of work. The participants interact by controlling (monitoring, regulating, changing, transforming) the state of the field of work. Thus, for the participants to be able to contribute purposefully to the cooperative effort, each of them need access to information pertaining to the state of the field of work: *what is the situation, what has happened, what is happening, what might happen?*

Interacting by changing the state of the field of work is an extremely restricted and convoluted way of interaction:

(1) Changing the state of the field of work, for instance by changing it, is not a symbolic act. It is a real act. It is what it appears to be.

(2) The allowed changes to the field of work are determined by the nature of the field of work (e.g., the degree of coupling between objects and processes, the extent to which changes are reversible).

(3) The content of the interaction (the meaning conveyed from A to B) is restricted by the extent to which the changes of the state of the field of work are visible to the others and what they may indicate to the others.

(4) The turn-around time of the interaction may be determined by the frequency of state changes in the field of work.

That is, apart from rare cases like the hot rolling mill or a booking system where the field of work changes predictably and linearly along a few parameters in an un-ambiguous way, the common field of work in itself does not provide adequate means for articulating the different contributions of different individuals to the cooperative effort. The need for other means of articulating distributed activities is especially manifest if the state the field of work changes dynamically, unpredictably, non-linearly, if events are ambiguous, if the state of affairs can only be ascertained indirectly, or if disturbances due to deficient coordination of activ-

ities could have severe consequences (as in the case of a nuclear power plant or air traffic control).

Thus, while mediation of cooperative work via a common field of work is fundamental to all cooperative work, the articulation of cooperative work requires a multitude of modes and mechanisms of interaction.

1.4.1. Modes of interaction

As observed above, articulation work is a multi-facetted phenomenon in the sense that work is articulated with respect to multiple dimensions: with respect to the field of work (objects, processes, sensors, effectors, representations etc.) and with respect to actors, responsibilities, tasks, activities, conceptual structures, informational and material resources.

With respect to each of these dimensions, articulation work involves an open-ended repertoire of modes and means of interaction that are meshed fluently in innumerous ways. A few examples will illustrate the point:

Maintaining reciprocal awareness: The articulation of the distributed activities in a cooperative setting normally requires the continuous formation among the members of the cooperating ensemble of a reciprocal awareness of the activities, concerns, and intentions of the other members of the ensemble. Or in the words of Heath and Luff: “The ability to coordinate activities, and the process of interpretation and perception it entails, inevitably relies upon a social organisation; a body of skills and practices which allows different personnel to recognise what each other is doing and thereby produce appropriate conduct.” (Heath and Luff, 1992, pp. 70 f.)

The development and maintenance of reciprocal awareness of the work of the other members of the ensemble may involve an ongoing process of inconspicuous, unobtrusive, and even “surreptitious” (Heath and Luff, 1992, p. 74) monitoring of the activities of the others, by seeing and hearing what the others are doing, where they are in the room; by noticing the level of letters in the in-box or the lack of certain parts in racks containing the buffer stock and so on.

The formation of reciprocal awareness through monitoring of the activities of the others is matched by an ‘inverse’ — and often equally inconspicuous and unobtrusive — effort of rendering activities visible to the others: modulating operations on the field of work, humming, thinking aloud, leaving traces, recording, logging, reporting, etc.

Directing attention: In articulating their joint effort, each of the members of the cooperative ensemble may deliberately — but not necessarily consciously — direct the attention of the other members to certain features of the state of the field of work, a possible problem, disturbance or danger, etc. by invoking a multitude of modes of interaction:

- by embedding cues, e.g., by marking particular items, for instance by positioning them in certain locations and ways, by highlighting them etc.

- by tacitly modulating work activities in uncommon, unusual, or abnormal ways;
- by gingerly humming, drumming, coughing, gazing;
- by overtly pointing, nodding etc. at particular objects;
- by warning explicitly (talking, shouting, annotating, writing).

Assigning tasks: As pointed out by Strauss, a wide variety of social modes of task allocation can be observed:

> "tasks can be imposed; they can be requested; also they can just be assumed without request or command; but they can also be delegated or proffered, and accepted or rejected. Often they are negotiated. And of course actors can manipulate openly or covertly to get tasks, or even have entire kinds of work allocated to themselves." (Strauss, 1985, p. 6.)

Moreover, tasks can be requested in countless ways: by nodding towards a thing, by highlighting an object of work, for instance by it at a certain spot (in the in-box, on the desk, pigeon hole), by explicit verbal request (post it note, office memo, executive command).

Handing over: In the course of a cooperative effort, responsibility for a certain process in the field of work may be handed over from one actor to another. Again, this can be done in different ways: the object itself may be handed over (e.g., the strip being on between operators of the hot rolling mill, finished parts being passed on to the next station in manufacturing), the interface to the control system may be handed over (e.g., the wheel on the bridge of a ship), a symbolic representation may be passed on as a semaphore and so on.

The point to be made by way of these examples, is that the myriad modes of interaction involved in articulation work cannot be ordered according to any simple conceptual scheme. Instead, a limited number of salient features of modes of interaction can be highlighted:

(1) Unobtrusive versus obtrusive: Modes of interaction can be more or less obtrusive: some modes of interaction such as pointing or tapping at an item or talking or shouting to colleagues are be highly intrusive in that they impose an obligation on the others to notice and react accordingly (more or less instantly). They therefore disrupt current activities (which may or may not be). Other modes of interaction can be quite inconspicuous, such as, for example, embedding cues, humming, gazing, thinking aloud, leaving traces.

(2) Embedded versus symbolic: Embedding cues by highlighting particular items belonging to the field of work or representing the field of work, for instance by positioning them in conspicuous ways, at unusual locations or in abnormal orientations, by marking them etc., has significant advantages in that it (1) uses items that are ready-at-hand, perhaps ubiquitous, and that are constantly monitored due to their status as belonging to the field of work and (2) therefore is more efficient and less intrusive and distracting than, for instance, pointing or talking or other modes of interaction that impose the role of a recipient on somebody.

Embedding cues in objects, for example by marking a certain feature in the field of work so as to convey to others that they should pay attention to a particular occurrence or take a particular action, is not, strictly speaking, a symbolic act. We are here following Peirce's distinctions:

> "I respect to their relations to their dynamic objects, I divide signs into Icons, Indices, and Symbols […]. I define an Icon as a sign which is determined by its dynamic object by virtue of its own internal nature. […] I define an Index as a sign determined by its dynamic object by virtue of being in a real relation to it. […] I define a Symbol as a sign which is determined by its dynamic object only in the sense that it will be so interpreted." (Peirce, 1901)

Thus, an artifact 'determined' by the field of work can be conceived of as having the function of an index: objects belonging to the field of work, means of data acquisition (e.g., sensors), representations that reflect the state of the field of work automatically (e.g., gauges, radar) or are made to represent the state of the field of work (e.g., flight strips).

The primary function of, for example, the flight strips is that of a representation of the state of the field of work, that is, of the current configuration of airplanes in the sector. The strip does not have the abstract nature that provides the degrees of freedom in its manipulation that otherwise makes *symbolic* representations so powerful. An individual flight strip should rather be seen as a *index* of a particular airplane — not in the sense that moving the strip will make the airplane move, of course — but in the sense that air traffic control relies predominantly and crucially on *the precise mapping* of the state of the airways onto the state of the strips so that any manipulation of the strip is subordinate to the objective of ensuring this mapping. That is, the repertoire of allowed operations on the flight strip is strictly limited by this primary function.

Any modulation of the way in which a strip is manipulated is therefore a sign embedded in the appearance of an artifact standing proxy for the state of the field of work. That is, the message is cloaked.

Interacting by manipulating some object or system belonging to or in an indexical relation to the field of work is a restricted way of interacting:

(a) The bandwith of embedded cues is limited to the degree of freedom offered by the role of the object in the field of work;

(b) the turn-around time of embedded cues may be limited by the frequency of state changes in the field of work;

(c) the message is garbled in that it is shrouded in the state of an object belonging to the field of work or representing certain features of the field of work.

On the other hand, the cases of the MRP II system and the *kanban* system shows the crucial importance, in some settings, of severing the (direct or automatic) coupling between the field of work and artifacts — namely when artifacts are used to stipulate and mediate the articulation of cooperative work. That is, such artifacts have to have a symbolic status, as opposed to an indexical.

(3) Ephemeral versus persistent: A wide range of modes of interaction are ephemeral in the sense that articulation work in these modes only exist in the flux of unfolding activities. For example,

- monitoring the activities of others, by seeing and hearing what the others are doing, where they are in the room; by noticing the level of letters in the in-box or the lack of certain parts in racks containing the buffer stock and so on;
- making one's own activities publicly visible by modulating operations on the field of work, humming, thinking aloud;
- directing attention by modulating work activities in uncommon, unusual, or abnormal ways, by humming, drumming, coughing, gazing, pointing, nodding, talking, shouting.
- allocating tasks by pointing, nodding, talking, shouting.

As soon as the articulation activities have been carried out and a new situation has arisen, the articulation that was achieved vanishes without trace, as it were — like the snows of yesteryear.

In important ways, the same applies to modes of interaction that involve embedding cues. While certainly based on the use of artifacts, embedding cues depend on the fate of the items conveying the cues in the ever-changing field of work. The embedded cues may be erased by state changes, or they may not. As vehicles of embedded cues, the highlighted objects live an uncertain life.

Because of the immediate feedback and the ensuing possibilities of detecting and recovering from misunderstanding, combined with expressive power provided by the vast repertoire of modes that can be combined at any time, these modes of articulation offer immense flexibility in terms of articulating activities in face of the mundane and dramatic contingencies of cooperative work.[1]

These modes of interaction are especially crucial in cooperative work settings where articulation work is time-critical (as in the case of process control or air traffic control). Articulating cooperative activities in such settings typically requires a permanently open channel of communication with minimal turnaround time, for example by having the operators in the same room at the same time, so as to convey the multitude of inconspicuous cues that are required for cooperators to acquire and maintain reciprocal and general awareness of the changing state of affairs within the cooperating ensemble, as well as the field of work at large. Likewise, articulation of distributed activities that involve discretionary decision making — as in the case of portfolio management — will typically require, at least intermittently, various negotiation processes. For this purpose, conventional co-located 'face-to-face' interactions provide the required large bandwith, not only in terms of gigabits per second but also, and more importantly, in terms of a rich variety of interactional modes with powerful and flexible social connotations.

---

1 "In oral face-to-face settings, abundant non verbal cues and a common physical environment help establish a referential framework not usually available for written communication." (Goody, 1987, p. 268 — quoting Reder).

On the other hand, however, these ephemeral modes of interaction do not provide strong support for making decisions and commitments concerning the articulation of cooperative work accessible to the members of the cooperating ensemble, independently of the situation, and independently of particular individuals or for supporting the development and implementation of stipulations for the ways in which in which cooperative work is to be conducted and articulated.

Written records (log books, recordings, minutes, memos etc.) provide persistence to decisions and commitments made in the course of articulation work: "The written language [reaches] back in time" (Goody, 1987, p 280). Written records are, in principle, accessible to any member of the ensemble, whatever its size and distribution in time and space. "Written systems can provide a larger number of people with the same information at one time." "Written messages are portable, allowing interaction without spatial constraints." On the other hand, "Written systems are much less dependent on physical arrangements" and "less time-dependent than oral systems." (Stinchcombe, 1974, pp. 50 f.).

As illustrated by the case of the list of approved shares in portfolio management, written artifacts can at any time be mobilized as a referential for clarifying ambiguities and settling disputes: "while interpretations vary, the word itself remains as it always was. (Though every reading is different, it is a misleading exaggeration of the literary critic to say that the text exists only in communication.)" (Goody, 1986, p. 6). However, and this is also illustrated by the portfolio management case, "written language is partly cut off from the context that face-to-face communication gives to speech, a context that uses multiple channels, not only the purely linguistic one, and which is therefore more contextualized, less abstract, less formal, in content as in form." (Goody, 1987, p. 287).

Written formulations encourage the decontextualization or generalization of stipulations of orderly cooperative work. In their very nature, written stipulations have been abstracted from particular situations in order to be addressed to the target audience in general, rather than delivered face-to-face to a specific group of people at a particular time and place (Goody, 1986, pp. 12 f.).

(4) Degree of local control: In terms of *stipulation* of articulation work, there is a clear ladder from, at the one end of the spectrum, modes of interaction that do not involve any pre-specified stipulations — to modes of interaction that involve the prescription incorporated in and the active mediation of pre-specified artifacts, at the other end of the spectrum:

(a) Ad hoc articulation (by means of monitoring others, directing attention, embedding cues in artifacts, and negotiating). This mode of articulation work offers a high degree of local control and hence very powerful means of recovery from misunderstanding and error and of handling contingencies. On the other hand, ad hoc articulation is highly inefficient when faced with recurring problems, and it may be difficult to anticipate the course of the cooperative effort and hold actors accountable.

(b) Articulation by means of conventions, i.e., the usual and expected way to do things. Whether the conventions are made explicit or merely observed tacitly, this mode wholly relies on mutual understanding and the will to adhere to the expectations, supported by various forms of social sanction. That given, it may still be difficult to interpret an action, anticipate the course of the cooperative effort and hold actors accountable.

(c) Articulation by means of stipulations supported by artifacts in the form of written statutes, e.g., standard operating procedures or accounting prescriptions. As opposed to conventions, stipulations supported by artifacts in the form of written statutes are, in principle, accessible by all at any time. That is, the fact that the stipulations are supported by symbolic artifacts makes everybody accountable in a far higher degree. The execution of the stipulation, however, relies completely on human recollection and the stipulation is quite open to interpretation (Hart, 1961; Goody, 1986).

(d) Articulation by means of stipulations supported by passively mediating symbolic artifacts, e.g., forms, organizational charts, thesauri, which by imposing a standardized format restricts the 'degrees of freedom' each user is faced with. In these cases, the execution of the stipulation no longer relies completely on human recollection and the range of (reasonable) interpretation is relatively narrow compared to, say, statutes.

Goody's discussion of lists, tables, matrices and other spatio-graphic devices for organizing linguistic items abstracted from the context of the sentence (e.g., in thesauri) is illustrative of this point:

> "We find, for example, a large number of lexical lists, of trees, roles, classes of various kinds, which possess several characteristics that make them differ from the categories that usually emerge in oral communication. First, they consist of isolated lexemes abstracted from the flow of speech, and indeed from almost any 'context of action' except that of writing itself. Secondly, they are formalized versions of classificatory systems that are to some degree implicit in language use but go beyond those classificatory systems in important ways. In particular they take category items out of the sentence structure, and group them by similarities, sometimes even providing them with unpronounced […] class indicators. Thus the categories are given a formal shape, a specific beginning and a definite end, into which each item has to fit […]. Moreover the boxes tend to be exclusive. Fruits end here; vegetables begin there; the tomato has to be placed in one box or table rather than another, setting aside […] the flexibility of oral usage which has greater toleration of ambiguity and anomaly, a greater contextualization." (Goody, 1987, p. 275).

(e) Articulation by means of stipulations supported by *actively* mediating symbolic artifacts, e.g., time tables, indexing systems for repositories, the ICD, kanban-systems, computer-based scheduling systems (e.g., MRP systems), workflow management systems. Of course, using such a symbolic artifact always requires certain social conventions and skills but in highly complex cooperative work settings these conventions and skills must be supplemented and supported by a mechanism in the form of an artifact, at least to the extent that it provides a plan for action that can be carried out without further consultation and a verifiable (interpersonal and situation-independent) basis for holding actors accountable.

1.4.2. Mechanisms of interaction

As shown in the preceding discussion of modes of interaction, in much of everyday working life, the required articulation of individual activities is managed effectively and efficiently by the rich variety of intuitive interactional modalities of everyday social life, so effectively and efficiently in fact that the distributed nature of cooperative work is not apparent, most of the time. People tacitly monitor each other; they make their activities sufficiently apperceptible for others; they take each others' past, present and prospective activities into account in planning and conducting their own work; they gesture, talk, write to each other, and so on. Accordingly, much of the research in CSCW has focused on providing enhanced means of communication, either in order to enable actors to cooperate more effectively and efficiently in spite of geographical distance, or in order to widen the repertoire of communication facilities.

However, in the complex work environments of modern industrial and administrative organizations, the problems of articulating distributed activities are at a different order of complexity. The everyday social and communication skills are far from sufficient in articulating the cooperative efforts of hundreds or thousands of actors engaged in myriads of complexly interdependent activities, perhaps concurrently, intermittently, or indefinitely.

In such settings, the articulation of the distributed activities of cooperative work requires a certain mode of interaction based on a class of symbolic artifacts that *stipulate and mediate* articulation work and thereby reduce the complexity of articulation work. We call these artifacts 'mechanisms of interaction'.

In order to serve this function, such a device must have the following characteristics:

1. **Artifact**: It must be publicly available in the sense that it does not solely reside 'in the head' of human actors (such as conventions) and it must be persistent in the sense that it is available independently of any particular situation. That is, it must exist in the form an artifact.
2. **Symbolic**: It must be possible to manipulate the mechanism independently of the state of the field of work. The artifact incorporating the mechanism of interaction must not be coupled in any strong, tight or irreversible way to the state of field of work. Appropriate manipulations of the device for articulation purposes should be feasible without unwanted side-effects on the field of work. That is, the artifact must have the character of a symbolic artifact.
3. **Standardized format**: It must provide affordances to and impose constraints on work articulation and it must make the state of work articulation at any given moment publicly perceptible.

A mechanism of interaction can thus be defined as a symbolic artifact that serves to reduce the complexity and cost of articulating the distributed activities of a cooperative work arrangement by *stipulating and mediating* the articulation of the distributed activities.

As a devise for the articulation of cooperative work, a mechanism of interaction should be distinguished from other symbolic artifacts regulating human affairs (e.g., signs such as “Stop!”, “Dead Slow!”, “Pharmacy”, “Wine & Spirits”, “Underground”, “Bus Stop”). A traffic light regulating the traffic at an intersection, for example, has certain facilities in common with a mechanism of interaction in cooperative work: It is a symbolic artifact that actively stipulates and mediates interaction. However, the drivers — as drivers — are not engaged in cooperative work; they are, rather, competing for the same resource (the road) and to each and every one of them the others are a mere nuisance. Their interaction is transient and superficial and does not entail the rich multiplicity of purposive reciprocal interactions of cooperative work.

Also, again for the sake of clarification, a mechanism of interaction should be distinguished from the information objects that are ubiquitous in cooperative work settings (such as letters, memoranda, drawings and reports, and the aggregation of such documents in the form of files and libraries) and that help to articulate and mediate distributed activities. Written documents are certainly symbolic artifacts but in so far as they serve to convey information and hence merely provide a medium of communication, they are not mechanisms of interaction. However, managing a complex flow of information objects may require mechanisms of interaction such as standardized formats (e.g., prescribed forms and routing instructions on file covers) and classification schemes (e.g., library catalogues and thesauri).

These artifacts can be conceived of as *mechanisms* in the sense that they (1) are objectified in some way (explicitly stated, artifactual), and (2) are deterministic or at least give reasonably predictable results when applied properly. And they are mechanisms *of interaction* in the sense that they reduce the complexity of articulating cooperative work.

As observed above, modes of interaction in articulation work differ with respect to the degree of local control. With a high degree of local control, the different modes of interaction do not impose domain-specific affordances and constraints on the conduct of articulation work. The affordances and constraints of these modes of interaction reflect the ones of the different media: bandwith, turn-around time, etc. On the other hand, when articulation work is stipulated and mediated by artifacts, the structure of the artifact inexorably embodies domain-specific affordances and constraints. Thus, in order to be able to support the articulation of distributed activities, a mechanism of interaction must represent certain pertinent aspects of the field of work and the organization of the ensemble. Conceptual schemes, for example, are obviously domain specific in that they represent a model of the conceptual structure of a domain. The same obviously applies to organizational structures. Also, the master schedule of a MRP system for production control in manufacturing embodies an elaborate model of the products, their components (Bill of Materials), the average production or purchasing time and cost of each component or subassembly, etc. Likewise, an organizational procedure refers to typical classes of cases characteristic of the

particular domain at hand. It may codify 'good practice,' recipes, proven methods, efficient ways of doing things, work routines. It may also convey information on the functional requirements to be met by the process and the product; it may highlight decisional criteria of crucial import; it may suggest a strategy for dealing with a specific type of problems (e.g., which questions to address first?); it may indicate pitfalls to avoid; or it may simply provide an *aide memoir* (such as a start procedure for a power plant or an airplane). And, finally, it may express some statutory constraints in which case disregard of the procedure may evoke severe organizational sanctions. In short, mechanisms of interaction like the ones mentioned above are enmeshed in the semantics of the particular work domains.

Mechanisms of interaction are local and temporary closures with a limited area of validity and they are by necessity underspecified. As observed by Gerson and Star, these mechanisms themselves require articulation work:

> "*Every real world system* […] *requires articulation* to deal with the unanticipated contingencies that arise. Articulation resolves these inconsistencies by packaging a compromise that 'gets the job done,' that is, that *closes the system locally and temporarily* so that work can go on." (Gerson and Star, 1986, p. 266)

Thus, mechanisms of interaction are not executable code but rather heuristic and vague devices to be interpreted and instantiated, maybe even by means of intelligent improvisation. Mechanisms of interaction are not automata but "resources": "plans are resources for situated action" (Suchman, 1987, p. 52). This observation applies to mechanisms of interaction in general.

To be made to work, they themselves need to be managed, i.e., constructed, maintained, developed, interpreted, applied, adapted, circumvented, modified, executed, represented, and negotiated. This secondary level of articulation work is, of course, also performed cooperatively (Schmidt, 1991).

At this point, however, some remarks of caution are required. The thesis that "plans are resources for situated action" does not, as yet, provide sufficiently explored conceptual and empirical foundations for designing mechanisms of interaction for CSCW systems. As argued above, the thesis is of fundamental importance to CSCW systems design, but for CSCW design purposes we need to investigate the relationship between plan and situated action in far more detail and far more precisely. In particular, we need to understand how pre-specified 'plans' incorporated in designed artifacts (schedules, procedures, classification schemes, etc.) may support practice.

Plans do not determine the course of action in any absolute or causal sense but different plans in different settings may determine the course of action differently, and in some settings plans do determine the course of action in a very 'strong' sense (e.g., timetables). These are issues that can — and should — be determined empirically (as opposed to philosophically). That is, for sociological research to provide a conceptual foundation for CSCW systems design a shift in perspective is necessary: What is it that makes plans, schedules, procedures, classification schemes etc. useful in the first place? What makes them 'resources'? Is it merely the fact that plans are underspecified in comparison with the rich multiplicity of

actual action that makes them "resources"? Is that really all? Is it indeed "*precisely* […] the fact that they *do not* represent those practices and circumstances in all of their concrete detail" that makes plans efficient and effective? What, then, makes one procedure more useful than another for a certain purpose in a specific setting? Which specific features in the designs of existing plans, schedules, procedures, classification schemes etc. make them amenable to their cooperative management and which features represents impediments to their cooperative management? Could a computer implementation of a specific mechanism of interaction enhance the ability of the given cooperative ensemble to articulate its distributed activities in a more flexible, effective, and efficient manner?

These are all researchable questions. That is, far from closing the book, so to speak, the concept of mechanisms of interaction for CSCW systems design opens up a host of intriguing and important problems.

1.5. Implications for CSCW systems design

Now, what are the implications for CSCW systems design? CSCW systems have generally failed to meet the requirements of users in actual cooperative work settings, primarily due to constraints imposed by current platform designs. It is becoming increasingly clear that current platforms in important ways are inadequate as platforms for CSCW systems. They are deficient for CSCW purposes in that they do not adequately support the seamless interweaving of individual and cooperative activities; a vast repertoire of alternative modes of interaction; the fluent and dynamic meshing of the available repertoire of modes of interaction; or the deeply material situatedness of articulation work, i.e., the fact that cooperative work is inextricably articulated with reference to the state of the field of work.

(1) Cooperative and individual activities are inextricably interwoven in daily work practice.

First, the boundary between individual and cooperative work is dynamic in the sense that people enter into cooperative work relations and leave them according to the requirements of the current situation and the technical and human resources at hand. That is, cooperative work arrangements emerge contingently, to dissolve again into individual work.

Second, in cooperative work settings, cooperative activities are punctuated by individual activities and vice versa. People shift between individual and cooperative activities and, while engaged in cooperative activities, they may be simultaneously involved in parallel streams of activity conducted individually.

Third, cooperative work is always conducted by individuals, and conversely, in cooperative work settings individual activities are always penetrated and saturated by cooperative work (Hughes et al., 1991; Heath and Luff, 1992).(Heath et al.,

1993). An activity carried out individually may be — or may any time become — part of a wider, loosely coupled cooperative activity.

A CSCW system should thus support the fluent meshing of individual work and cooperative work. In all its generality, this statement may seem uncontroversial. Nevertheless, most of the existing CSCW software products do not support this fluency. For example, when composing an email message the user should not be required to shift to a special editor and leave the word processor normally used for composing letters, writing reports etc. The same applies to CSCW facilities supporting cooperative authoring, conferencing, etc. The commercial groupware product ASPECTS, for example, allows multiple users to cooperate on writing a document. However, they are required to leave their single-user word processor and shift to the word processing facility of ASPECTS in order to cooperate. The effect of this that the system creates an impedance between cooperative and individual activities.

Since the *means of communication* required by the modes and mechanisms of interaction are semantically neutral in the sense that they can be applied (with different scope) in articulation work in all work domains, we will argue that these means of communication *should be conceived of as functions of the platform*. That is, CSCW facilities that support cooperative work by supporting various modes of interaction by increasing the bandwith of the communication channel or by reducing the turnaround time should not be conceived of as applications or be implemented as part and parcel of applications but as *platform functions* accessible to the appropriate applications (and, in the case of, say, desk top video conferences, to actors directly). If they are not conceived of *and implemented* as general system functions that can be accessed from and combined with applications, the delicate and dynamic relationship between cooperative and individual work breaks down. This applies to single-user as well as multi-user applications.

(2) While it is unlikely that the infinitely rich variety of modes of interaction in the articulation of cooperative work can be replicated in CSCW systems, the above analysis of modes of interaction indicates that certain facilities are required of CSCW systems for actors to be able to articulate their distributed activities in relation to computer systems in a sufficiently fluid way.

First, it seems necessary for actors to be able to control the articulation of their cooperative activities in terms of parameters such as different kinds and degrees of *obtrusiveness* and different kinds and degrees of *persistence*.

The control of articulation work in terms of such parameters might be conceived of in the same way as access control (in shared object servers) and floor control (in shared view systems). That is, the different policies of obtrusiveness should be user-selectable (Rodden and Blair, 1991). In addition, since it is unlikely that a finite set of policies can be identified, the policy ‘control panels’ should be open to respecification and addition — in much the same way as suggested by Greenberg with respect to turntaking protocols (Greenberg, 1991).

Second, since embedding cues in artifacts that are part of the field of work — and hence in items that are ready-at-hand, perhaps ubiquitous, and constantly monitored — plays a crucial role in the articulation of cooperative activities, actors should, in principle, be able to highlight any object in the computer environment in multiple ways, with different degrees of obtrusiveness and persistence. Again, this implies that actors should be able to control the way in which an object is highlighted, how the highlighting is propagated within the cooperative ensemble, who has access to changing the status of the highlighting, and so on. Since facilities that could meet these requirement will encroach upon what has heretofore been thought of as single-user applications, such facilities will have radical implications for the design of the operating system of a CSCW platform.

(3) A vast — presumably open-ended — array of modes of interaction is involved in the articulation of cooperative work. These different modes are combined and meshed dynamically, according to the requirements of the specific situation at hand, and they are meshed fluently and, more often that not, effortlessly. A CSCW system should support the fluent interweaving and combination of modes of interaction.

In sum, in order to meet these very general requirements — support the fluent meshing of individual and cooperative activities as well as the multitude of modes of interaction — the allocation of function between general platform facilities and specific applications should be planned and designed carefully.

(4) Since mechanisms of interaction are enmeshed in the semantics of the particular work domains, a mechanism of interaction should be conceived of as an abstract device incorporated in a software application (e.g., a CASE tool, an office information system, a CAD system, a production control system, etc.) so as to support the articulation of the distributed activities of multiple actors with respect to that application.

The purpose of the concept of mechanisms of interaction is thus to facilitate the design of domain-specific software applications in such a way that they incorporate the mechanisms of interaction as devices that support the articulation of distributed cooperative activities with respect to these applications — without imposing on actors an undue impedance between articulation work and work.

Now, cooperative work is articulated along multiple dimensions: who, what, where, when, how, etc.? These dimensions of articulation work are inter-dependent and these different aspects of articulation are thus themselves to be meshed in a fluent way. Since mechanisms of interaction are local and temporary closures, no mechanism of interaction has global validity. Hence, in order not to impose artificial distinctions and thus disrupt the ongoing articulation work, facilities should be provided that support the linking of different mechanisms (Malone et al., 1992). Mechanisms of interaction should therefore be conceived of as abstract devices that support the fluid interrelation of articulation work with respect to the *multiple applications* required to do the work in a particular setting. For instance, in the case of mechanical design, project management tools, CAD

tools, process planning systems, classification schemes for common repositories (of components, work in progress, drawings, patents), and so on.

The dimensions of articulation work — who, what, where, when, how, etc. — are interdependent and these different aspects of articulation are thus themselves to be meshed in a fluent way. Hence, in order not to impose artificial distinctions and thus disrupt the ongoing articulation work, facilities should be provided that support the linking of different mechanisms (Malone et al., 1992).

(5) It was argued previously that mechanisms of interaction require persistent cooperative management in order to be useful as a means of reducing the complexity and the cost of articulating distributed activities and that this management activity is itself a cooperative activity. We can thus state the following requirements for a CSCW application incorporating a mechanism of interaction (Schmidt, 1991; Schmidt, 1992):

(1) It should be visible to the members of the ensemble in terms of basic concepts pertaining to the articulation of cooperative work

(2) It should make the incorporated mechanism accessible to users and, indeed, support users in interpreting the mechanism and evaluating its rationale and implications.

(3) It should support users in applying and adapting the mechanism to the situation at hand; i.e., it should allow users to tamper with the way it is instantiated in the current situation, execute it or circumvent it, etc.

(4) It should be 'malleable' in the sense that it supports actors in modifying the underlying mechanism and in creating new mechanisms in accordance with the changing organizational realities and needs.

(5) Since the management of mechanisms of interaction is itself a cooperative activity, the system should support the documentation and communication of decisions to apply, adapt, modify, circumvent, execute, etc. the underlying mechanism.

(6) And in all of this the CSCW system as a whole, i.e., the CSCW platform, should support the process of negotiating the interpretation, application, adaptation, modification, circumvention, execution etc. of the mechanisms of interaction incorporated in various applications by providing general facilities for enacting and meshing an array of modes and means of interaction.

(7) Since the system should support cooperative management of the mechanism of interaction, the system should support multiple users in modifying the mechanism of interaction while being immersed in the very flow of distributed activities. This raises a host of problems such as control of propagation of changes and management of (in)consistency.

References

Andersen, N. E., F. Kensing, J. Lundin, L. Mathiassen, A. Munk-Madsen, M. Rasbech, and P. Sørgaard: *Professional Systems Development. Experience, Ideas, and Action*, Prentice-Hall, Englewood Cliffs, New Jersey, 1990.

Aoki, Masahiko: *A New Paradigm of Work Organization: The Japanese Experience*, WIDER Working Papers, vol. 36, World Institute for Development Economics Research, Helsinki, Finland, 1988.

Babbage, Charles: *On the Economy of Machinery and Manufactures*, (4th ed., 1835), Charles Knight, London, 1832. [Reprint published by Augustus M. Kelley, Publishers, Fairfield, New Jersey, 1986].

Bahrdt, Hans Paul: *Industriebürokratie. Versuch einer Soziologie des Industrialisierten Bürobetriebes und seiner Angestellten*, Ferdinand Enke Verlag, Stuttgart, 1958.

Bannon, Liam, and Kjeld Schmidt: "CSCW: Four Characters in Search of a Context," *EC-CSCW '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.

Barfod, Ari: *Forsøg med nye produktionssystemer*, Driftsteknisk Institut, Danmarks Tekniske Højskole, Februar, 1982.

Barnard, Chester I.: *The Functions of the Executive*, Harvard University Press, Cambridge, Mass., 1938.

Bellers, John: *Proposals for raising a colledge of industry of all useful trades and husbandry, with profit for the rich, a plentiful living for the poor, and a good education for youth*, London, 1696. - [Quoted in Karl Marx: *Das Kapital. Kritik der politischen Ökonomie*, vol. 1 (1867); MEGA, vol. II/5, p. 263.].

Bowker, Geoffrey, and Susan Leigh Star: "Situations vs. Standards in Long-Term, Wide-Scale Decision-Making: The Case of the International Classification of Diseases," *Proceedings of the Twenty-Fourth Annual Hawaii International Conference on System Sciences, Kauai, Hawaii, January 7-11, 1991*, edited by J. F. Nunamaker Jr. and R. H. Sprague Jr., IEEE Computer Society Press, 1991, vol. IV.

Ciborra, Claudio U.: "Reframing the role of computers in organizations: The transaction costs approach," *Proceedings of Sixth International Conference on Information Systems, Indianapolis, December 16-18, 1985*, 1985.

Cicourel, Aaron V.: "The Integration of Distributed Knowledge in Collaborative Medical Diagnosis," in *Intellectual Teamwork: Social and Technological Foundations of Cooperative Work*, ed. by J. Galegher, R. E. Kraut and C. Egido, Lawrence Erlbaum Associates, Hillsdale, New Jersey, 1990, pp. 221-242.

Commons, John R.: *Institutional Economics*, University of Wisconsin Press, Madison, 1934.

Cyert, Richard M., and James G. March: *A Behavioral Theory of the Firm*, Prentice-Hall, Englewood-Cliffs, New Jersey, 1963.

Dahrendorf, Ralf: *Sozialstruktur des Betriebes*, Gabler, Wiesbaden, 1959.

de Tracy, Destutt: *Élémens d'ideologie. Pt. 4.5. Traité de la volonté et ses effets*, Paris, 1826. - [Quoted in Karl Marx: *Zur Kritik der politischen Ökonomie (Manuskript 1861-63)*; MEGA, vol. II/3.1, p. 237.].

Ferguson, Adam: *An Essay on the History of Civil Society*, 1767. [Reprint published by Edinburgh University Press, Edinburgh, 1966].

Gerson, Elihu M., and Susan Leigh Star: "Analyzing Due Process in the Workplace," *ACM Transactions on Office Information Systems*, vol. 4, no. 3, July 1986, pp. 257-270.

Goody, Jack: *The Logic of Writing and the Organization of Society*, Cambridge University Press, Cambridge, 1986.

Goody, Jack: *The Interface Between the Written and the Oral*, Cambridge University Press, Cambridge, 1987.

Greenberg, Saul: "Personalizable groupware: Accomodating individual roles and group differences," in *ECSCW '91. Proceedings of the Second European Conference on Computer-Supported Cooperative Work*, ed. by L. Bannon, M. Robinson and K. Schmidt, Kluwer Academic Publishers, Amsterdam, 1991, pp. 17-31.

Grudin, Jonathan: "Why groupware applications fail: problems in design and evaluation," *Office: Technology and People*, vol. 4, no. 3, 1989, pp. 245-264.

Gunn, Thomas G.: *Computer Applications in Manufacturing*, Industrial Press, New York, 1981.

Gunn, Thomas G.: *Manufacturing for Competitive Advantage. Becoming a World Class Manufacturer*, Ballinger, Cambridge, Mass., 1987.

Harper, Richard, John A. Hughes, Dan Shapiro, and Wes Sharrock: *Ordering the Skies: The Sociology of Coordination Work*, Routledge, London, Forthcoming.

Harper, Richard H. R., and John A. Hughes: "What a f-ing system! Send 'em all to the same place and then expect us to stop 'em hitting. Managing technology work in air traffic control," in *Technology in Working Order. Studies of work, interaction, and technology*, ed. by G. Button, Routledge, London and New York, 1993, pp. 127-144.

Harper, Richard R., John A. Hughes, and Dan Z. Shapiro: *The Functionality of Flight Strips in ATC Work. The report for the Civil Aviation Authority*, Lancaster Sociotechnics Group, Department of Sociology, Lancaster University, January, 1989a.

Harper, R. R., J. A. Hughes, and D. Z. Shapiro: "Working in harmony: An examination of computer technology in air traffic control," *EC-CSCW '89. Proceedings of the First European Conference on Computer Supported Cooperative Work, Gatwick, London, 13-15 September, 1989*, 1989b, pp. 73-86.

Harper, R. R., J. A. Hughes, and D. Z. Shapiro: "Harmonious Working and CSCW: Computer technology and air traffic control," in *Studies in Computer Supported Cooperative Work. Theory, Practice and Design*, ed. by J. M. Bowers and S. D. Benford, North-Holland, Amsterdam etc., 1991, pp. 225-234.

Harrington, Joseph: *Computer Integrated Manufacturing*, Krieger, Malabar, Florida, 1979.

Harrington, Joseph: *Understanding the Manufacturing Process. Key to Successful CAD/CAM Implementation*, Marcel Dekker, New York, 1984.

Hart, H. A. L.: *The Concept of Law*, Oxford University Press, London, 1961.

Heath, Christian, Marina Jirotka, Paul Luff, and Jon Hindmarsh: "Unpacking Collaboration: the Interactional Organisation of Trading in a City Dealing Room," in *ECSCW '93. Proceedings of the Third European Conference on Computer-Supported Cooperative Work, 13-17 September 1993, Milan, Italy*, ed. by G. De Michelis, C. Simone and K. Schmidt, Kluwer Academic Publishers, Dordrecht, 1993, pp. 155-170.

Heath, Christian, and Paul Luff: "Collaborative Activity and Technological Design: Task Coordination in London Underground Control Rooms," in *ECSCW '91. Proceedings of the Second European Conference on Computer-Supported Cooperative Work*, ed. by L. Bannon, M. Robinson and K. Schmidt, Kluwer Academic Publishers, Amsterdam, 1991, pp. 65-80.

Heath, Christian, and Paul Luff: "Collaboration and Control. Crisis Management and Multimedia Technology in London Underground Control Rooms," *Computer Supported Cooperative Work (CSCW). An International Journal*, vol. 1, no. 1-2, 1992, pp. 69-94.

Holt, Anatol W.: "Coordination Technology and Petri Nets," in *Advances in Petri Nets 1985*, ed. by G. Rozenberg, Lecture Notes in Computer Science, ed. by G. Goos and J. Hartmanis, vol. 222, Springer-Verlag, Berlin, 1985, pp. 278-296.

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*, ed. by L. Bannon, M. Robinson and K. Schmidt, Kluwer Academic Publishers, Amsterdam, 1991, pp. 309-323.

Hughes, John A., David Randall, and Dan Shapiro: "Faltering from Ethnography to Design," in *CSCW '92. Proceedings of the Conference on Computer-Supported Cooperative Work, Toronto, Canada, October 31 to November 4, 1992*, ed. by J. Turner and R. Kraut, ACM Press, New York, 1992, pp. 115-122.

Hughes, John A., Dave Randall, and Dan Shapiro: "From Ethnographic Record to System Design. Some experiences from the field," *Computer Supported Cooperative Work (CSCW). An International Journal*, vol. 1, no. 3, 1993, pp. 123-141. - Forthcoming.

Hughes, John A., Dan Z. Shapiro, Wes W. Sharrock, and Robert Anderson: *The Automation of Air Traffic Control*, Lancaster Sociotechnics Group, Department of Sociology, Lancaster University, October, 1988.

IBM: *Communications Oriented Production Information and Control System*, vol. 1-12, IBM Corporation, Technical Publications Dept., White Plains, New York, 1972.

Kaavé, Bjarne: *Undersøgelse af brugersamspil i system til produktionsstyring*, M.Sc diss., Technical University of Denmark, 1990.

Kaavé, Bjarne: Personal communication, November, 1992.

Kasbi, Catherine, and Maurice de Montmollin: "Activity Without Decision and Responsibility: The Case of Nuclear Power Plants," in *Distributed Decision Making. Cognitive Models for Cooperative Work*, ed. by J. Rasmussen, B. Brehmer and J. Leplat, John Wiley & Sons, Chichester etc., 1991, pp. 275-283.

Kern, Horst, and Michael Schumann: *Industriearbeit und Arbeiterbewußtsein. Eine empirische Untersuchung über den Einfluß der aktuellen technischen Entwicklung auf die industrielle Arbeit und das Arbeiterbewußtsein*, vol. 1-2, Europäische Verlagsanstalt, Frankfurt am Main, 1970.

Kling, Rob: "Social Analyses of Computing: Theoretical Perspectives in Recent Empirical Research," *Computing Surveys*, vol. 12, no. 1, March 1980, pp. 61-110.

Malone, Thomas W., and Kevin Crowston: "What is Coordination Theory and How Can It Help Design Cooperative Work Systems," in *CSCW '90. Proceedings of the Conference on Computer-Supported Cooperative Work, Los Angeles, Calif., October 7-10, 1990*, ACM press, New York, N.Y., 1990, pp. 357-370.

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*, ed. by J. Turner and R. Kraut, ACM Press, New York, 1992, pp. 289-297.

Marx, Karl: "Einleitung" (1857); in K. Marx and F. Engels: *Gesamtausgabe (MEGA)*, vol. II/1.1, Dietz Verlag, Berlin, 1976, pp. 21-45.

Marx, Karl: *Zur Kritik der politischen Ökonomie* (1861-63); in K. Marx and F. Engels: *Gesamtausgabe (MEGA)*, vol. II/3.1-II/3.6, Dietz Verlag, Berlin, 1976-1982.

Marx, Karl: *Das Kapital. Zur Kritik der politischen Ökonomie. Erster Band* (Hamburg1867); in K. Marx and F. Engels: *Gesamtausgabe (MEGA)*, vol. II/5, Dietz Verlag, Berlin, 1983.

McGraw-Hill: *McGraw-Hill Encyclopedia of Science and Technology*, (6th Edition), McGraw-Hill, New York etc., 1987.

Mickler, Otfried, Eckhard Dittrich, and Uwe Neumann: *Technik, Arbeitsorganisation und Arbeit. Eine empirische Untersuchung in der automatischen Produktion*, Aspekte Verlag, Frankfurt am Main, 1976.

Mintzberg, Henry: *The Structuring of Organizations. A Synthesis of the Research*, Prentice-Hall, Englewood Cliffs, New Jersey, 1979.

Monden, Yasuhiro: *Toyota Production System. Practical Approach to Production Management*, Industrial Engineering and Management Press, Institute of Industrial Engineers, Norcross, Georgia, 1983.

Montesquieu: *Lettres persanes*, 1721. [Reprint published by GF-Flammarion, Paris, 1964].

Newton, Janice: "Technology and Cooperative Labour Among the Orokaiva," *Mankind*, vol. 15, no. 3, December 1985, pp. 214-222.

Orlikowski, Wanda J.: "Learning from NOTES: Organizational Issues in Groupware Implementation," in *CSCW '92. Proceedings of the Conference on Computer-Supported Cooperative Work, Toronto, Canada, October 31 to November 4, 1992*, ed. by J. Turner and R. Kraut, ACM Press, New York, 1992, pp. 362-369.

Peirce, Charles Sanders: "On Signs and the Categories" (1901); *The Collected Papers of Charles Sanders Peirce*, edited by C. Hartshorne, P. Weiss, and A. W. Burks, vol. 8, Harvard University Press, Cambridge, Mass., 1931-1966, pp. 220-231.

Perrow, Charles: *Normal Accidents. Living with High-Risk Technologies*, Basic Books, New York, 1984.

Popitz, Heinrich, Hans Paul Bahrdt, Ernst A. Jüres, and Hanno Kesting: *Technik und Industriearbeit. Soziologische Untersuchungen in der Hüttenindustrie*, J. C. B. Mohr, Tübingen, 1957.

Rasmussen, Jens: "A Cognitive Engineering Approach to the Modelling of Decision Making and Its Organization in Process Control, Emergency Management, CAD/CAM, Office Systems, Library Systems," in *Advances in Man-Machine Systems Research*, ed. by W. B. Rouse, vol. 4, JAI Press, Greenwich, Conn., 1988, pp. 165-243.

Rodden, Tom, and Gordon Blair: "CSCW and Distributed Systems: The Problem of Control," in *ECSCW '91. Proceedings of the Second European Conference on Computer-Supported Cooperative Work*, ed. by L. Bannon, M. Robinson and K. Schmidt, Kluwer Academic Publishers, Amsterdam, 1991, pp. 49-64.

Rouse, William B.: *Systems Engineering Models of Human-Machine Interaction*, System Science and Engineering, ed. by A. P. Sage, Horth-Holland, New York and Oxford, 1980.

Savage, Charles M. (ed.): *Fifth Generation Management for Fifth Generation Technology (A Round Table Discussion)*, Society of Manufacturing Engineers, Dearborn, Michigan, 1987.

Schmidt, Kjeld: *Analyse af et investeringscenter [Analysis of a portfolio management agency]*, Dansk Datamatik Center, Lyngby, Denmark, 15 December, 1987. - [Confidential report, in Danish].

Schmidt, Kjeld: *Analysis of Cooperative Work. A Conceptual Framework*, Risø National Laboratory, DK-4000 Roskilde, Denmark, June, 1990. [Risø-M-2890].

Schmidt, Kjeld: "Riding a Tiger, or Computer Supported Cooperative Work," in *ECSCW '91. Proceedings of the Second European Conference on Computer-Supported Cooperative Work*, ed. by L. Bannon, M. Robinson and K. Schmidt, Kluwer Academic Publishers, Amsterdam, 1991, pp. 1-16.

Schmidt, Kjeld: *Initial Notes on Mechanisms of Interaction*, Risø National Laboratory, Roskilde, Denmark, 18 November, 1992. [COMIC-Risø-3-1].

Schmidt, Kjeld, and Liam Bannon: "Taking CSCW Seriously: Supporting Articulation Work," *Computer Supported Cooperative Work (CSCW). An International Journal*, vol. 1, no. 1-2, 1992, pp. 7-40.

Schonberger, Richard J.: *Japanese Manufacturing Techniques. Nine Hidden Lessons in Simplicity*, The Free Press, New York, 1982.

Sheridan, Thomas B., and William R. Ferrell: *Man-Machine Systems: Information, Control, and Decision Models of Human Performance*, (paperback ed. 1981), The MIT Press, Cambridge, Mass., 1974.

Smith, Adam: *The Wealth of Nations*, vol. 1-2, 1776. [Reprint published by Dent, London, 1964-66].

Star, Susan Leigh: "The Structure of Ill-Structured Solutions: Boundary Objects and Heterogeneous Distributed Problem Solving," in *Distributed Artificial Intelligence*, ed. by L. Gasser and M. Huhns, vol. 2, Pitman, London, 1989, pp. 37-54.

Stinchcombe, Arthur L.: *Creating Efficient Industrial Administrations*, Academic Press, New York and London, 1974.

Strauss, Anselm: "Work and the Division of Labor," *The Sociological Quarterly*, vol. 26, no. 1, 1985, pp. 1-19.

Strauss, Anselm: "The Articulation of Project Work: An Organizational Process," *The Sociological Quarterly*, vol. 29, no. 2, 1988, pp. 163-178.

Strauss, Anselm, Shizuko Fagerhaugh, Barbara Suczek, and Carolyn Wiener: *Social Organization of Medical Work*, University of Chicago Press, Chicago and London, 1985.

Suchman, Lucy A.: "Office Procedures as Practical Action: Models of Work and System Design," *ACM Transactions on Office Information Systems*, vol. 1, no. 4, October 1983, pp. 320-328.

Suchman, Lucy A.: *Plans and situated actions. The problem of human-machine communication*, Cambridge University Press, Cambridge, 1987.

Susman, Gerald I., and Richard B. Chase: "A Sociotechnical Analysis of the Integrated Factory," *The Journal of Applied Behavioral Science*, vol. 22, no. 3, 1986, pp. 257-270.

Thompson, James D.: *Organizations in Action. Social Science Base of Administrative Theory*, Mc Graw-Hill, New York etc., 1967.

Ure, Andrew: *The Philosophy of Manufactures: or, an Exposition of the Scientific, Moral, and Commercial Economy of the Factory System of Great Britain*, Charles Knight, London, 1835.

Wakefield, Edward: *A View of the Art of Colonization, with present reference to the British Empire; in letters between a stateman and a colonist*, John W. Parker, London, 1849.

Winograd, Terry, and Fernando Flores: *Understanding Computers and Cognition: A New Foundation for Design*, Ablex Publishing Corp., Norwood, New Jersey, 1986.

Woods, David D.: "Coping with complexity: the psychology of human behavior in complex systems," in *Tasks, Errors and Mental Models. A Festschrift to celebrate the 60th birthday of Professor Jens Rasmussen*, ed. by L. P. Goodstein, H. B. Andersen and S. E. Olsen, Taylor & Francis, London etc., 1988, pp. 128-148.

Part 2

Computational mechanisms of interaction

Table of Contents, Part 2

Computational Mechanisms of Interaction: Notations and Facilities

Kjeld Schmidt*, Carla Simone°, Peter Carstensen*,
Betty Hewitt*, and Carsten Sørensen*

*Risø National Laboratory
°University of Milano

> "The new capabilities at which coordination technology aims depend on finding and installing appropriate conceptual and structural units with which to express tasks, their diverse relations to each other and to the people who ultimately bear responsibility for them." "To be useful, this must be done in a flexible yet well-integrated manner, with plenty of leeway for the unpredictability of real life."
> (Holt, 1985, p. 281)

2.1. The concept of mechanisms of interaction

By involving multiple actors, cooperative work is characterized by an inexorable and inescapable aspect of distributed decision making, not only in the usual sense that activities are distributed in time and space, but also — and more importantly — in the sense that the actors are semi-autonomous in terms of heuristics and conceptualizations.

The distributed character of cooperative work varies depending on a number of factors, e.g., 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, the apperceptive complexity posed by the field of work and hence the variety of heuristics involved, and so on. The more distributed the activities of the cooperative work arrangement, the more complex the articulation of the activities of that arrangement.

With low degrees of complexity, the articulation of cooperative work can be achieved by means of the modes of interaction characteristic of everyday social life. In fact, under such conditions, the required articulation of individual activities in cooperative work is managed *so* effectively and efficiently by the rich variety of intuitive interactional modalities of everyday social life that the distributed nature of cooperative work is not apparent — most of the time. People

tacitly monitor each other; they make their activities sufficiently apperceptible for others; they take each others' past, present and prospective activities into account in planning and conducting their own work; they gesture, talk, write to each other, and so on. Accordingly, much of the research in CSCW has focused on the 'shallow' support strategy of providing enhanced means of communication (desk top audio and video, shared displays, etc.), either in order to enable actors to cooperate more effectively and efficiently in spite of geographical distance, or in order to widen the repertoire of communication facilities.

However, with the complex work environments of modern industrial and administrative organizations, the problems of articulating distributed activities are at a different order of complexity. The everyday social and communication skills are far from sufficient in articulating the cooperative efforts of hundreds or thousands of actors engaged in myriads of complexly interdependent activities, perhaps concurrently, intermittently, or indefinitely.

In order to handle a high degree complexity of articulation work, and handle it efficiently, articulation of cooperative work requires certain modes of interaction that involve 'deep' support by means of a category of artifacts that we have dubbed 'mechanisms of interaction'. As argued in the analysis of "Modes of mechanisms of interaction" in Part 1, a mechanism of interaction can be defined as a device for reducing the complexity of articulating distributed activities of large cooperative ensembles by *stipulating and mediating* the articulation of the distributed activities.

In order to serve this function, such a device must have the following characteristics:

1. **Artifact**: It must be publicly available in the sense that it does not solely reside 'in the head' of human actors (such as conventions) and it must be persistent in the sense that it is available independently of any particular situation. That is, it must exist in the form an artifact.
2. **Symbolic**: It must be possible to manipulate the mechanism independently of the state of the field of work. The artifact incorporating the mechanism of interaction must not be coupled in any strong, tight or irreversible way to the state of field of work. Appropriate manipulations of the device for articulation purposes should be feasible without unwanted side-effects on the field of work. That is, the artifact must have the character of a symbolic artifact.
3. **Standardized format**: It must provide affordances to and impose constraints on work articulation and it must make the state of work articulation at any given moment publicly perceptible.

As a device for the articulation of cooperative work, a mechanism of interaction should be distinguished from other symbolic artifacts regulating human affairs (e.g., "Stop!", "Dead Slow!", "Pharmacy", "Wine & Spirits", "Underground", "Bus Stop"). A traffic light regulating the traffic at an intersection, for example, has certain facilities in common with a mechanism of interaction in

cooperative work: It is a symbolic artifact that actively stipulates and mediates interaction (Gustavsson, 1993). However, the drivers — as drivers — are not engaged in cooperative work; to the contrary, they are competing for the same resource (the road) and to each and every one of them the others are a mere nuisance. Their interaction is ephemeral and superficial and does not entail the rich multiplicity of purposive reciprocal interactions of cooperative work.

Also, again for the sake of clarification, a mechanism of interaction should be distinguished from the information objects that are ubiquitous in cooperative work settings (such as letters, memoranda, drawings and reports, and the aggregation of such documents in the form of files and libraries) and which help to articulate and mediate distributed activities. Written documents are certainly symbolic artifacts but in so far as they serve to convey information and hence merely provide a medium of communication, they are not mechanisms of interaction. However, managing the complex flow of information objects may require mechanisms of interaction such as standardized formats (e.g., prescribed forms and routing instructions on file covers) and classification schemes (e.g., library catalogues and thesauri).

The definition of a mechanism of interaction given above is, of course, static: It does not relate to the fact that the allocation of function between human and artifact in the design and use of mechanisms of interaction is dynamic and evolving. There is a clear ladder from, at the one end of the spectrum, modes of interaction that do not involve any pre-specified stipulations — to modes of interaction that involve the active mediation of pre-specified artifacts, at the other end of the spectrum:

- Ad hoc articulation (by means of monitoring others, directing attention, embedding cues in artifacts, and negotiating). This mode of interaction offers a high degree of local control and hence very powerful means of recovery from misunderstanding and error and for handling contingencies. On the other hand, ad hoc articulation is highly inefficient when faced with recurring problems, and it may be difficult to anticipate the course of the cooperative effort and hold actors accountable.
- Articulation by means of conventions, i.e., the usual and expected way to do things. Whether the conventions are made explicit or merely observed tacitly, this mode relies wholly on mutual understanding and the will to adhere to the expectations, supported by various forms of social sanction. Given that, it may still be difficult to anticipate the course of the cooperative effort and hold actors accountable.
- Articulation by means of stipulations supported by artifacts in the form of written statutes, e.g., standard operating procedures or accounting prescriptions. As opposed to conventions, stipulations supported by artifacts in the form of written statutes are, in principle, accessible by all at any time. That is, the fact that the stipulations are supported by symbolic artifacts makes everybody accountable in a far higher degree. The execution of the stipula-

tion, however, relies completely on human recollection and the stipulation is quite open to interpretation.

- Articulation by means of stipulations supported by passively mediating symbolic artifacts, e.g., forms, organizational charts, thesauri, which by imposing a standardized format restricts the 'degrees of freedom' each user is faced with. In these cases, the execution of the stipulation no longer relies completely on human recollection and the range of (reasonable) interpretation is relatively narrow compared to, say, statutes.
- Articulation by means of stipulations supported by *actively* mediating symbolic artifacts, e.g., time tables, indexing systems for repositories, kanban-systems, computer-based scheduling systems, workflow management systems.

Of course, using a mechanism of interaction always requires certain social conventions and skills but in highly complex cooperative work settings these conventions and skills must be supplemented and supported by a mechanism in the form of an artifact, at least to the extent that it provides a plan for action that can be carried out without further consultation and a verifiable (interpersonal and situation-independent) basis for holding actors accountable.

Given the infinite versatility of computer systems, it is most likely — and this is the underlying contention of the work within Strand 3 of COMIC — that computer-based mechanisms of interaction can provide a degree of visibility and flexibility to mechanisms of interaction that was unthinkable with previous technologies, typically based on inscriptions on paper or cardboard. This opens up new prospects for moving the allocation of functionality between human and artifact with respect to articulation work. The challenge is to change the allocation of functionality between human and artifact — so that much of the drudgery of articulation work (boring operations that have so far relied on human effort and vigilance) can be delegated to the artifact; so that cooperative ensembles can articulate their distributed activities with a higher degree of flexibility and effectiveness, and, no doubt, so that the cooperative ensembles will be able to handle an even higher degree of complexity in the articulation of their distributed activities!

The aim of providing 'deep' and at the same time flexible support for the articulation of distributed activities is shared by many researchers in CSCW, witness (e.g., (Johnson, 1992; Kaplan et al., 1992b; Malone et al., 1992; Kreifelts et al., 1993)). What distinguishes the approach taken here, is the *conception of mechanisms of interaction as abstract devices incorporated in domain-specific applications.*

As observed above, modes of interaction in articulation work differ with respect to the degree of local control. With a high degree of local control, the different modes of interaction do not impose domain-specific affordances and constraints on the conduct of articulation work. The affordances and constraints of these modes of interaction reflect those of the different media: bandwith, turn-

around time, freedom of movement, etc.(Gaver, 1992). On the other hand, when articulation work is stipulated and mediated by artifacts, the structure of the artifacts inexorably embodies domain-specific affordances and constraints. That is, a symbolic artifact stipulating and mediating the articulation of distributed activities is domain specific.

In terms of CSCW systems architectures, then, the following general architecture can be outlined: On one hand, a CSCW system should provide a variety of domain-independent facilities supporting local control of articulation work (i.e., the plethora of enhanced computer-based means of communication). On the other hand, in order to support large-scale cooperative work, a CSCW system should provide facilities for stipulating and mediating articulation work by means of affordances and constraints embodied in domain-specific applications (Schmidt and Rodden, 1993).

The concept of mechanism of interaction has been developed in order to tackle the latter aspect of CSCW systems in a systematic manner. A mechanism of interaction should thus be conceived of as an abstract device incorporated in a software application (e.g., a CASE tool, an office information system, a CAD system, a production control system, etc.) so as to support the articulation of the distributed activities of multiple actors with respect to that application.

There are, however, certain classes of applications that seem to be entirely devoted to articulation work such as, for example, project management systems (e.g., NOW UP-TO-DATE, ACTIVE MEMORY, TASK MANAGER). By providing facilities for mapping Actors and Tasks, these applications support highly abstract aspects of articulation work that are common to a variety of work domains (e.g., construction, manufacturing design, software engineering, advertising). In a certain sense, these systems can be likened to spreadsheet programs that provide generic facilities for designing and manipulating domain-specific arithmetical models: they are standard tools for designing domain-specific project management applications. Being standard software programs, however, these systems do not provide facilities for interfacing the articulation with respect to Actor and Task with other dimensions of articulation work (e.g., the field of work, activities, conceptual structures, resources).

A computational mechanism of interaction should thus be conceived of as an abstract device incorporated in a particular software application (e.g., a CASE tool, an office information system, a CAD system, a production control system, etc.) so as to support the articulation of the distributed activities of multiple actors with respect to that application. The concept of mechanisms of interaction has been developed in order to facilitate the design of domain-specific software applications in such a way that they incorporate the mechanisms of interaction as accessible and manipulable devices that support the articulation of distributed cooperative activities with respect to these applications — without imposing on actors an undue impedance between articulation work and work. The conception of mechanisms of interaction as abstract devices provides a conceptual foundation for making a distinction between (a) the domain-specific functionality of

applications (e.g., accounting, software engineering, project management, scheduling, production control) and (b) the facilities required in order to support the articulation work involved in doing the work (in addition to the host of other facilities to be provided by a CSCW platform, such as desktop audio and video, shared displays, shared object services and the concomitant turntaking and access control protocols). As abstract devices, the mechanisms of interaction are intended to facilitate the design of domain-specific applications *in such a way* that they support the fluid interrelation of articulation work with respect to the *multiple applications* required to do the work in a particular setting, without imposing any impedance on the articulation of cooperative work with respect to different application. For instance, in the case of mechanical design, project management tools, CAD tools, process planning systems, classification schemes for common repositories (of previous designs, components, work in progress, drawings, patents), and so on.

A mechanism of interaction can be seen from two aspects: On one hand, the underlying abstract logic or grammar of the mechanism's behavior, which we call its *notation*. On the other hand the *facilities* which allow users to access and manipulate the mechanism of interaction.

2.2. The concept of notations of computer artifacts

The purpose of the concept of notation in this context is to provide a means for analyzing and specifying the behavior of an artifact that mediates articulation work in a systematic and abstract manner.

In this section we will discuss the concept of notations of computer artifacts for the purpose of a comparative analysis of underlying notations of mechanisms of interaction in current CSCW applications and environments.

In our context, the term 'notation' is used in the classical sense of a set of symbolic primitives, a set of rules of operation specifying how primitives may be combined (syntactic rules), and a set of rules stating what a given statement means (semantic rules) (Naur, 1992).

When the term 'notation' is used in this way, 'notations' of all sorts can be attributed to computer systems.

First, programming languages are obviously notations — computational notations: they provide an explicit set of primitives, syntactic and semantic rules. This is simple enough. At another level, however, a notation as embodied by a programming language can be described by means of a different notation (e.g., BNF or VDM). So, when referring to a specific notation it is important that one is perfectly clear as to which notation one is referring to: the target notation itself or the notation used to describe the target notation.

Similarly, development shells (e.g., HYPERCARD) provide notations in the shape of the programming language provided by the shell (e.g., HYPERTALK and the associated generic graphical objects such as buttons and blinds) and the notation of the shell user interface.

Second, the interface of a specific software application (e.g., MICROSOFT WORD) can be conceived of as a notation, that is, the primitives and the rules of combination and application of graphical facilities such as windows, dialog boxes, alert boxes, and so forth.

Third, and most importantly for our purpose, the functionality of a specific software application can be conceived of as a specific notation of interaction (Figure 2-1), in the sense that (1) it provides a set of primitives as incorporated in the various selectable options (e.g., commands, menu choices, dialog box options, classes of objects, etc.); (2) it imposes certain constraints on how these primitives can be combined (e.g., options are dimmed when not applicable according to syntax), and (3) when evoked the different options invoke certain operations.

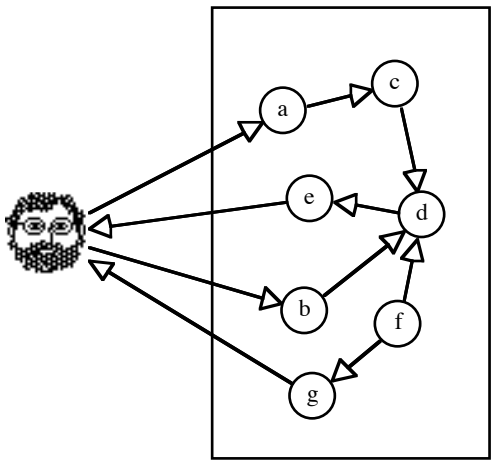


Figure 2-1. The functionality of a specific software application can be conceived of as a specific notation of human-computer interaction in the sense that the different options (a, b) are a set of primitives that can only be combined in certain ways and that have certain effects (e, g).

Conceived of in this way, the notation is *implicit* in the sense that it is not directly observable; it can only be inferred by analyzing the behavior of the system, that is, through a process of reverse engineering.

This use of the term 'notation' may seem unorthodox — but it makes perfect sense in so far as the behavior of the artifact can be described by and attributed to an explicit notation, through reverse engineering, *as if* the explicit notation had existed prior to the design of the artifact and had been used to specify its behavior.[1]

---

1 Actually, attributing an 'implicit notation' to the behavior of an artifact is not quite as unorthodox as it may seem. A similar approach has, for instance, been taken in the field of dialogue modeling including several attempts at specifying the interaction between users and the computer. Hartson and Hix (1989) suggests a framework for human-computer interface development. A central characteristic of the their framework is an explicit distinction between "structural models" theoretically describing structure of

In the context of this investigation, the notations we need to understand are the *notations of mechanisms of interaction*. That is, our investigation focuses on notations of computational mechanisms of interaction at the 'use level' or the semantic level of articulation work. Using the term 'notation' in the rather unorthodox way outlined above enables us to subject existing CSCW systems to an analysis of the incorporated mechanisms of interaction that systematically abstracts from those aspects of the systems that are not pertinent to articulation work. So, when studying existing CSCW systems, the objective has not been to study and compare the variety of notations as embodied in these systems or all the facilities and characteristics of these systems, but to study *the notations of exactly those features of the systems that mediate the articulation of cooperative work.* Similarly, in the case of specification languages and CSCW shells, the objective has been to study exactly those facilities of the systems that can be used to model, encode, and implement facilities that mediate the articulation of cooperative work.

In keeping with the above discussion, computational notations of mechanisms of interaction may be conceived of as being either explicit or implicit: notations for describing, specifying, simulating, and/or incorporating mechanisms of interaction in software systems, for example abstract notations (e.g., Petri nets, Dialogue cells), 'specification languages' (e.g., ICN), conceptual schemes for modelling cooperative work (e.g., COSMOS, AMIGO), or the software construction languages of the various shells and tools for developing CSCW applications (e.g., LOTUS NOTES, OVAL, CONVERSATIONBUILDER, EGRET) — are obviously explicit notations.

On the other hand, software systems that in some way mediate the articulation of cooperative work, for example CSCW systems such as THE COORDINATOR, ASPECTS, and INFORMATION LENS, can be conceived of as providing a certain notation for the articulation-related interactions between the actors. By way of illustration, consider two actors articulating their cooperative activities (in part, probably) by means of some computer artifact (Figure 2-2). The artifact offers certain options (functional primitives) that can be combined in certain ways and that will have particular effects in terms of articulating the interaction between the two.

---

the user-computer communication in general terms, and the "representation of the interface" specifying a particular instance of the human-computer interaction.

The Dialogue Cells (Borufka et al., 1982) is an example of a notation addressing the structural model and the state-transition based RAPID/USE (Wasserman, 1985) is an example of the representation level of their framework. An example of a set of specification languages (notations) covering both approaches is the Command Language Grammar (CLG) (Moran, 1981). The idea of CLG is to cover three different viewpoints: 1) as elaborating the structure of a system's interface and user-system interaction, 2) as describing the users understanding (model) of the system, and 3) as representations for specifying the design of the concrete system design (Moran, 1981, p. 3). The first view reflects, in the terms of Hix and Hartson the structural model and the third view reflects the representation.

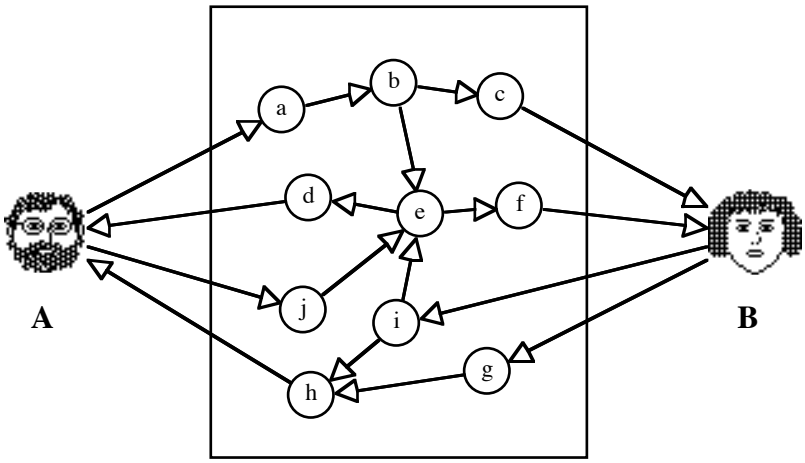


Figure 2-2. Two actors articulating their cooperative activities by means of a computer artifact. The artifact offers certain options (functional primitives) that can be combined in certain ways and that will have certain effects in terms of articulating the interaction between the two.

Now, what does it mean to *make explicit* notations of coordination mechanisms that are incorporated in software systems and therefore only observable indirectly, by means of reverse engineering?

Let us, for the sake of the argument, take THE COORDINATOR as an illustration. It is a very useful example because the notation of the incorporated mechanism of interaction has been made explicit by the designers, not in the form of facilities of the system itself, of course, but in their writings on the system (Winograd and Flores, 1986). For example, the diagram in Figure 2-3 presents the core mechanism of interaction of the system. The notation of this mechanism has been derived from speech act theory, interpreted as a state transition machine.

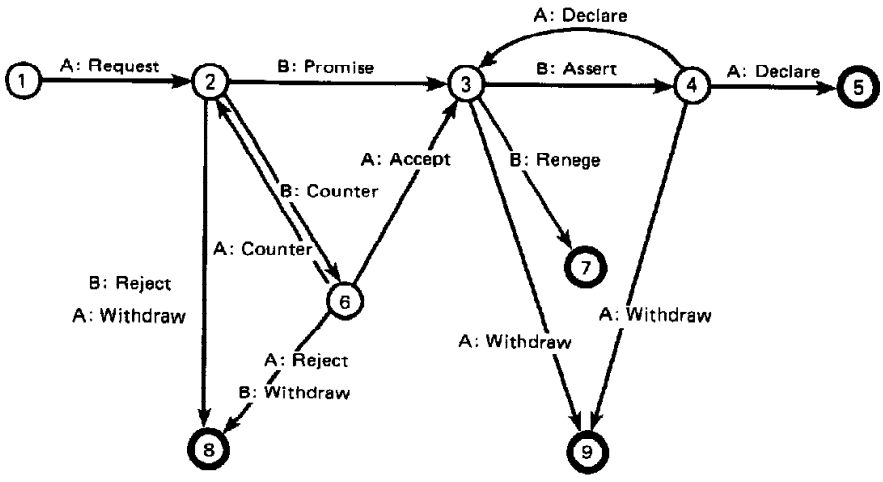


Figure 2-3. The notation of the mechanism of interaction incorporated in THE COORDINATOR (Winograd and Flores, 1986, p. 65).

It is well known that THE COORDINATOR has been severely criticized from different perspectives. The basic tenet of the critique has been that it imposes a cer-

tain structure on the interaction. This critique is tossing the baby out with the bath water. The problem with THE COORDINATOR, in our opinion, is not that it provides a mechanism of interaction but (1) that THE COORDINATOR uses 'conversation for action' as a metaphor for articulation work and hence only supports articulation work in terms of obligations; (2) that the specific form of mechanism derived from that is not generally valid and may not be valid beyond work domains characterized by explicit command and control structures; and (3) that the way the mechanism of interaction is incorporated in the system does not allow actors to manage the mechanism: it provides no *visibility and control* of the mechanism to users.

Be that as it may, the point to be made here is that THE COORDINATOR is an excellent example of a CSCW system that incorporates a mechanism of interaction that *can be* (and, indeed, *has been*) made explicit. What we have tried do with respect to CSCW systems (cf. Part 3) is, in a way, to do what the Winograd et al. did themselves for THE COORDINATOR.

The point can be clarified further by taking a mundane single-user spreadsheet that does not support articulation work in any way as an example. A particular spreadsheet (as opposed to the program by means of which it is made and manipulated) may for instance incorporate a model of the budget of a department (Figure 2-4). The spreadsheet model will behave in a certain way as values of the different cells are changed and the logic of that behavior can be conceived of in terms of a specific (implicit) notation.

		January	February
Income			
	Fiscal Year allotment	21,000	21,000
	Contracts	20,000	25,000
	Consultancy	1,400	2,000
	Contributions from sponsors	4,000	4,000
	Total	46,400	52,000
Expenditures			
	Labor	24,000	24,000
	Travel and subsistence	14,000	12,000
	Durable equipment	2,500	2,500
	Consumables	2,000	2,000
	Services	5,000	5,000
	Total	47,500	45,500
Balance		(1,100)	6,500

Figure 2-4. A simple model of the budget of a department in the shape of a spreadsheet. The behavior of the model can be conceived of in terms of a specific (implicit) notation.

Now, a spreadsheet program such as MICROSOFT EXCEL also provides an *explicit* notation for specifying the incorporated model. This notation provides as set of primitives (data types and operators), aggregations of operators in the form of

complex functions (Figure 2-5), and a script notation for specifying the relations between cells in the Formula Bar (Figure 2-6):

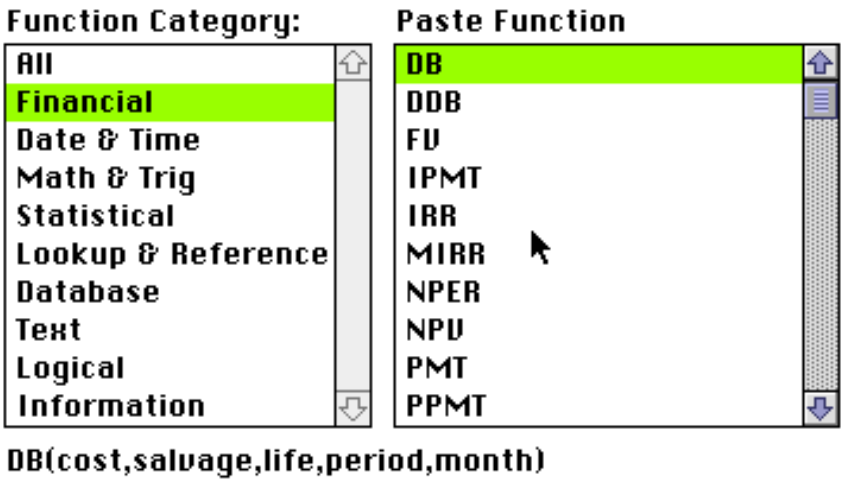


Figure 2-5. MICROSOFT EXCEL provides a set of primitives and aggregations of primitives (in the form of complex functions) for specifying relationships between cells.

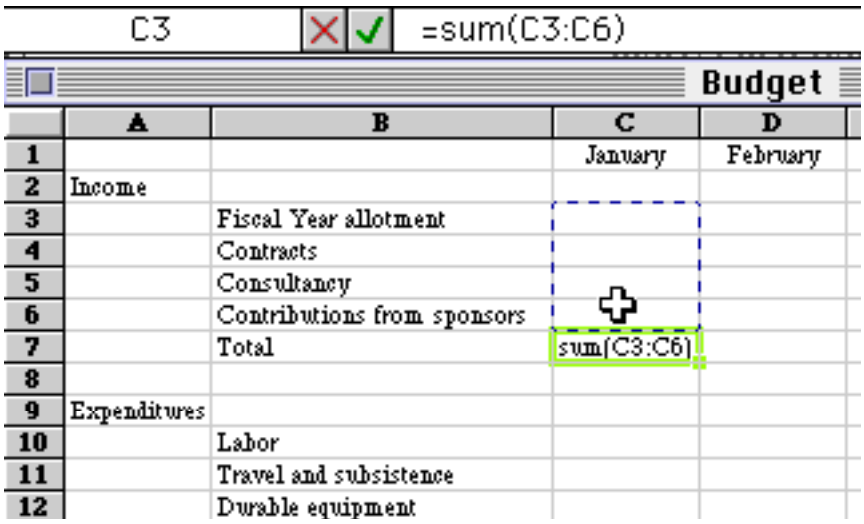


Figure 2-6. MICROSOFT EXCEL provides a notation for specifying relationships between cells.

In the following we need to distinguish these notations, both in order to handle the analyses of mechanisms of interaction in existing CSCW systems and in order to discuss the requirements for computational notations for designing mechanisms of interaction. We will refer to the notation embodied in the behavior of the artifact, whether explicit or not, as the *α notation*, whereas we will call the notation for specifying the α notation the *β notation*.

In the case of the spreadsheet example, the β notation is readily available (in the Formula Bar) for specifying and respecifying the α notation of the budget model (Figure 2-6). The α notation is explicit in the sense that relationships

between cells are expressed in the β notation, that is, the β notation is used a the descriptive notation for the α notation. In the case of THE COORDINATOR, on the other hand, the α notation is implicit and no β notation is incorporated in the system (it only exists in the literature).

2.3. Notations of computational mechanisms of interaction

The purpose of the concept of notation is not primarily to provide a means for analyzing the behavior of mechanisms of interaction in current CSCW applications and environments but, eventually, to serve as a design concept for the development of CSCW systems.

In accordance with the terminology introduced in Section 2.2 above, we will distinguish between the α notation as embodied by the mechanism (implicitly or explicitly) and the β notation used for its specification.

The α notation is the set of primitives and syntactic and semantic rules that govern the behavior of a specific mechanism of interaction. Figure 2-7 illustrates (in an extremely simplistic manner) how an α notation of a mechanism of interaction for workflow management might look like. The notation in the example stipulates and mediates a specific routing of documents between various tasks that, in turn, are assigned to specific organizational roles. In order to function, the α notation assumes that the abstract types (in casu, document types, task types, and agent types) are ‘instantiated’ in the sense that they are locally and temporarily made to designate particular documents, activities, and persons.

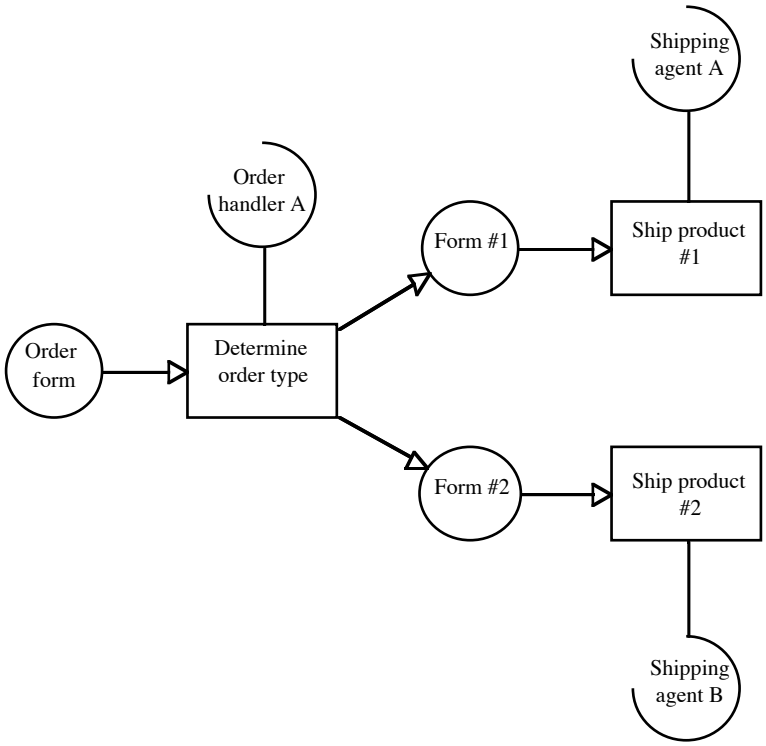


Figure 2-7. The α notation of a (extremely simplistic) mechanism of interaction for workflow management. The notation in the example stipulates and mediates a specific routing of documents between various tasks that, in turn, are assigned to specific organizational roles. — In the example given here, boxes designate task types, circles designate document types, and wedges designate agent types.

Facilities for re-designing the α notation will have to provide an appropriate notation for the specification of the new α notation, that is, a β notation.

The β notation is a notation enabling users to re-design the α notation, by re-combining the pre-specified β notation primitives (e.g., organizational roles, document types, task types) in different ways, according to the syntactic rules of the β notation (Figure 2-8).

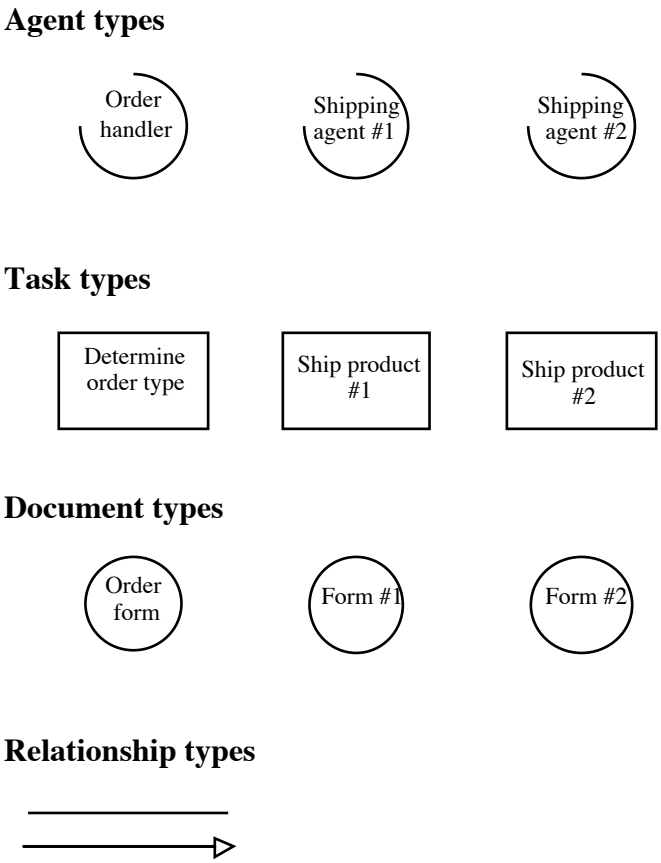


Figure 2-8. The β notation for re-designing the α notation of Figure 2-7. The β notation provides a set of primitives that can be re-combined in various ways, according to the syntactic and semantic rules of the β notation.

Now, no mechanism of interaction can apply to all aspects of articulation work; mechanisms of interaction address and will surely always address specific aspects of articulation work. One application may for example focus on providing support for scheduling, another may focus on providing support for work flows, yet another may focus on providing support for classification, and so forth. They are "local and temporary closures". That is, in their cooperative effort actors will have to deal with multiple mechanisms of interaction (incorporated in different applications).

In order not to impose artificial distinctions and thus disrupt the ongoing articulation work, it should be possible to link the different mechanisms, locally and temporarily, to get the work done. Therefore, a generic notation for specifying domain-specific β notations is required.

This third notation, the γ notation, is a notation enabling users to change the β notation, by adding primitives to the β notation (e.g., new primitives such as organizational roles, document types, task types) or by changing the syntactic and semantic rules governing the behavior of the β-primitives (Figure 2-9).

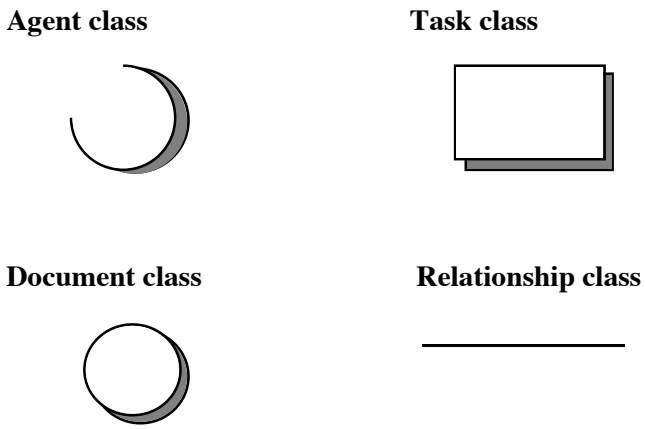


Figure 2-9. The γ notation for re-specifying the β notation of Figure 2-8. The γ notation provides generic classes of primitives pertinent to articulation work that can be defined according to a set of attributes so as to specify the β-primitives and their behavior. — The term 'class' is used here in the normal sense of a aggregation of phenomena with certain pertinent properties in common.

As an ultimate goal, the different mechanisms should be constructed by means of an abstract notation specified for each dimension so as to support genuine flexibility across applications (Malone et al., 1992).

In other words, the ultimate aim is to develop a γ notation at an operating system level that can be used a generic building blocks for all types of β notations and thereby for all types of mechanisms of interaction (in the form of α notations) (Figure 2-10).

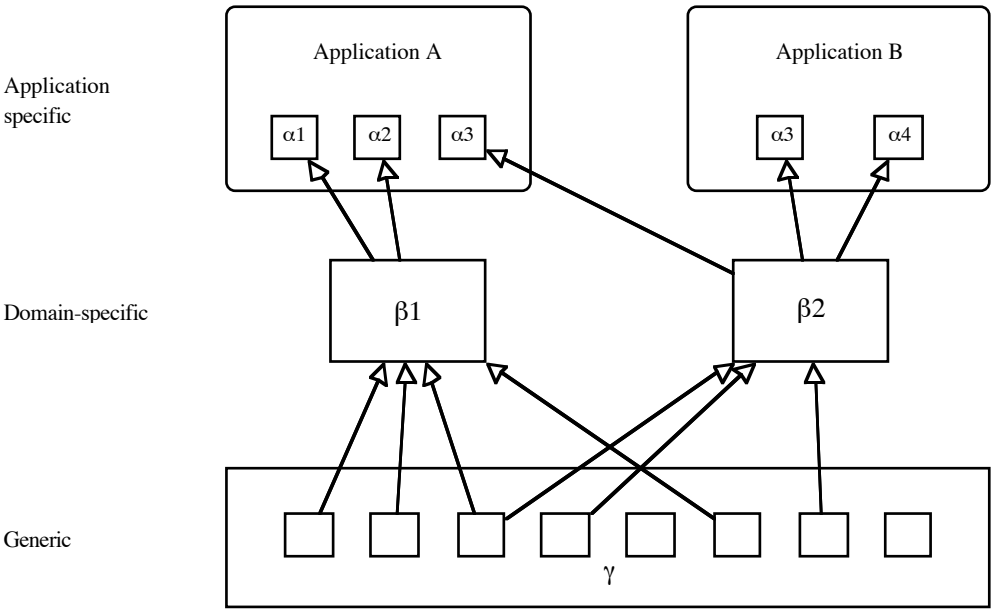


Figure 2-10. The γ notation should be conceived of as a comprehensive notation (at the semantic level of articulation work) for specifying domain-specific β notations and, in turn, the application-specific α notations of the mechanisms of interactions incorporated in different applications. Hence, the γ notation is an extension of the operating system. The exact location of the β notation in a CSCW systems architecture is not yet certain. They can usefully but vaguely be thought of as domain-specific 'add-on' or 'plug-in' extensions of the operating system.

Whether it is possible to develop a γ notation that can specify any β notation and thereby specify the behavior of any mechanism of interaction is still an open question.

2.4. Semantic scope of computational mechanisms of interaction

Articulation work can be conceived of as the overhead activities required by the fact that multiple relatively autonomous decision makers are involved in cooperative work and that their distributed activities must be articulated (coordinated, meshed) in terms of who, what, where, when, how, with which result, by means of what, and so forth (cf. Part 1).

Articulation of cooperative work is thus required with respect to multiple dimensions:

(1) Articulation in terms of **actors**, that is, the actual or potential participants in the cooperative effort whose cooperative activities are being articulated (in different capacities such as roles, jobs, individuals, collectives): Which partners are potentially relevant for a particular project in terms of skills, competing commitments etc.? Who are available when?

(2) Articulation in terms of **tasks**, that is, in terms of an operational intention (goals to attain, obligations and commitments to meet): What is the prob-

lem? What is to be done? Who should do it? Should I do it? Which task is (normally, advisably, or according to statute) to be undertaken in which circumstances, by which actor, based on what information and which criteria, creating what information? What is the (normal, advisable, or statutory) relation between tasks (procedure, workflow)?

(3) Articulation in terms of **activities**, that is, in terms of an unfolding course of purposive action. What are the others doing, and why? What have they done, what will they be doing, etc.? Do they cope?[1]

(4) Articulation in terms of **conceptual structures**, that is, in terms of the relationship between categories used within a specific community as ordering devices: definition, classification, prototypical, causal, genetic, historical, means/end relations.

Articulation in terms of common **resources**:

- **information** resources (documents, letters, applications, notes, files, memos, reports, drawings): Which actor can access, change, delete, copy which information resources? To which actor is the object to be displayed, in which format? Which actor can see who doing what to which objects?
- **material** resources (materials, components, assemblies). Which materials, components, assemblies are available where, when, how, in which quantity? What are their characteristics?
- **technical** resources (tools, fixtures, machinery, software applications). What are their operational characteristics (machining tolerance, suitability for different kinds of materials and material dimensions, processing time and cost)?
- **infrastructural** resources (rooms, buildings, communication facilities, transportation facilities). What are their operational characteristics (capacity, location, compatibility, turnaround time, bandwith)?

Furthermore, articulation work is never done in the abstract, it is always done in relation to and in terms of a wider context:

- the demands and constraints as posed by the **work environment**;
- the state of **field of work**;
- the wider **organizational setting**.

And, finally, articulation work is, of course, done with reference to abstract systems of reference: **time** and **space**.

---

1 The terminology used here comes from the Scandinavian tradition of systems devlopment (Andersen et al., 1990): An *activity* is used to denote a work process as an unfolding course of action, but only those aspects of a work process that are relevant to doing the work with the currently available resources, not all other incidents that may occur in the same course of action but which are of no consequence for getting the work done (like spilling coffee). — The concept of a task, on the other hand, is used to denote an operational intent, irrespective of how it is implemented. A task is expressed in terms of *what*, an activity in terms of *how*. A task can be *accomplished*, an activity can *cease*.

Articulation work with respect to these dimensions (in terms of objects and wider frames of reference) involves a multitude of *operations*. For example, an actor may point to an indicator in order attract attention to a particular aspect of the state of the field of work; an actor may accept or reject a task or simply undertake it; an actor may countermand an order, tacitly or overtly; an actor may make certain activities publicly visible by leaving traces, embedding cues, speaking aloud and so on.

Objects of articulation work	*Elemental operations with respect to objects of articulation work (examples)*
Actors	enroll A, assign B, reserve C; move D, place E
Responsibilities	allocate, assume, volunteer; hand over; accept, reject;
Tasks	point out, express; divide, relate; allocate, assume, volunteer; accept, reject; order, countermand; accomplish, assess; approve, disapprove;
Activities	do; make publicly perceptible, monitor, be aware of; explain, question;
Conceptual structures	define, classify, instantiate, relate, exemplify; posit, accept, challenge;
Informational resources	copy to A, move from B, transfer; access, block; read, interpret, relate;
Material resources	consume, move from A, place near B; name, characterize; procure, deploy, reserve;
Technical resources	use, move, place; name, characterize; procure, deploy, reserve;
Infrastructural resources	name, characterize; reserve;

External systems of references of articulation work	
Work environment: Demands and constraints	define, interpret, challenge
Field of work: State of affairs	direct attention to, make sense of
Organizational setting	define, navigate;
Space	refer to coordinates in…
Time	refer to points in…

Figure 2-11. Elemental operations with respect to objects of articulation work (examples). The purpose of the Figure 2-is *not* to anticipate the definition of primitives that will eventually transpire in the course of the project, but to indicate, by way of example, what would be the *appropriate semantic level* for the primitives of a notation of computational mechanisms of interaction.[1]

The primitives of a notation of a mechanism of interaction must be defined with respect to these dimensions of articulation work and the concomitant operations. However, these dimensions of articulation work are interdependent and the different operations of articulation work are thus themselves to be meshed in a

1 A similar attempt to identify "basic coordination processes" is made by Malone and Crowston (1992).

fluent way. A notation should support the fluid meshing of articulation with respect to different dimensions.

As far as the semantic scope of a notation is concerned, it is reasonable to discuss requirements with respect to semantic level and comprehensiveness:

1. **Semantic level**. The primitives of a notation of a mechanism of interaction must be expressed at the semantic level of articulation work, that is, at a level corresponding to the level *exemplified* by the table in Figure 2-11.

2. **Comprehensiveness**. As noted above, different CSCW applications will provide support for articulation work with respect to different dimensions of articulation work. One application may for example focus on providing support for scheduling, another may focus on providing support for workflows and yet another focus on providing support for classification, and so forth. So, when analyzing or designing an computational notation (explicit or implicit), it is necessary to determine its 'application domain' within the wider 'domain' of articulation work.

2.4.1. Semantic level

For a computational mechanism of interaction to be useful for supporting articulation work, the primitives of the notation must be expressed at the semantic level of articulation work, that is in terms of operations of articulation work with respect to actors, tasks, activities, conceptual structures, resources, the field of work, and so on.

That is, the components and manipulations provided by the mechanism should directly correspond to notions and objects of everyday articulation work. A high degree of transparency is required.

These requirements go both for the actual notation addressed during the daily execution (use) of the mechanism (the α notation), for the β notation used when re-specifying the structures and procedures embedded in the mechanism, and the γ notation for changing the specification of the primitives and rules of the β notation.

When characterizing computational notations of mechanisms of interactions with respect to their semantic level, the following questions are relevant:

- Which primitives are available when interacting via the mechanism during actual work?
- Which descriptive notation is available to visualize the α notaion?
- Are the primitives of the α notation expressed in terms of articulation work, i.e., do the primitives reflect actual objects and operations of articulation work?
- Can the user perceive the α notation (primitives, syntactic and semantic rules, etc.) of the mechanism of interaction in its entirety?
- If the user can change (re-specify) the notation (cf. the discussion of malleability), which primitives are introduced at the β notation level?

- How does the β notation expressed in terms of articulation work?

In order to have a common framework of reference, it is useful to compare the set of dimensions suggested in Section 2.4 with the set of user interface components used in different applications.

To illustrate — and further discuss — the relevant aspects of the semantic level we will briefly discuss a few examples taken from some existing CSCW-applications.

ACTIVE MEMORY (Denizon, 1990) is a calendar and planning application that provides general facilities for defining activities and deadlines, and distributes these among group members. ACTIVE MEMORY handles a "memory" (a plan file) for each user/actor. Here planned tasks, activities, etc. can be defined by means of types of activities, contents, periods of time, and participants. Each activity can be prioritized. Furthermore, deadlines for the activities and the frequency for repeating the deadline warnings (at the receiver's side) can be defined. The users themselves define how and when reminders are activated. Each activity is shown and visualized as a line in a spread-sheet like table (the planning panel). The primitives manipulated are then actors, calendars, and activities (tasks) including a description, a priority, a deadline, etc. These primitives reflects the dimensions that the articulation work would deal with regarding planning and project follow up. Intuitively we could argue that the primitives of the α notation seems to be at a proper level. The main procedure primitives (functions) in ACTIVE MEMORY are edit message, send message, accept message and refuse message. To some extent they also seems to be on a proper semantic level since they partly reflects "typical" activities in the articulation work. There are however a number of other relevant procedure primitives missing, e.g., return a "don't know"-answer, or get an overview of status of a particular activity. Regarding the β notation it simply does not exist in ACTIVE MEMORY. The structures and rules implemented in ACTIVE MEMORY cannot be changed.

A much more complex system is OVAL (FRY ET AL., 1992; MALONE ET AL., 1992). OVAL is a 'proof of concept' demonstration system rather than a finished application. Its main purpose is to show how the basic building blocks in the system work together to enable integration of many types of information and applications, and at the same time provide users with the ability to tailor this integration to their actual needs. The system allows users to create and modify cooperative work applications for themselves without requiring the help of professional programmers. The system aims to allow users to keep track of and share knowledge about messages, people, tasks, projects, companies, meetings and other things with which they work. Users can create various kinds of programmable actors to help them organize and respond to this knowledge. They can also use programmable actors to find electronic messages in which they are interested, to notice overdue tasks, and to notify people of coming deadlines.

OVAL is based on four key elements to be used in creating a variety of customizable applications: 1) Objects that are template based data sheets ordered in a hierarchical manner containing fields that can be manipulated in several ways, 2)

Views containing semi-editable tables. Views can be customized to display various kinds of information in any object field, 3) Actors which are rule-based programs. The user can program a set of production-rules (IF-THEN rules) to automatically perform actions upon series of objects at specified times or events, and 4) Links representing hypertext like relations between objects. Knowledge represented in links can be used by rules and in creating displays.

The objects, views, actors and links, and relevant manipulation on these constitute a very general notation that can be used for constructing a wide variety of different applications. This indicates, that OVAL can be used for developing a computational mechanism of interaction which embeds an α notation at a proper level regarding the actual objects (dimensions) in the field of work and the environment, i.e., a proper level in relation to the actual objects in the articulation work. To discuss the semantic level of the α notation, we can say that this is totally depending on the actual application built. Regarding the β notation (primitives and procedures on these) for designing an α notation with a proper semantic level, we must claim that these intuitively do not seem to fit (be transparent with) the typical level of articulation work; objects views and links are at a rather primitive level and only corresponds to typical objects in the field of work and the environment if they are combined into much more complex structures.

When discussing semantic level and OVAL it might be relevant also to discuss NOTES. NOTES supports cooperation within a "virtual" workgroup (cf. Section 3.2). NOTES includes facilities such as access controls, free text fields, and calculated fields. Considering the semantic level, the templates in NOTES are equivalent to object types in OVAL; the databases in NOTES are equivalent to folders in OVAL; and the views in NOTES are equivalent to views of collections of objects in OVAL. Even though NOTES allows very knowledgeable users to do certain kinds of sorting and filtering using views, it does not appear to be designed to make this easy for end users, and it does not have active actors like those in OVAL. So, even though we can build mechanisms with a proper α notation — which address the concepts the user are directly interacting with — we again can argue that the β notation used for designing the α notation is commonly adopted from information systems based on database systems. The notation does not cover primitives at a proper semantic level.

2.4.2. Comprehensiveness

A mechanism of interaction can be more or less comprehensive in the sense that it can support the articulation of cooperative work along one, several, or all of the dimensions of articulation work. Thus, a mechanism of interaction can only be considered totally comprehensive (or general) if it can support the articulation of cooperative work along all the given dimensions. However, depending on the application domain not all of these dimensions will be necessary, but it is crucial to be able to build computational mechanisms using any combination of the di-

mensions, or the usefulness of the concept of a mechanism of interaction is diminished.

The table in Figure 2-12 below show the different dimensions of articulation work and the CSCW systems that support articulation work with respect to some of these dimensions.

Dimensions of Articulation Work

Applications	Tasks	Activities	Actors	Categories
Coordinator	■		■	
Active Memory	■		■	■	■
WooRKS/ UTUCS	■		■	
Domino	■		■	
Aspects		■	■		■
gIBIS			■	■	
Answer Garden				■	

Figure 2-12. The comprehensiveness of a number of CSCW systems in terms of the dimensions of articulation work supported.

The rest of this section is a discussion of the listed applications in terms of the dimensions they cover. Each of the applications is described in detailed in Part 3 of this report.

THE COORDINATOR. THE COORDINATOR is a groupware product which is based on the conversation paradigm (cf. Section 3.2). The only way users can coordinate their activities is through conversations, and THE COORDINATOR provides support for creating and maintaining conversations. It provides an Agenda by which the users can organize commitments and conversations in progress. The available dimensions of articulation work within coordinator are Tasks and Actors. The mechanism of interaction incorporated into THE COORDINATOR induces an articulation of work that makes reference to a predefined structure for conversations i.e., the α notation is explicit. Actors are a dimension of articulation work supported by THE COORDINATOR as any user knows about other users and

can have conversations with them. In this case the Actors are the users and there are no other sorts of Actors. THE COORDINATOR deals with the articulation of Tasks as it focuses on assigning Tasks to Actors and monitoring these Tasks (the articulation of work is supported by explicit relationships between task and actors). THE COORDINATOR is not articulated along the dimensions of Activities, as there is no support for the Actors actually doing anything such as shared editing, nor are there any Informational Resources such as shared documents and there is no way that the Actors can categorize their conversations, i.e. there is no support for articulation of conceptual structures.

ACTIVE MEMORY. ACTIVE MEMORY (cf. Section 3.3) is a calendar and planning application. It provides general facilities for defining activities and deadlines, and distributes these among group members. The dimensions of articulation embedded within ACTIVE MEMORY are Tasks, Actors, Informational Resources and Conceptual Structures. Everything that needs to be defined within Active Memory is an 'Activity'. This must not be confused with activity as a dimension of articulation, so 'Activity' in quotes is an Active Memory 'Activity', and activity without quotes is a dimension of articulation work. The users identify tasks to be done and develops an 'Activity' for that task, which can specify who to send it to (groups or individuals), what its priority is, deadlines, what needs to be done, who is to do it etc. Each users who accepts the 'Activity', has it added to their 'Plan'.

ACTIVE MEMORY articulates along the dimensions of Tasks and Actors. Actors can publish information, such as documents which become part of a 'shared space', and if the publisher makes any alterations to the document, then this is automatically sent to those who have subscribed to this document. ACTIVE MEMORY therefore articulates along the dimension of Informational Resources. Initially, ACTIVE MEMORY contains several predefined shared Categories, however each person can redefine these categories or develop new ones. This redefinition or development of new categories is done individually, so it is debatable whether ACTIVE MEMORY actual supports the articulation of work along the dimension of Conceptual Structures. ACTIVE MEMORY does not support the dimension of activity, as there is no way of, for example, generating a new 'Activity' jointly, although an 'Activity' can be altered by suggestions from receivers of this 'Activity'.

ASPECTS. ASPECTS (cf. Section 3.3) is a simultaneous conferencing application. It will allow users to share the writing, drawing and painting activities necessary for creating and modifying documents.

The dimensions of articulation that ASPECTS incorporates are Activities, Actors, and Informational Resources. The Activities within ASPECTS are the actual doing of the writing and drawing within the shared documents. If a document belonging to a user is needed for the conference, it can be distributed to other participants in the conference. This document is therefore being used as an Informational Resource to articulate the work. An ASPECTS conference maintains a list of the conference participants, so all participants know who is in the

conference, therefore ASPECTS is articulated along the dimension of Actors. As ASPECTS is a very general tool for writing and drawing, there is no support for any particular cooperative tasks, therefore ASPECTS does not support the articulation of work along the Task dimension.

WooRKS and **UTUCS**. UTUCS (cf. Section 3.2) is a framework for the development of workflow applications with the addition of communication functionality. A workflow is a plan of the tasks to be done, when and by whom, and contains the relationships between Tasks and the conditions for resolution of the Tasks. The communication facility offers a way to handle exceptions that can arise during procedure execution of the workflow. When a breakdown blocks the planned workflow, a conversation on that subject, among the involved group of people, can be developed within UTUCS.

UTUCS supports the articulation of work on the dimensions of Tasks and Actors, as the workflow stipulates both tasks and the people who are to do the Tasks. UTUCS does not support the articulation of work on the dimensions of informational resources or activities, as there is no support for the sharing of documents or support for the actual activities necessary for the completion of a task, such as developing an invoice.

DOMINO. DOMINO (cf. Section 3.2) is an office procedure system for modeling and monitoring structured office processes in organizations. Cooperative office processes are modeled by coordination procedures that describe the information flow necessary to accomplish a common task within a group of persons. A coordination procedure is performed action by action, and is carried out by various actors, who in the performance of that action, take on a particular role. DOMINO mediates and controls communication related to tasks by notifying the participants about actions due, by providing them with the information needed, and by routing the results of such actions to the responsible parties.

DOMINO supports the articulation of work along the dimension of Tasks, as does UTUCS, by providing the ability to assign people to tasks, and the conditions under which these tasks can be performed and the relationships between tasks. This can be mediated by an email like communication facility. In DOMINO we have the Actors who are involved in the tasks, however these Actors are not the actual people as in UTUCS, but are the roles that these people can take on in the performance of each task. As in UTUCS there is no support for articulation along the dimensions of Informational Resources or Activities.

ANSWERGARDEN. The ANSWERGARDEN (cf. Section 3.4) is a general hypertext system intended to support the development of an organizational memory. It can be considered as a mechanism of interaction in some situations, for example, if it is being used by a project group, but it can also be used to support groups of people who are not involved in cooperative work. It consists of a net of questions and answers which grows as people input questions and experts answer them.

The only dimension of articulation supported by ANSWERGARDEN is Conceptual Structures. For example a user can ask questions about a subset i.e., a category of the net such as 'What can I do to resolve problems with the print spooler mechanism?' This might lead them to several lower level questions and answers. Users can also 'jump' into the net wherever they want. The only articulation that is done (and possibly needed) is via questions and answers which are categorized. The people using this system are not Actors in the terms of dimensions of articulation as it is not possible to find out from ANSWERGARDEN who asked the questions and who answered them. Similarly there are no documents acting as Informational Resources and no specific support for tasks.

gIBIS. gIBIS (cf. Section 3.4) is a system based on the principle that the design process is a conversation between a number of protagonists each holding a specific viewpoint. The model focuses on the Key Issues. Actors can hold different positions on these issues and the positions can have arguments associated with them. There is only a limited set of legal moves. In this way, different trees are built up for each issue. gIBIS supports simultaneous access and updating of the issue groups by multiple users on a common local area network.

The different trees, corresponding to different issues articulate the cooperative work of the users along the dimensions of Conceptual Structures, as each tree is a separate category. The different nodes in the tree are connected by arcs such as, 'Questions suggested by', thus linking the design group into the network of trees. Therefore there is support for who has asked/answered a question. So the articulation work is also supported along the dimensions of Actors, not like in ANSWERGARDEN, where there are no people within the 'garden', only fruitful questions and answers. There are also no documents acting as Informational Resources and no specific support for Tasks.

CSCW Environments. Apart from the applications, it is also relevant to discuss how some of the environments, platforms, etc., designed for specifying CSCW-applications, supports the requirement of comprehensiveness. We will briefly discuss three of these.

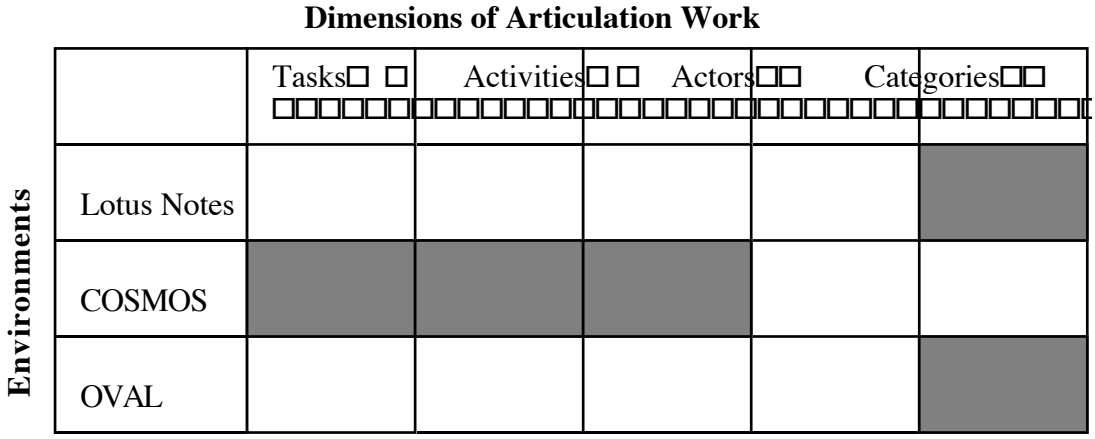


Figure 2-13. The comprehensiveness of environments, platforms, etc., designed for specifying CSCW-applications.

The scope of LOTUS NOTES (cf. Section 3.2) is to support cooperation within a virtual workgroup, i.e. geographically distant people with different knowledge, whose structure and members evolve in time. This is done through offering functional primitives for disseminating information, routing information, and manipulations on shared databases on the server. NOTES databases differ slightly in structure and terminology from traditional ones. A database consists of any number of records called notes. A note is a general structure holding a variety of information. Users can classify documents by categories, link them within the databases, and use filters to access or modify them. Further a number of facilities supporting the use of specific forms on the documents and facilities for viewing the information in a specific perspective are embedded in the environment. Regarding support of the above discussed dimensions of articulation work, NOTES only provides primitives for information resources. From a support for articulation point of view, NOTES is first of all a database environment supporting document sharing of different sorts. Although it might be possible to build applications addressing tasks, activities, etc. by use of NOTES the primitives offered are not very comprehensive.

The COSMOS (a COnfigurable Structured Message Oriented System) environment regards cooperative work as organized around practices which involve the coordination of different actions by different participants in the production of some entity. These working practices are represented as Communication Structures (CSs) described in a formal definition language (SDL). A CS is an abstract representation of a working practice. The COSMOS environment contains libraries of CSs which can be instantiated. An instantiated CS is referred to as an activity in COSMOS. The perspective on cooperative work is an extension of the language/action perspectives. The CS contains actions, rules, roles, objects, conditions, and temporal order constraints. A more detailed analysis of COSMOS can be seen in Section 3.5. So, regarding the support of the identified dimensions of comprehensiveness the COSMOS environment

addresses more of these than NOTES. Primitives reflecting actors, tasks, and activities are included.

A third environment for application building worth mentioning is OVAL. The main purpose of OVAL is to provide basic building blocks for integration of many types of information and applications, and at the same time provide users with the ability to tailor this integration. Users can create and modify cooperative work applications for themselves without requiring the help of professionals, e.g. setup 'programmable agents' to help them organize and respond to knowledge, find electronic messages, notice overdue tasks, and notify on deadlines. OVAL is based on four key elements: 1) Objects (template based data sheets), 2) Views (customizeable semi-editable tables), 3) Agents (rule-based programs), and 4) Links between objects. These elements plus manipulation of them constitutes a very general notation that can be used for constructing a wide variety of different applications. So, of identified dimensions of articulation work only the informational resources are directly addressed. As discussed later in this section, the primitives in OVAL are at a too low semantic level for articulation but can be used for specifying applications addressing actors, tasks, etc.

2.5. Facilities supporting access and manipulation of computational mechanisms of interaction

In addition to the requirements in terms of semantic scope discussed in the previous section, a mechanism of interaction must provide a set of facilities supporting actors in accessing and manipulating the notation.

This section discusses the requirements for such facilities — with illustrations from existing CSCW systems.

As argued previously, a mechanism of interaction incorporated in a computer-system should provide the following facilities:

(1) A mechanism of interaction should be 'malleable' in the sense that it can be managed by the cooperating ensemble itself. That is, the mechanism of interaction should be visible to the members of the ensemble and amenable to modifications by workers while in the process of work.

(2) The primitives of the notation should be expressed at a semantic level appropriate to the work context, that is, the primitives should be expressed in terms of basic concepts pertaining to the articulation of distributed activities in cooperative work.

(3) Since the system should support cooperative management of the mechanism of interaction, the system should support multiple users in modifying the mechanism of interaction while being immersed in the very flow of distributed activities. This raises a host of problems such as control of propagation of changes and management of (in)consistency.

To specify this further, we have identified the following facilities: User control of the execution, Malleability, Visibility, Propagation of changes, History, Support of maintenance of organizational context, and Openness. Each of these will be discussed in turn in the following.

It should be noticed that the facilities are in no way distinct categories. Their respective contributions to the support for access and manipulation are extremely intertwined. So, the facilities should, at this stage of the project, be seen as a tentative frame of reference.

Each of the following sections discuss one of the facilities. In each of the sections we first discuss the core aspects of the facility and then illustrate these aspects by means of examples from particular applications.

2.5.1. Control of execution

The next relevant facility to address when discussing computational mechanisms of interaction — and notations for these — is the level of control the user has, when executing and using the mechanism, i.e., to what extent is the user in control of making local and temporary changes in the structure or rules of a mechanism and how is this user control supported by the mechanism itself. Local and temporary changes could typically be actions like:

- suspend some of the parameters influence on the mechanisms interpretation of the embedded rules.
- overrule some of the embedded rules and procedures.
- rewind (or undo) a procedure.
- escape the actual situation
- restart the mechanism from another point, i.e., understanding the mechanism in terms of states and transitions, this means continue from another state.
- etc.

Control of the execution is then a question of the actors possibilities for temporarily taking over control of the structure and functionality of the mechanism and then afterwards bringing the mechanism “back on track” in a well-defined state. It might be that the expected space of possibilities must be abandoned and a new ‘state’ has to be reached by means of an unforeseen/unplanned sequence of actions. A typical example of such situations are exception handling occurring in workflow execution. In other words, is it possible for the actor to identify a state, maybe a totally unforeseen one, leading by means of suitable actions to a consistent one from which the ‘normal’ evolution can be started again? If an explicit process is absent, the only information concerning the dynamic behavior of the mechanism, that can be recovered, is a sort of trace. A sequence of the past events, possibly augmented by some information about the context, the motivations and the like. Therefore, recovering a consistent state might mean either

resetting or aborting the sequence, or to try a roll-back along the trace to find an intermediate consistent state.

Modifications to the notation (i.e., structures and procedures) — even temporary ones — can cause generation of a new (part of the) artifact, that is not compatible with its environment. For example, when a workflow modification alters the sequence in which the expected results are produced, or changes their format so that they cannot be interpreted and used. The check of compatibility is closely related to aspects of user/actor control. This is further addressed in Section 3.8.

As mentioned the user in control of execution only addresses temporary changes to the mechanism (the notation) in a specific situation. In terms of the α- and β notations (cf. Section 2.4) this means temporarily changing the α notation. Another important facility is the possibility of making permanent changes to the α notation by use of a β notation. This is addressed separately in Section 3.4 on malleability.

The aspect of controllability is, as most of the other aspects, closely related to several other aspects discussed in this section, especially those addressing the semantic level, malleability, and visibility.

When characterizing computational mechanisms of interactions regarding user control of execution, the following questions becomes essential:

- Which control of changes does a user have?
- Which controlling actions is offered to the actor and what are the consequences of using them?
- How are the actions offered to the actor? E.g. by opportunities to add, delete, or redefine primitives, syntactic rules and semantic rules, or by offering manual control of the mechanism, etc.
- Can a user circumvent the stipulations of the mechanism gracefully, i.e., without abandoning the services provided by the system?
- How is the adoption of the mechanism managed for a particular situation?
- Does the mechanism provide means for supporting cooperative management of changes to the mechanism (e.g., for maintaining consistency or dealing with inconsistency)?
- etc.

Actor controllability of the mechanism can typically be identified as either:

(1) No control, i.e., the user has no possibilities or support for taking over the actual control. There is no way for the user to temporarily stop, change, rewind, restart, etc. the mechanism. The preprogrammed functionality of mechanism will be executed.

(2) User partly in control, i.e., the user can in some situations control some aspects of how the mechanism is executed.

(3) Total control, i.e., the user can at any given time overrule or change the structures and procedures embedded in the mechanism. It is the user who decides when and in which state the mechanism can be restarted.

To exemplify the previous discussion on controllability of computational mechanisms of interaction lets briefly go through a few examples:

THE COORDINATOR (cf. Section 3.2) is an example of a mechanisms of which the user has no control of the execution. It embodies the principles of the conversation paradigm, based on the universal vocabulary of Speech Acts and provides its users with support of cooperation through conversation management. The notation used for describing how the conversations introduce conventions in the communication is based on finite automata. Each conversation is composed of single messages that reflect the taxonomy of the speech-act theory. The conversation is totally pre-structured and there are no possibilities offered to the user for temporarily changing it.

Examples of CSCW-systems in which the users have partial control are ASPECTS (Group Technologies Inc., 1990) is a simultaneous conference application. The moderator of a conference can during a session change the mediation level of the conference, i.e., the moderator is in partial (although not very much) control of how the mechanism is functioning.

The objective of UTUCS is to support the development of workflow applications, where communication offers a way to handle exceptions that can arise during procedure execution. Each application is defined by a set of activities and their relations in terms of input/output and control flow. These relations can be redesigned by the application engineer, but are not accessible to users. This means that the actor cannot have control over these during execution. There is however a partial solution to this. The actor can start a conversation about actions that have not been included in the procedure definition and to later link these to the procedure, and during the execution of a procedure the actor can decide to suspend the procedure. So, some control is in the hands of the user/actor.

An example of a notation offering some control to the user is the COSMOS environment (cf. Section 3.5). The Communication Structures (CS) offers the possibility that users may be supported during run-time in editing fragments of structure definition. A special mechanism (called SPACE for Structure PeripherAl Communication Environment) is provided for this purpose. In SPACE, an action or an action sequence can be declared as “done” if it is no longer relevant. Similarly, a condition can be forced to be true or false. So, the fact that users may have to depart from the structure provided by a CS to handle the exigencies of situations as they unfold at run-time is recognized and handled explicitly.

Only few examples (if any) exist in which the user has — at least in specific situations — total control of the mechanism of interaction. A good example is the Kanban system (described in detail in Part 1) although one could argue that the actors do not have total control. The kanban system is not a computer based system, but it is, however, a good example of a system in which the user can choose to overrule the mechanism. The kanban system is usually used in manufacturing for controlling the production. A set of “kanban-cards” act as the coordination mechanism, both as carrier of information about the state of affairs and the pro-

duction order conveying an instruction to initiate certain activities. In theory the kanban does not provide facilities allowing the actors to anticipate disturbances and to obtain an overview of the situation. However, in organizations using the kanban system the actors continuously change the configuration of the mechanism in many ways, for example by pocketing a card for a time or leaving the card on a fork-lift truck. The actors can then, temporarily overrule or change the structures and procedures of the notation.

2.5.2. Malleability

In addition to the required user control of the execution of a mechanism of interaction in terms of facilities supporting local and temporary changes to the behavior of the mechanism of interaction, users will require facilities supporting the design and re-design of the notation, for example re-designing a workflow. That is, the mechanism of interaction must be *malleable* in the sense that it supports users in making *global and permanent changes* to the notation.

The malleability of a notation of a mechanism of interaction can be conceived of in infinite variety, from complete absence of malleability to complete malleability. There is, of course, an inescapable trade-off between 'flexibility' and 'utility', i.e., between the malleability of a mechanism of interaction and the support it is able to provide in terms of reducing the complexity of articulation work. The obvious approach to this dilemma is to conceive of the notation of the mechanism as a multiple level architecture.

Software is, of course, always malleable in the sense that the program can be re-programmed. In this context, however, the requirement of malleability assumes that the behavior of the mechanism of interaction can be controlled at the semantic level of articulation work, in a notation that makes sense to the actors.

The above discussion of the α, β, and γ notations (Section 2.3) provides a way to address the requirement of malleability. The β notation is a notation enabling users to re-design the α notation, by re-combining the pre-specified β notation primitives (e.g., organizational roles, document types, task types) in different ways, according to the syntactic rules of the β notation, whereas the γ notation, is a notation enabling users to change the β notation, by adding primitives to the β notation (e.g., new primitives such as organizational roles, document types, task types) or by changing the syntactic and semantic rules governing the behavior of the β-primitives.

With respect to articulation work, the α-, β-, and γ notations are at the same semantic level. They are not at different levels of implementation but at different levels of specificity. By contrast, the notations provided by most existing CSCW-application environments such as OVAL, AOCE, and others are at another, 'lower', semantic level

DOMINO (cf. Section 3.2) is a good example of a workflow management system that supports the design and re-design of the α notation.

In DOMINO, the notation used to design the cooperative office procedure is the specification language CoPlan-S, that allows for the specification of alternative and concurrent courses of action in the form of workflows. An office procedure specification has three parts: the declaration of the roles, the declaration of the forms, and the description of the information flow, which is a list of statements describing the input-output behavior of the actions. Procedures are initiated and monitored by the procedure coordinator exclusively communicating with the participants' user components.

The information flow of a coordination procedure in DOMINO can be represented graphically (Figure 2-14): actions are represented by boxes, and channels by circles. The input/output relation between actions and forms is represented by arrows connecting the respective boxes and circles. Notches in action boxes indicate that these actions produce alternative sets of forms and thus contain a decision.

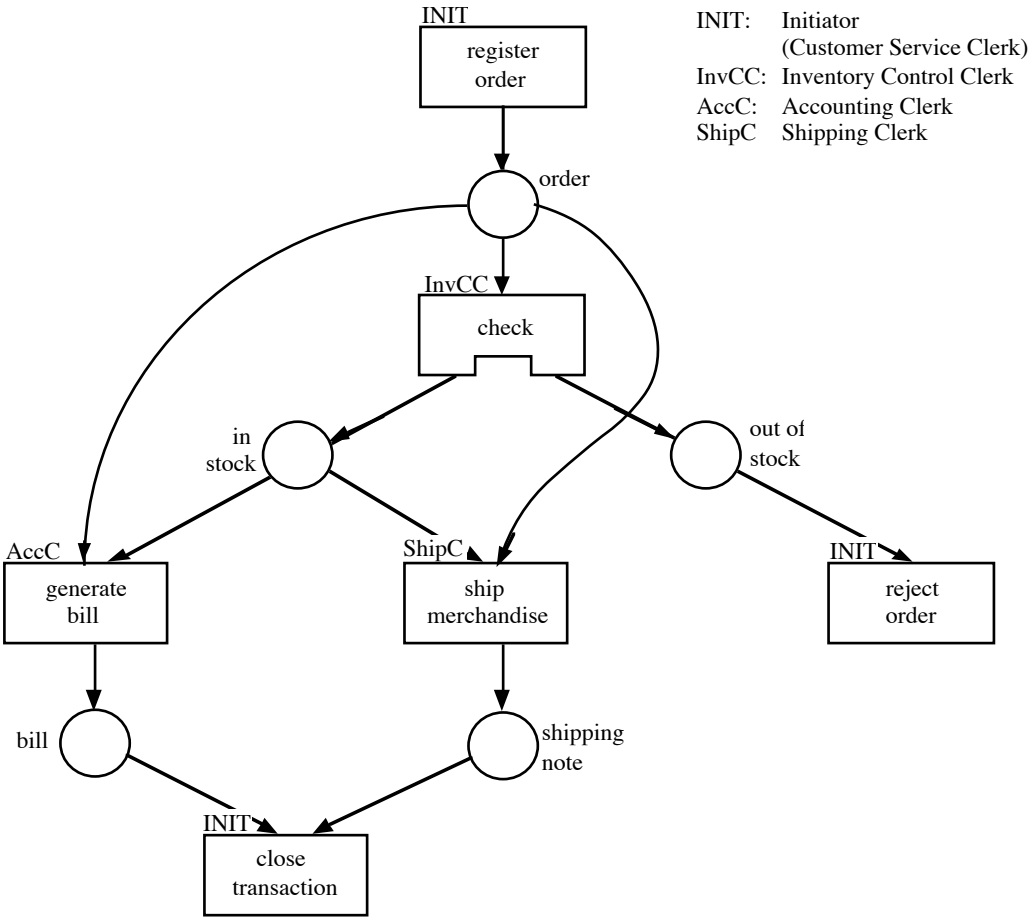


Figure 2-14. Example of a coordination procedure in DOMINO: Order processing

In DOMINO the β notation cannot be changed by end users but can have additions made by means of the γ notation by programmers when the system is brought down. The mechanism of interaction embedded in DOMINO thus supports an α- and β notation but not a γ notation.

UTUCS (cf. Section 3.2) supports some degree of malleability to the user. UTUCS supports a group of actors, interconnected in a communication network, in handling the conversations carried on by means of different communication media: e-mail, face-to-face and group meeting. The basic item is the conversation. UTUCS is developed in WooRKS (cf. Section 3.2) which supports development of workflow applications including exception handling. WooRKS applications are defined by a set of activities and their relations in terms of input/output and control flow. The relations can be re-designed by the application engineer, but are not accessible to users. UTUCS provides a partial solution of this problem. Users can define a conversation covering actions that have not been included in the procedure definition and later link these to the procedure.

An example of a notation offering malleability to the user is the COSMOS environment (cf. Section 3.5). In the COSMOS environment, working practices are represented as Communication Structures (CSs) using a formal language called SDL (Structure Definition Language). A CS is an abstract representation of a working practice, independent of any particular occasion of use, and the COSMOS environment is regarded as containing libraries of CSs which can be instantiated in the support of cooperative work. An instantiated CS is referred to as an activity in COSMOS. The intention of COSMOS was that SDL should be available to any user of the system, i.e., any user could author CSs in SDL, register them in the COSMOS environment, and instantiate them for use. As such, the overall functionality of the COSMOS environment is in principle fully end-user configurable (malleable). How this exactly would work in practice is, however, unclear.

2.5.3. Visibility

Because computational mechanisms of interaction support aspects of articulation work it is important to assess the need for visibility. The actors using a computational mechanism of interaction must be able to assess both the status of the mechanism in order to decide on further courses of action, as well as consult the notation of the mechanism when needing to modify it. Visibility relates, in principle, to all the objects of articulation work (cf. Section 2.4) supported by the mechanism, e.g., tasks, activities, actors, conceptual structures, and resources.

The visibility of a computational mechanism of interaction has two aspects: The visibility of the content of the mechanism and the visibility of the notation of the mechanism — the α and β levels discussed in Section 2. One of the basic aspects of visibility is whether or not the effect of an option is visible to the user, before and after its invocation, i.e., the visibility of the functionality offered, and the local consequences of invoking functionality. Another aspect is the visibility of the underlying notation — can the user perceive the notation of the mechanism of interaction in its entirety, e.g., primitives, syntactic rules, semantic rules? If yes, which β notation is used to display the α notation?

The issue of visibility of the notation — explicit versus implicit notations — is crucial in order to manage the problem of keeping consistency and governing propagation of changes. In our opinion, the disadvantage of having an explicit "process" embodied in the artifact can be balanced by the advantage of having a stronger support for managing changes made to the artifact.

In absence of an explicit process, the only information concerning the dynamic behavior of the system that can be recovered is a sort of trace, a sequence of the past events, possibly augmented by some information about the context, the motivations and the like. Therefore, recovering a consistent state means either resetting or aborting the sequence, or to try rolling-back along the trace to find an intermediate consistent state.

Having the process explicit means being able to 'see' the whole space of possibilities of the different behaviors. In this case, it is possible not only to apply the just mentioned strategies, but also to identify a consistent state different from the ones the system went through during its evolution. In other words, is it possible to identify a state, maybe a totally unforeseen one, leading by means of suitable actions to a consistent one from which the 'normal' evolution can be started again. We are not claiming this becomes an easy job but just a more flexible and supported way to manage modifications that maintain consistency.

The fact that we promote the usefulness of an explicit process does not mean that it should always be presented to the user. The α notation can adopt different modalities of human computer interaction: e.g., THE COORDINATOR (cf. Section 3.2) does not present the full automaton in its interface but just the next possible moves. The crucial thing is that the process should be available to the user, explicitly, if and when this is useful for solving problems. In the same vein, a full description of a procedure can be hidden in favor of the presentation of just the next possible actions but it should be made available in the case of the recovery procedures or modifications mentioned above. The two types of presentation are likely to meet two different requirements: a 'short-sighted' view that avoids uninteresting information when the path is safe; a 'far-sighted' view that provides a more global information when the situation is critical.

Different actors might have different needs for visibility in different situations. When articulation work is progressing according to the plan, one scope can be optimal. Here, the underlying structure and notation of the mechanism would perhaps only create confusion. In exceptional situations when a mechanism needs to be redefined or re-programmed, the user must, for example, be able to get an overview of the status of the whole system in order to decide on proper action to be taken, e.g., through performing "what-if" experiments, simulations, in order to assess the overall consequences of actions. Simulation requires a fully unambiguous semantics of both the meta notation and the language used to describe the mechanisms of interaction. Moreover, it is quite well recognized that a user-oriented simulation requires a graphical presentation consistent with the syntax and the semantics of the 'linear' language. This type of correspondence is used in the coordination module COP of WooRKS (cf. Section 3.2), where the linear lan-

guage ADL expressed in a BNF-like notation can be represented also in a Petri net based language. This double representation is, however, only adopted for the purpose of achieving performance in the implementation and not in order to provide simulation capabilities. Käkölä (1993) is using Petri nets in order to describe Role Interaction Nets (RIN). RIN aims at supporting the coordination of tasks and roles within a domain with respect to temporal ordering and degree of concurrency.

AnswerGarden (cf. Section 3.4) is a good example of how the underlying structure of the mechanism can be visualized in a hypertext graph. The visibility of the mechanism helps users to navigate the structure. Similarly, Information Control Net (ICN) which is a technique focusing on modeling office work analyzed in (Carstensen and Schmidt, 1993), provides a semantic net of organizational functions and processes (cf. Figure 2-15).

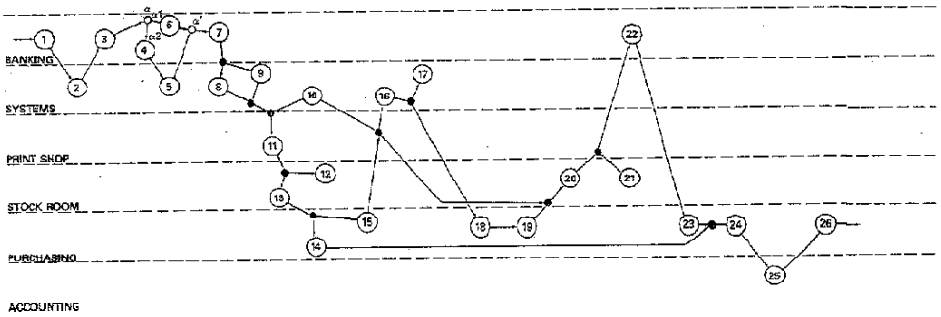


Figure 2-15: An example of a Information Control Net (from Cook, presented in COMIC-Risø-2-1).

In ASPECTS (cf. Section 3.3), the primitives of the notation, i.e., the names of participants, is visible to all participants in a session. The participants can also see who is editing, who is not editing, and who is waiting to edit. Changes in the mediation level, for example from having no access protocol to either a turn-taking protocol or having a single moderator, are however not visible if the participants are not aware at the moment of change. Furthermore, the moderator is not visible to participants in a moderated session. This information is not shown, for example, in the list of conference participants.

2.5.4. Governing propagation of changes

If computational mechanisms of interaction are to be easily adopted in the cooperative work arrangement they must, at least to some extent, be malleable (cf. Section 2.5.2). This implies that the users can adapt mechanisms of interaction to particular work situations with respect to both the content of the mechanism (at the α level), as well as changing the definition of the underlying protocol (the β-level). The ability to make distributed changes, both on- and off-line, raises the very important question of how to propagate the changes. There are two aspects of the propagation of changes: 1) The support of maintenance of consistency,

problems related to (in)consistency checks and propagation of changes in consequence of modifications of the mechanism or of its use; and 2) the user-control of propagation of changes, i.e., to what extent has the user control over the extent and the consequences of the propagation of changes.

Propagation of changes can be interpreted as a continuum between a simple notification of the modification to other users, to a chain of changes triggered from an initial one — a problem obviously more difficult to solve. Due to the intrinsic distributed nature of CSCW applications, it is very likely that a modification has impact on other system components. This implies that changes made to either the content or the definition of the mechanism of interaction requires awareness of consequences for the environment. This awareness can only be acquired by reasoning about the process as a whole.

The notions of (in)consistency and propagation of changes deserve some clarification. Modifications can cause inconsistency in two ways: a) generating different images of the same artifact in different actors' work-space: a typical example is the time table owned by the different actors of the Underground Control Room (Heath and Luff, 1992); or b) generating a new (part of the) artifact which is not compatible with its environment: for example, when a workflow modification alters the sequence in which the expected results are produced, or changes their format so that they cannot be interpreted and used. The problem of consistency between different images, is reduced by the use of the current technology. This implies that the problem of latency between the changes and the distributed updates is not having sufficient bandwidth. This relates to the visibility facility: how are propagated changes made visible, e.g., in real-time or on request?

The difficulty of keeping consistency and governing propagation of changes are problems that affect all types of mechanisms of interaction. It is closely related to the degree of central control embedded in the artifacts. On the one hand, if they only embody little central control, they allow more flexible usage as they delegate more handling of the articulation of work (the metawork) to the users. This implies that the metawork becomes the true source of complexity. On the other hand, if artifacts embody more control, flexibility has to be reached by handling exceptions 'within the mechanism'. This implies that the notations provided by the mechanisms must have sufficient means for redefining both content and definition of the mechanisms. Here, the complexity arises from handling the exceptions in a more formalized context. In both cases, keeping consistency and managing propagation of changes is something that has to be understood and adequately supported.

Regarding propagation of changes made to the content, or data, of a computational mechanism of interaction, the notion of *space of possibilities* is important. The space of possibilities is the set of legal states which the user can choose in any given situation. We have identified two cases: a) the same space of possibility of actions is maintained; or b) the expected space of possibility must be abandoned and a new state has to be reached by means of an unforeseen/unplanned sequence of actions. Both cases implies the necessity of being aware of how local

modifications impact more global parts of the system. This idea is usually captured by the notion of interface between local and global behaviors.

The Kanban system (cf. Part 1) is an example of a mechanism of interaction where the same space of possibility of actions is maintained: the activities on and triggered by this special card can tolerate some time delay, some mis-position, some overriding of the recorded data. The tolerance of CSCW applications concerning modifications on, for example, the ordering of actions in case of concurrency, or flexible priorities, is an important property. This class of modifications requires a notation able to identify/highlight the constraints within which the behavior can vary still keeping system operability: time constraints, data constraints and constraints concerning the behavior expected by other system components. In other words, the system's tolerance to perturbations.

A typical example of a situation where the expected space of possibility must be abandoned, is exception handling occurring in workflow execution. Part 1, amongst others, describes the unplanned (cooperative) actions necessary to recover missing documents and to ensure consistency of the information in a procedure of an Accounting Office. This class of modifications requires a notation able to support recovery procedures, either by describing predefined 'stable' states, or by supporting their construction. In many cases, recovery procedures imply propagation of changes, at least at the local data level, and more generally at the level of notification of changes to the interacting components.

When performing permanent off-line modifications to a mechanism, such as redesign of a workflow in order to improve its performance, or the redesign of an object to fulfill new requirements of the working context, the notion of interface becomes crucial to establish if and how the changes at the local level affect the interaction with other components. If changes have a global effect on the system, the activation of the modified version is only possible after performing other changes in order to maintain a stable system. Here, the notation should support detection of which components are affected and how, and furthermore support the identification of acceptable solutions guaranteeing the consistency. If permanent modifications are performed on-line it is necessary to identify a 'stable' state to which the new definition can be applied and where the consistency is guaranteed. In most CSCW applications the modifications are usually off-line and governed by a special role, e.g., 'the application engineer'.

A viable approach for evolving different strategies for handling consistency could be to rethink ideas from the field of CASE tools into the field of computational mechanisms of interaction. Vessey et al. (Vessey et al., 1992) characterize the following three strategies for handling consistency in specifications made on CASE tools: (1) Restrictive (2) guided, and (3) flexible CASE tools. The tripartite describes how the tool guides the developer in the process of making specifications. Restrictive tools encourage the developer to work in a top-down mode of operation. Using a bottom-up approach is impossible. Similarly, alternating between specifying on different levels of abstraction is not possible either. Guided tools suggests a top-down approach, but do not force this. At any give level, the

tool supplies information about the next level. Flexible tools support a free choice in mode of operation, and offer at any given stage the possibility of performing checks of the internal and hierarchical consistency of specifications.

The EGRET framework (Johnson, 1992) implements a flexible strategy with the three states: 1) The Consensual State which marks some consistency in the beginning of the process; 2) The Exploration State which will allow actors to make specifications that are inconsistent; and 3) The Consolidation State where divergences are analyzed and solved. The EGRET framework does, however, not provide computational mechanisms for negotiating inconsistency — the consolidation process.

Most of the CSCW applications reviewed are not modifiable, and those which are, implement very simple strategies for handling inconsistency. Very simple means of propagating changes are most often implemented. Consistency checks of compatibility, is essentially disregarded in all the reviewed CSCW applications. The applications analyzed provide, in general, either restrictive strategies in the sense that inconsistency is not allowed, as for example in ASPECTS (cf. Section 3.3), or they provide no strategy for handling inconsistency, i.e., the applications allow inconsistencies but does not provide check facilities. In ACTIVE MEMORY (Section 3.3), for example, both the predefined categories and the categories shared by the users can be redefined by each user, thus resulting in inconsistency. There are, however, no computer support for consistency checks, nor for recovery.

ANSWERGARDEN (Section 3.4) supports the development of organizational memory, and is only relevant to the field of mechanisms of interaction when used for articulation work within projects. The users can attach comments to the hypertext network. The network itself is managed by an expert, or moderator. When a user attaches a comment to a node, an email is automatically sent to the moderator. Based on the email messages received, the moderator chooses how the network should be changed. ANSWERGARDEN implements a very strict user-control over propagation of changes. The normal user can only suggest changes by commenting on the existing structure. This implies a very simple strategy for handling inconsistency, i.e., the consistency is put in the hands of a single moderator.

DOMINO (Section 3.2) models structured office procedures. The work plans, that are coordinated by actors, can be modified during execution. It is, however, only the initiator of a work plan that has the rights to modify the work plan. Inconsistencies are detected and handled by the applications' Procedure Control module. This module checks for: unintended cycles in the causal net, deadlocks, possibility of a successful termination of the work plan, and availability of resources.

The TASK MANAGER (Section 3.2) is an asynchronous, geographically distributed system supporting the coordination of work plans, tasks, and exchange of documents and messages. The content of work plans can, as in DOMINO, be modified during execution, and mechanisms for handling propagation of changes

is therefore needed. The mechanism chosen is, however, quite simple, the tool simply imposes a linear ordering on concurrent updates of task attributes, and hence resolves inconsistency.

The AMIGO Advanced project (cf. Section 3.5) tried to address the problems in how to propagate changes. The project developed a model of group communication, the AMIGO Activity Model (AAM), based on activities (defined as the intention or goal of a group) representing group communication processes. One of the components in the AAM, the Rule Component, specifies what should happen when changes occur. The project further states "It is relatively easy to define this positive case in the specification. The Component must, however, also define what is to happen if things go wrong, e.g. if time limits are not kept, if unknown Message Objects are received, if specified regulations are disregarded, etc. A set of default regulations for checking the state of an Activity against these situations and possible deadlocks may be supported" (Pankoke-Babatz, 1989). However, although they recognize the problem, the AMIGO project does not offer a solution.

2.5.5. History of mechanism evolution

The variability of working situations where mechanisms of interactions are used to articulate work requires control and malleability. For the users to effectively exploit these fundamental facilities they need to be provided with additional support to:

- fully understand the semantics of the mechanism of interaction under usage in order to apply the appropriate control for dealing with unexpected situations, and
- reconstruct the evolution of the mechanism in order to access and retrieve earlier versions that are possibly more adequate to the current situation or suitable starting points for the redesign of the mechanism of interaction.

To this aim the availability of the history of the evolution of the mechanism is a basic support. More specifically, as the mechanism of interaction can evolve at least in two aspects — its use and its notation — it is useful to record the history of the evolution of both of them. It is important to stress from the very beginning that the extent to which each aspect is sensible depends on the type of mechanism of interaction under consideration, basically because of the possibly overwhelming amount of information made available to the user. This requires a strong "control of the execution" by the user, i.e., a flexible, cooperatively definable, context dependent strategy of 'what has to be recorded when', and 'what has to be made directly available to whom'.

The analyzed applications show several examples of the history of use.

Conversation based applications like THE COORDINATOR, WooRKS-UTUCS, and CHAOS (cf. Section 3.2) maintain the history of the conversation steps that is accessible by the two interlocutors of the conversation. In the various cases each step can have associated with it various types of information pertinent to the time in which the step was performed (e.g., documents, references to the organizational

context or to the communication context that generated it). In this case, the history is an initial support for recording argumentation during negotiation. This capability is fully provided in gIBIS (Section 3.4) since both the dialogue structure (the 'rhetorical moves') and the classification of the move contents are specialized just for argumentation.

In the reviewed cases of the mechanism of interaction in the form of procedures or workflows, the history concerns their execution. For example, WooRKS applications (Section 3.2) hold the history of each procedure either under execution or already concluded together with temporal information, the net of activities that temporally preceded it and the list of all documents produced. Moreover from a conversation it is possible to retrieve the activity/procedure that generated it, and from a procedure it is possible to see all the conversations about it. When a conversation about an activity/procedure is opened, it is linked to the history of this latter, as well as to the documents that have been enclosed in the conversations.

AWMS (Section 3.2) graphically tracks any transaction in a given business process with options to see the historical performance of individual workflows and the historical profile of individual and group performances. This type of information is a valuable input for any decision about the next steps to be executed or about the necessity of redesigning the overall process.

The situation seems to be different in the case of history of changes to the notation. No applications explicitly supports (or mentions) this. Versioning, a problem that is so well known in software development, seems to be totally disregarded in the development of support of work articulation, as if its design and effectiveness were not equally fundamental.

Apart from the obvious case of maintaining the various versions of procedure or workflow design, a quite peculiar type of history could be constructed in the case of ANSWERGARDEN (Section 3.4) whose incorporated mechanism of interaction is implemented through a branching network of diagnostic questions about the work environment. In fact, if the system could provide a temporal/causal relationship among the various updates, it could make available the history of the evolution of the represented 'organizational memory'.

2.5.6. Support of organizational maintenance

Work articulation along all its dimensions happens in the 'wider context' of the work environment, field of work and organizational context. The articulation and the context have a mutual influence since articulation allows the performance of tasks and of role playing by the various actors whose accomplishment modifies the context where further articulation takes place. Consistency between mechanisms supporting articulation work and their context of use is extremely important. A major source of ineffectiveness in cooperation is the possible inconsistency between the work articulation and execution framework, and the organizational context. This is especially critical when artifacts are used to support either

the articulation of cooperative work or the organizational context or both — since they can be an obstacle to the social behavior reducing the above mentioned inconsistency.

The design of mechanisms of interaction supporting work articulation has therefore to incorporate a deep understanding of the mutual influence each mechanism has with its organizational context. This is required in order to guarantee the consistency of the functionalities. An example can be found in CHAOS (Section 3.2), where the accomplishment of commitments 'to do' and the agreement on commitments 'to be' are associated with the modification of the actor's profile in terms of skill and role, respectively. This piece of information becomes automatically available for subsequent work articulations along the dimension of the involved actors.

Another example is the access control model proposed in Shen and Dawan (Shen and Dawan, 1992) to link the access capability of the users to the dynamics of their role definition. Even if this application is more at the level of a CSCW development environment, the proposed model can be considered as an initial proposal of a link between mechanisms of interaction and the organizational context at the articulation work level as far as the dimension of the different types of resources are concerned.

2.5.7. Openness

Articulation work occurs in a specific work settings which determine the type of mechanism actually needed. Consequently, it is likely that users want to conFigure 2-frameworks for work articulation by combining mechanisms of interaction serving different purposes and having different characteristics. A very important requirement for this is that the mechanisms supporting articulation work preserve their capabilities under composition, i.e., it needs to be open with respect to combination of multiple mechanisms of interaction.

Present CSCW applications incorporate very few mechanisms that can be combined, and in all case their combination is governed by 'ad hoc' policies. For example, consider the applications combining workflows management and conversations for exception handling. As the existing computational mechanisms of interaction are far from providing, on their own, the desired facilities, their combination really has little to add.

At the present stage of the research, it is difficult to envisage how in general the combination of mechanisms of interaction can be handled. The facilities described in the previous sections have to be understood and defined in much further detail, and the embedded notations have to be identified, before it really makes sense to address the problems of combining mechanisms of interaction. However, bearing in mind the necessity of combining mechanisms of interaction while maintaining their effectiveness, from the very beginning it is fundamental to avoid notations that prevent any level of compositionality, especially in the aim

of envisaging an underlying architecture to support the interaction mechanisms for cooperative work.

2.6 Related Research

In the previous chapters we have discussed characteristics of mechanisms of interaction and of computational notations embedded in these. The analysis of the broad range of CSCW systems covered in Part 3 of this report, was concerned with establishing whether any of the functionality of articulation work was embedded within these systems and applications, and whether the notations (the α, β, and γ notations) used to model the articulation work and the cooperative work were appropriate. The analysis of the systems and environments served as an initial assessment for evaluating the concept of mechanisms of interaction and enabled us, together with the analysis of the field studies, to clarify, extend and make more precise what constitutes a mechanism of interaction. The notations developed within the analyzed systems and environments varied from those derived from a theoretical background of office modeling supporting formal office procedures (e.g., DOMINO) to those addressing interpersonal communication based of conversation theory (e.g., THE COORDINATOR).

Most of the notations embedded in the analyzed systems and environments are only applicable for a particular type of interaction, i.e., the embedded notations only addresses (supports or specifies) specific elements of articulation work or very narrow work domains. What we are particular interested in, in our work, is a more general notation for expressing articulation work at an appropriate level (a γ notation. In following this goal it is worthwhile to look at other projects, systems, and environments working along the same lines. In the rest of this section we will discuss the few attempts to develop CSCW environments, that seeks to provide flexible notations.

CONVERSATIONBUILDER. CONVERSATIONBUILDER (Kaplan et al., 1992b) was developed as a support tool for providing flexible active support for (collaborative) work activities. This flexibility is achieved by providing "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). CONVERSATIONBUILDER contains 'ActionSpace' windows; one for each conversation, which displays the state of that particular conversation. Users can be in many conversations simultaneously and switching between the different windows constitutes switching between the different conversations. ConversationBuilder also contains a shared editor that includes hypertext abilities and users can edit text and graphics to develop artifacts such as shared documents. Because CONVERSATIONBUILDER is an open environment, other tools such as compilers can be included. The users are able to browse through the system, and can find out how the various conversations they are involved in are related, show the relationship between the various artifacts and check the versions of a particular

artifact. Actions such as an utterance in a conversation, changes to text when editing, the addition of new hypertext nodes, etc. are immediately sent to all users. The system can also inform the users of who else is looking at a particular artifact, who is editing what, which participants in a conversation are actively working on it etc. Bogia et al. (1993) list following specific facilities of the CONVERSATIONBUILDER:

Openness:	The ability to extend the environment to particular collaboration models, incorporate new tools, media etc.
Flexibility:	The ability to dynamically adapt the environment to new or changing work practices.
Activeness:	The ability of the environment to 'understand' and provide support for ongoing collaborative activities.
Data:	Facilities which allow data to be shared, viewed in multiple contexts, explicitly related etc.

From this it can be seen that CONVERSATIONBUILDER provides support for articulation work along many different dimensions and covers to some degree most of the facilities of a mechanism of interaction we have discussed. CONVERSATIONBUILDER is basically based upon an understanding of both work, articulation 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). This is similar to our approach of seeing mechanisms of interaction as being separated from the field of work itself.

The openness of the CONVERSATIONBUILDER enables many different tasks to be supported, and that together with the ability to check the state of the system including the users, shows that it provides support for the articulation of the tasks. Do however notice, that the understanding of openness differs from our. In our terms openness regards support for combining multiple mechanisms of interaction. Furthermore, by providing a UIMS (User Interface Management System) for developing new interfaces, CONVERSATIONBUILDER seeks to support the building of new protocols (legal moves in a conversation) at a proper semantic level.

The CONVERSATIONBUILDER is based on a theory of conversation for action: "Our vision is based on the work of Winograd and Flores concerning the equivalence of language and action" (Kaplan et al., 1992b, p. 378)). A mechanism based upon obligations then is central. This implies that only domains characterized by command and control structures are supported. Hence, the notation is to some extent domain specific. This comment is very similar to our previous comments on THE COORDINATOR (cf. section 2.2). The CONVERSATIONBUILDER does however provide visibility and control of the mechanism.

To summarize, the CONVERSATIONBUILDER offers a flexible environment covering a number of relevant facilities and offering a set of primitives at an appropriate semantic level, but — opposite to our approach — their work has not yet addressed the problem of identifying the fundamental characteristics in articulation work. The notation is based on a theory of conversation and a number of aspects of articulation work are not addressed at all.

EGRET. Another environment for providing support of collaborative activities is the EGRET Framework (Johnson, 1992). EGRET aims to support what is termed 'exploratory group work', for example software engineering. This type of work is characterized by being dynamic and unpredictable (changes in the team, tasks that are not fully understood, group processes that change as the tasks are better understood and unfold, etc.). The work is exploratory and experimental. Introducing new forms of support into the work "may impact on the domain in unpredictable ways" (Johnson, 1992, p. 298).

> "Computer support for the exploratory definition and evolution of artifacts such as the design template structure requires addressing several fundamental issues. First, what is the appropriate structure and process model for defining such collaborative artifacts? Second, how can that model support changes in the appropriate structure of artifacts as the collaborative activity proceeds? Finally and perhaps most importantly, how can the exploratory definition and adaptation of artifacts be accomplished in a truly *collaborative* fashion?" (Johnson, 1992, p. 300, his italics)

As this quote shows, Johnson is aware of the need to support a dynamically changing environment, and EGRET does this by giving the users the ability to develop new 'schemas' which can include tasks, descriptions, names etc. For example schemas for a software engineering group could consist of a hierarchy of tasks, activities, roles for task activities, etc., and all members of the group would initially see the same schemas. However each member of the group can develop these schemas by adding new nodes which can depart from the default node schema in endless ways, i.e., members of a group can invent different solutions to problems. EGRET keeps track of the variation by providing the functionality for listing the deviations from each schema. This enables all group members to see, for example, the number and types of solutions to a particular problem.

Outside of EGRET they can then come to a consensus as to the appropriate solution to the problem (the consolidation phase). Once this has been achieved, EGRET can once again be made consistent such that the nodes and the node schemas match and all members of the group have new consistent schemas from which they can begin to depart again.

Local and temporary changes to the structure, and visualization of these, are supported by the notation in the EGRET Framework, i.e., a high degree of control of execution and visibility of changes are supported. After the consolidation phase, some of these changes can be made permanent, so some malleability is provided. When the users are changing the schemas and the nodes, the structure of EGRET is inconsistent. This was a deliberate policy, but after the consolidation phase, the schemas can be once again be made consistent. All the changes are

propagated throughout the database. Although EGRET was built specifically for design teams there is no support for the actual tasks of software design.

The main aim of the EGRET Framework is to support small groups of people working closely together on projects, where, although the main goal is known, the process of achieving it is exploratory and experimental. It is not aimed at explicitly providing support for the articulation aspects of the cooperative work, rather it provides support for the *result* of the cooperative exploration and gives the members of the group a common knowledge base on which to begin further exploration. By providing this kind of support some articulation work is supported as the users can see changes made by others. The EGRET Framework supports specific aspects of articulation, but — as for the CONVERSATIONBUILDER — the framework does not address the core aspects of what articulation work consists of, and only domains characterized by having specific structures (here consensus structures) are supported. The notation provided by the framework, then only supports some highly specialized domains. The EGRET Framework provides a notation at a appropriate semantic level, but the primitives are specialized in terms of schemas representing current state of consensus. The aim of our work is identification of a general notation for expressing articulation work at an appropriate semantic level, and for this purpose the approaches in EGRET are much too specialized.

OVAL. OVAL is described in Section 3.3, and the properties has been discussed several places in the previous sections. The main purpose of OVAL is to provide basic building blocks for integration of many types of information and applications, and at the same time provide users with the ability to tailor this integration. The basic ideas and concepts will not be described here. It is, however an important contribution regarding notations for specifying mechanisms of interaction. Opposite the notations in the CONVERSATIONBUILDER and the EGRET Framework the notation in OVAL (the specification language) is intended to be a general notation for expressing articulation work. The basic primitives of objects, views, agents, and links are very general. The notation in OVAL then become very flexible. The specification level is however very primitive, and the basic primitives provided by the notation do not correspond to objects in general articulation work, they rather reflects typical structures in a computer. The degree of comprehensiveness is low. Again, identifying the essential elements of articulation work has not been addressed, and is then, of course, not reflected in the notation.

To summarize this section we can say that no one has considered the articulation of distributed and complexly interdependent activities as the primary concern in the design of their systems and notations. The environments (and notations) reviewed seems to fall into two groups: (1) Some provides primitives at an appropriate semantic level and supports a fairly high degree of comprehensiveness. These are however typically domain specific. CONVERSATIONBUILDER is a good example. (2) Others are extremely flexible and provides a very general notation. These are at a very low semantic level and offers very little com-

prehensiveness. OVAL is the best example. None of them seems to have explicitly addressed characterizing articulation work in general, and their models and notations are usually based on theories of human conversation, office procedure modeling, etc., not on a theory, or understanding, of articulation work.

We feel that to develop mechanisms for interaction specifically for the support of articulation of distributed activities will provide a better conceptual framework within which to provide support for roles, tasks, actors, activities, etc. This requires, however, that the question of, what characterizes articulation work is explicitly addressed, i.e. the investigations regarding the content of the γ notation needs to be taken first. None of the above discussed attempts have done this.

2.7. Conclusions

The work reported on in this deliverable is an analysis of existing computational notations within CSCW systems (Part 3) and evaluation of them with regard to mechanisms of interaction. This together with input from the analysis of existing field studies (Part 1) has informed the continuing development of computational mechanisms of interaction: the notation, the features, the dimensions of articulation work to be supported, and the primitives of mechanisms of interaction.

In developing CSCW systems it is vital to formalize articulation work in order to know:

- how to determine the allocation of functionality between artifacts and people,
- what aspects of articulation can be supported via information technology,
- which aspects of the articulation can be made unnecessary due to different designs of cooperative working arrangements,
- what must be left in the hands of the cooperating agents.

The analysis of field studies and the analysis of a cross section of CSCW systems: applications, environments and the specification languages used within them, has led to a better understanding of what is meant by the articulation of cooperative work, and hence to the conceptual foundation for designing computational mechanisms of interaction.

Articulation is the overhead of any cooperative working arrangement. Depending on the situation and the type of cooperative work (See Part 1), some of the functionality of the articulation work is allocated to artifacts such as schedules, time tables, standardized office procedures etc. but in many situations the articulation is achieved through the interactions of the people involved in the cooperative work. By looking at existing CSCW systems and applications we realized that support for articulation work (using mechanisms of interaction or not), was not a primary concern in the design of these systems. However, some

support was achieved due to the necessity for support of particular cooperative work arrangements or in generic shared applications.

From these analyses and the initial hypothesis of mechanisms of interaction we have further developed and expanded the idea of mechanisms of interaction to enable us, in the future, to use them in designing support for the articulation of cooperative work within CSCW systems. A mechanism of interaction is a symbolic artifact, publicly perceptible and publicly perceivable and depending on the domain and the field of work, can articulate the cooperative work along many dimensions, such as: task, actors, activities, information resources etc. To enable mechanisms of interaction to effectively support articulation work they must contain facilities for accessing and manipulating them such as: making them visible, propagating the changes, modifying the mechanisms etc. and they must be at the correct semantic level for the support of articulation work within organizations. To specify and control the behavior of mechanisms of interaction we have developed the three notations (α, β, γ) which capture those aspects of articulation work pertinent to many different cooperative working arrangements.

To be able to build CSCW systems from the perspective of supporting articulation work by incorporating mechanisms of interaction into the design, we will continue to develop requirements; both of the facilities and for formalizing the notation, and consider the implications this will have on many cooperative working arrangements within many different fields of work.

2.7.1. Requirements for mechanisms of interaction

Facilities. In the previous chapters of this report we have discussed requirements for mechanisms of interaction regarding the facilities of malleability, control of execution, visibility, propagation of changes, history, organizational maintenance, and openness. In this sections we will briefly go through the most essential facilities and discuss them in terms of possible requirements derived from the work done so far. The requirements of the facilities have direct implication on the development of the notations used for describing and specifying the objects, rules and primitives (the α, β and γ notations) of mechanisms of interaction.

Before discussing the facilities a few general comments needs to be said: The facilities are discussed in very general terms. Further investigation is required on whether the listed facilities cover all central aspects, and on how the facilities are related to each other. Furthermore, each facility can be described in terms of a continuum from complete absence of the facility to complete presence. Finally, the requirements of mechanisms of interaction regarding the facilities will differ from domain to domain. In some work settings it might be necessary to have a high degree of a specific facility, whereas it could be completely irrelevant in other settings. The ones that have been identified so far as the most important are:

- Malleability and semantic level

  Support of malleability is perhaps the most essential facility. The requirements of malleability are due to the fact that actors now and then need to re-

design the structures and rules that constitute the mechanisms. They need to be able to re specify and adapt the mechanism of interaction to contingencies and changing requirements within the cooperative work environment.

To support the requirements of malleability, the notation, used for specifying the objects and rules (the β notation) has to aid domain experts (not programmers) in 'reprogramming' the mechanisms themselves. The β notation has to contain a set of primitives at the proper semantic level, i.e., primitives that reflects well-known conceptual structures in the domain. A useful definition of 'a proper semantic level' needs further investigations.

- Control of execution and visibility

  Even though the mechanisms are malleable, the situated character of most work situations requires temporary changes of the mechanisms done 'on the spot' by the actors. Furthermore the need for malleability and temporary changes introduces a requirement for visibility, both of the content of the mechanism and of the notation of the mechanism — the α and β notation.

  To make the consequences of temporary or permanent changes visible to the actors the notation must provide some kind of simulation capabilities. The actors need support for 'running' a simulated execution before implementing the changes in the mechanisms themselves. This of course requires an explicit representation of the syntax and semantics of the notation describing the execution (the α notation). In its turn, the capability of simulating behaviors relies upon both unambiguous semantics and modularity and levels of abstraction that allow for considering just the pertinent information.

- Propagation of changes

  Derived from the fact that temporary and permanent changes are needed, a number of requirements regarding how the changes are propagated arise.

  First of all, the propagation problem concerns how consistency, both local and global, is maintained. The mechanisms must contain consistency checking. How this reflects the requirements for the α and β notations needs further investigations. As mentioned above the actors needs to be able to see the whole state of the field of work and the work arrangement and view the consequences before applying them. Again, some simulation support is needed. Furthermore, the notation describing the components in the mechanisms (the α notation) needs to be explicit to ensure the (other) actors actually discover and reflect on the changes.

  The propagation problem also regards aspects of user-control of the propagation. The notations then needs to offer primitives supporting modeling of structures for controlling the consistency-checking and the propagation of the implemented changes.

Notations. In considering the notations we need to discuss the different types of support necessary, the primitives of mechanisms of interaction and begin to

show how these primitives can be combined into useful mechanisms for supporting articulation work. The notations for describing mechanisms of interaction were first introduced in Section 2.3. A mechanism of interaction is represented by three distinct notations due to the flexibility inherent in the articulation of many different sorts of cooperative work arrangements within many different domains, and the necessity for reducing the complexity of the articulation work. We need to be able to express the usage of a mechanism of interaction, the structure of it, build new mechanisms of interaction (the α notation) and develop new primitives of the β notation such as different types of roles, different processes, different informational resources etc. To do this we need the γ notation. The notations analyze and specify the underlying logic or grammar of the mechanisms of interaction. They are different abstractions and represent the degree of specificity of conceptual primitives.

A notation consists of a syntax, a semantics and a set of rules. The α notation is the most specific and is related to the field of work and the domain and describes the actual mechanism of interaction, such as a work flows; the β notation is domain specific and contains the primitives of the α notation and rules for combining them, and the γ notation is for defining new primitives of the β notation. The γ notation has to be at the level of operating system primitives, *but* at the semantic level of articulation work. The notations therefore represent structured information, the rules for manipulating them and the protocols for using them. These notations formalize the information, procedures, processes, constraints etc. The notation and its presentation to the user has to have an unambiguous and coherent semantics. It has to provide a means for manipulation of the objects and the rules: specifically, simple rules of combination of the syntactic and semantic construct; the possibility of identifying and circumscribing the scope of the modification and the possibility of checking the correctness of the manipulation with respect to its environment .

An important question for the future is whether it is possible to develop *a finite set of primitives* of the γ notation that is fully comprehensive and at the correct semantic level for all possible mechanisms of interaction. If this is possible then the concept of mechanisms of interaction is extremely flexible, as any conceivable mechanism of interaction can be developed at a later stage.

Representation of the notations. To use the notations in the design of mechanisms of interaction within CSCW systems, they must be represented by some formal description. We need to decide which is the most appropriate ‘language’ for each of the different notations. We could consider using state transition diagrams, Petri Nets and their developments (e.g., Diplans (Holt, 1988)), BNF type grammars, formal methods such as ‘Z’, algebraic approaches such as Milner’s Synchronous Calculus of Communicating Systems (SCCS), etc. We need to decide which is the best approach for each of the three notations, whether all three can be represented using one approach and whether we need to develop our own ‘language’ for formally specifying the three notations. Käkölä (1993) in his paper on embedded systems for understanding coordinated work has

developed a language called Role Interaction Net (RIN), based on organizational role theory and Petri Nets. A RIN consists of a set of concurrent roles and techniques from Petri nets and interactions between the roles can be specified. Another approach was taken by Aiello et al. (1984) for modeling the office structure. The office model is represented in terms of interacting agents where each agent is characterized by a structural component and the office functions that the agent is allowed to perform. The formalism can also represent information about access to resources, and the structure of communication and control within the office. These types of languages have potential for use as representations of mechanisms of interaction, as they go some way towards representing the interactions of people and roles.

As mentioned earlier, malleability is perhaps the most essential facility that needs to be supported and the α, β and γ notations must reflect this. This suggests that the way in which the γ notation is implemented is of fundamental importance in enabling new primitives of the β notation to be developed, and hence new mechanisms of interaction.[1]

A model initially recognized in the field of artificial intelligence as being useful and subsequently used in developing programming languages, is computational reflection. Computational reflection allows a computational system to alter its behavior, thus altering its structure and conversely, altering the structure of the system will change its behavior. This type of functionality is what is needed to provide support for malleability. The model of computational reflection together with the notion of 'meta-object protocol' has been incorporated into CLOS (The Common Lisp Object System) (Bobrow et al., 1988). CLOS is a language written in CLOS thus allowing the application programmer to extend it. Meta-objects are the objects, classes and methods of CLOS and the meta-object protocol is the rules for the behavior and interactions of these meta-objects. It specifies, among other things, the functions for controlling instance creation. The application programmer can change, for example, the methods for instance creation, thus allowing different types of instances to be created. As the meta-objects are objects within CLOS, the system is reflective. As argued by Dourish (1992), CLOS has relevance for the design of CSCW systems, and we suggest that it is conceivably possible in CLOS to implement primitives of the γ notation at the proper semantic level for articulation, thereby allowing us to build mechanisms of interaction; however this needs further research.

Also, important pointers for the development of formalisms for describing α, β, and γ notations might be found in the body of research in the knowledge representation field. One of the topics explored in this field is how to represent

---

1 The basic building blocks (the γ notation) for implementing support for articulation work will be operation system features. In fact, in the field of dialogue modelling and design, similar ideas have been suggested, e.g., when discussing future work in HCI, Hartson and Hix argues that "Interface support environments will become integrated into the operating system and hardware architecture." (Hartson and Hix, 1989, p. 58)

both static and dynamic aspects of tasks, e.g. in the DAIDA project (Jarke et al., 1992). Chung, Mylopoulos and others (Chung, 1991; Mylopoulos et al., 1992), presents a notation for representing functional and non-functional requirements in order to express goals and constraints, and in order to provide malleability. Lessons could be learned on how to design the γ-notation by investigating these formalisms for modelling tasks.

In addition, the relationships between tasks and subtasks, activities, roles, etc. are often time dependent, and the α notation has to be able to represent this. We also need to be able to express causal relationships within the α notation, for example, in the analysis of ASPECTS (Group Technologies Inc., 1990), we needed a notation that showed the order in which people could join or leave a conference.

We must however bear in mind that due to the intrinsic distributed nature of potential CSCW systems, the mechanisms of interaction have to handle both distributed actions and distributed states. The changes to notations made by many people, simultaneously or not, can only occur in real time if the system supports concurrency.

With regard to languages suitable for the description, specification, and analysis of computational mechanisms of interaction, a large body of formal calculi and models may be relevant. Some key features of mechanisms of interaction, however, challenge the functionality of many of these models in interesting ways, including aspects of information access and visibility, and of dynamic reconfigurability, prompted by user intervention or changes in the work setting. Of particular interest from this perspective is the π-calculus of Milner, Parrow, and Walker (Milner et al., 1992). This calculus has recently been proposed explicitly to support the modelling of systems with dynamically changing interconnection structure. Its suitability as a foundation for the description of MoI's will be investigated, in the first hand through case studies.

2.7.2. Prospects for the future

In the work discussed in this volume we have come some way towards the goal of supporting articulation work by developing the theory of mechanisms of interaction, the facilities and the computational notations suitable for different mechanisms of interaction. The main objective for next years' work is to further develop the concept of computational mechanisms of interaction, including developing the possible finite set of primitives of the γ notation, and consider the implications this will have on the future of CSCW systems and the impact on cooperative work within organizations. To be able to do this, the following topics have to be taken into consideration:

(1) From the development of computational mechanisms of interaction involved both the field studies and the analysis of CSCW systems and applications, we have seen that mechanisms of interaction can be used as a common framework for bridging the gap between the social

investigations in the field and the conceivable CSCW applications. This is an issue worth further exploration.

(2) In order to provide a proper conceptual foundation for the design of mechanisms of interaction for CSCW applications, further empirical and theorecial sociological research is required into the use of 'plans' in situated action, in particular the use of symbolic artifacts for stipulating and mediating articulation work.

(3) Any mechanisms of interaction, computational or social, are used within an organizational context. We will need to know for example, who, when and how has the right and responsibility of changing a procedure or the structure of an object, for the design of the mechanisms. The design of computational mechanisms of interaction is therefore very likely to include the notion of roles and other aspects of organizational policies. The implication this might have are worth considering and ties in with the work in Strand 1 of COMIC.

(4) As mentioned in Section 2.2, the notation of a mechanism of interaction can be explicit or implicit. The implicitness or explicitness of the notation will depend on the organizational setting, the field of work, the application and the current state of the task. In many of the CSCW systems studied, implicit notations are typical of the enactment level and the user only learns what can happen at certain stages within an application by experience: they cannot see the whole process. The developing notations for mechanisms of interaction will allow the implicitness or explicitness of a mechanism to be rapidly changed according to changing circumstances, and will also allow the users control over how explicit/implicit the mechanism is depending on what is being made visible at any one time, e.g. the whole work flow or just a particular task.

(5) In section 2.5 we considered the facilities of mechanisms of interaction as though they were independent but as we have begun to show, they are all highly interdependent and the relationships between them need further investigation.

(6) Section 2.4 listed the dimensions of comprehensiveness (the objects of articulation work) of mechanisms of interaction and gave some examples of the elemental operations on these dimensions. We need to consider whether this list incorporates all dimensions of articulation work, find a systematic way of developing the articulation operations with respect to these dimensions, and study the implications for the design of the notations.

(7) The facilities provided within mechanisms of interaction and the objects of articulation work are related in many ways and this relationship needs to be further explored.

(8) With the objective of identifying general features and requirements of languages suitable for the description, specification, and analysis of computational mechanisms of interaction further explorations are required. In fact, a large body of formal calculi and models may be relevant. Some key features of mechanisms of interaction, however, challenge the functionality of many of these models in interesting ways, including aspects of information access and visibility, and of dynamic reconfigurability, prompted by user intervention or changes in the work setting.

(9) The relation between mechanisms and applications on the one hand and the operating system on the other hand is critical for making the concept of mechanism of interaction into a operational design construct. A particular problem in this context, is the location of the domain-specific β notation in the CSCW systems architecture. The concept of domain is quite elusive, however, and needs further investigation.

2.8. References

Aiello, Luigia, Daniele Nardi, and Maurizio Panti: "Modeling the office structure: A first step towards the office expert system," in *Second ACM-SIGOA Conference on Office Information Systems, June 25-27, 1984, Toronto, Canada*, ed. by C. A. Ellis, ACM, New York, 1984, pp. 25-31.

Andersen, N. E., F. Kensing, J. Lundin, L. Mathiassen, A. Munk-Madsen, M. Rasbech, and P. Sørgaard: *Professional Systems Development. Experience, Ideas, and Action*, Prentice-Hall, Englewood Cliffs, New Jersey, 1990.

Bobrow, Daniel, Linda Demichchiel, Richard Gabriel, Sonya Keene, Gregor Kiczales, and David Moon: *Common Lisp Object System Specification*, X3J13, June, 1988. [X3J13 Document 88-002R]. - Cited in Paul Dourish: *Computational Reflection and CSCW Design*, Internal Report EPC-92-102, Rank Xerox Cambridge EuroPARC, Cambridge, UK, 1992.

Bogia, Douglas P., William J. Tolone, Celsina Bignoli, and Simon M. Kaplan: "Issues in the Design of Collaborative Systems: Lessons from ConversationBuilder," *Design of Computer Supported Cooperative Work and Groupware Systems. 12th International Workshop on "Informatics and Psychology", Schärding, Austria, June 1-3, 1993*, 1993. - Forthcoming.

Borufka, H. G., H. W. Kuhlmann, and P.J.W. Ten Hagen: "Dialogue cells: A method for defining interactions," *IEEE Computer Graphics Applications*, July 1982, pp. 25-33.

Carstensen, Peter, and Kjeld Schmidt: *The Procrustes Paradigm: A Critique of Computer Science Approaches to Work Analysis*, COMIC Working Paper, Risø National Laboratory, (version 2.0), May, 1993. [COMIC-Risø-2-1].

Chung, Lawrence: "Representation and Utilization of Non-Functional Requirements for Information System Design," *Advanced Information Systems Engineering, Third International Conference CAiSE '91, Trondheim, Norway*, edited by R. Andersen, J. A. Bubenko and A. Sølvberg, Springer-Verlag, 1991, pp. 5–30.

Denizon, Christine: *Active Memory*, ASD Software, Inc., Montclair, California, 1990.

Dourish, Paul: *Computational Reflection and CSCW Design*, Internal Report, Rank Xerox Cambridge EuroPARC, Cambridge, UK, 1992. [EPC-92-102].

Fry, Christopher, Kum-Yew Lai, and Thomas Malone: *Oval*, v. 1.1, Massachusetts Institute of Technology, Cambridge, Massachusetts, 1992.

Gaver, William W.: "The Affordances of Media Spaces for Collaboration," in *CSCW '92. Proceedings of the Conference on Computer-Supported Cooperative Work, Toronto, Canada, October 31 to November 4, 1992*, ed. by J. Turner and R. Kraut, ACM Press, New York, 1992, pp. 17-24.

Group Technologies Inc.: *Aspects — The First Simultaneous Conference Software for the Macintosh*, Group Technologies, Inc., Arlington, Virginia, 1990.

Gustavsson, Rune: *Mechanisms of Interaction (MoI) and Interfaces at Different Levels*, COMIC Working Paper, SICS, 30 March, 1993. [COMIC-SICS-3-2].

Hartson, H. Rex, and Deborah Hix: "Human-computer interface development: Concepts and systems for its management," *ACM Computing Surveys*, vol. 21, no. 1, March 1989, pp. 5-92.

Heath, Christian, and Paul Luff: "Collaboration and Control. Crisis Management and Multimedia Technology in London Underground Control Rooms," *Computer Supported Cooperative Work (CSCW). An International Journal*, vol. 1, no. 1-2, 1992, pp. 69-94.

Holt, Anatol: "Diplans: A New Language for the Study and Implementation of Coordination," *ACM Transactions on Office Information Systems*, vol. 6, no. 2, April 1988, pp. 109-125.

Holt, Anatol W.: "Coordination Technology and Petri Nets," in *Advances in Petri Nets 1985*, ed. by G. Rozenberg, Lecture Notes in Computer Science, ed. by G. Goos and J. Hartmanis, vol. 222, Springer-Verlag, Berlin, 1985, pp. 278-296.

Jarke, M, J. Mylopoulos, J. W. Schmidt, and Y. Vassiliou: "DAIDA: An Environment for Evolving Information Systems," *ACM Transactions on Information Systems*, vol. 10, no. 1, January 1992, pp. 1–50.

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*, ed. by J. Turner and R. Kraut, ACM Press, New York, 1992, pp. 298-305.

Käkölä, Timo: "Doing by understanding: embedded systems for understanding coordinated work," in *Human-Computer Interaction: Software and Hardware Interfaces. Proceedings of the Fifth International Conference on Human-Computer Interaction, (HCI International '93), Orlando, Florida, August 8-13, 1993*, ed. by G. Salvendy and M. J. Smith, vol. 2, Elsevier, Amsterdam, 1993, pp. 973-978.

Kaplan, Simon M., William J. Tolone, Elsa Bignoli, Douglas P. Bogia, and Alan M. Carroll: "Orthogonal Support Aids Collaborative Tasks," *Schärding '92,* , 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*, ed. by J. Turner and R. Kraut, ACM Press, New York, 1992b, pp. 378-385.

Kreifelts, Thomas, Elke Hinrichs, and Gerd Woetzel: "Sharing To-Do Lists with a Distributed Task Manager," in *ECSCW '93. Proceedings of the Third European Conference on Computer-Supported Cooperative Work, 13-17 September 1993, Milan, Italy*, ed. by G. De Michelis, C. Simone and K. Schmidt, Kluwer Academic Publishers, Dordrecht, 1993, pp. 31-46.

Malone, Thomas W., and Kevin Crowston: *The Interdisciplinary Study of Coordination*, Center of Coordination Science, MIT, 1992.

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*, ed. by J. Turner and R. Kraut, ACM Press, New York, 1992, pp. 289-297.

Milner, R., J. Parrow, and D. Walker: "A Calculus of Mobile Processes, I-II," *Information and Computation*, vol. 100, no. 1, 1992, pp. 1-40, 41-77.

Moran, Thomas P.: "The command language grammer. A representation of the user interface of interactive computer systems," *International Journal of Man-Machine Studies*, vol. 15, 1981, pp. 3-51.

Mylopoulos, John, Lawrence Chung, and Brian Nixon: "Representing and Using Nonfunctional Requirements: A Process-Oriented Approach," *IEEE Transactions on Software Engineering*, vol. 18, no. 6, June 1992, pp. 483–497.

Naur, Peter: *Computing: A Human Activity*, ACM Press, New York, 1992.

Pankoke-Babatz, Uta (ed.): *Computer Based Group Communication: The AMIGO Activity Model*, Ellis Horwood, Chichester, 1989.

Schmidt, Kjeld, and Tom Rodden: "Putting it all together: Requirements for a CSCW platform," *Design of Computer Supported Cooperative Work and Groupware Systems. 12th International Workshop on "Informatics and Psychology", Schärding, Austria, June 1-3, 1993*, 1993. - Forthcoming.

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*, ed. by J. Turner and R. Kraut, ACM Press, New York, 1992, pp. 51-58.

Vessey, I, S. L. Jarvenpaa, and N. Tractinsky: "Evaluation of Vendor Products: CASE Tools as Methodology Companions," *Communications of the ACM*, vol. 35, no. 4, April 1992, pp. 90-105.

Wasserman, A. I.: "Extending transition diagrams for the specification of human-computer interaction," *IEEE Transactions on Software Engineering*, vol. SE-11, no. 8, 1985.

Winograd, Terry, and Fernando Flores: *Understanding Computers and Cognition: A New Foundation for Design*, Ablex Publishing Corp., Norwood, New Jersey, 1986.

Part 3

Analysis of computational mechanisms of interaction in current CSCW systems

Table of Contents, Part 3

3.1. Introduction

In order to deal with computational notations for mechanisms of interaction in CSCW it is natural to start with an investigation of the existing CSCW applications to have a picture of what has been already achieved starting from an engineering point of view, i.e., a standpoint that is orthogonal to the one taken in the field case investigations, as it it biased by what can actually be implemented in computers.

The literature about CSCW contains several presentations of CSCW applications together with their comparison, classification and assessment. This large amount of elaboration is not directly usable for the purposes of our research project since its leading criteria does not explicitly focus on the basic notion of mechanisms of interaction as defined in Part 1 and on their features illustrated in Part 2. In fact, this specific standpoint requires distinction between tools supporting the execution of work and tools supporting the articulation of work, a distinction that requires an original way to look, present and discuss existing CSCW applications. This is the reason for the further analysis of CSCW applications contained in this section.

The above mentioned goal requires both selecting a meaningful set of applications and identifying a set of evaluation criteria sensible to deal with mechanisms of interaction and their notations.

As for the first requirement, the proposed selection exploits the experience and knowledge of the involved researchers and at the same time aims at covering the categories of applications that are significant from the point of view of the (notation for the) mechanisms of interaction they embody. This coverage is therefore not exhaustive but broad enough to serve as a basis for an initial assessment of what is achieved and a further elaboration of the topic (see Part 2).

More specifically, the selection contains applications that support different modalities of interaction, both synchronous and asynchronous, based either on sharing of data or on workflow management, or on a combination of the two. We considered them from the point of view of the mechanisms of interaction they incorporate. The various applications show different capabilities for supporting articulation work, both from the point of view of the specificity of the provided support and of the dimensions of articulation work they consider. The ultimate goal of this investigation is not to identify any original taxonomy for classifying applications, rather to extract from each of them what can be recognized as a mechanisms of interaction and to identify, where possible, the computational notations adopted to make them available to the users. For this reason, the structure of the presentation of the CSCW applications follows the contributions provided by the different research groups involved in WP 3.2, so that internal consistency, cross-refences and considerations are maintained and properly contextualized.

As for the choice of the criteria for analyzing the identified mechanisms of interaction, the leading idea is to obtain results that are not only valuable per se but especially meaningful in relation to one of the most important general issues mentioned in the Technical Annexe (page 41), namely to investigate the "apparent gap between the underspecified 'natural' notions of social mechanisms of interaction and computational notations which can be interpreted by computers". Therefore, the basic set of criteria is deeply influenced by the research performed in the field studies reported in Part 1. As mentioned in the Preface, the analysis of some of the existing CSCW applications was the first step of the research effort during the first year of the COMIC project. In order to make this effort possible a first agreement on the criteria to be adopted was necessary. To make this part of the deliverable self-consistent, listed below are the critieria for the analysis together with their definitions. The reader can perceive some slight inconsistencies with respect to the terminology and definitions used in the previous parts of the deliverable. This is due to the fact that, once the analysis had been accomplished, the criteria and their definitions were further elaborated into a set of facilities to be used in the identification of the basic requirements of the mechanisms of interaction to support work articulation and the notations required for making them usable in the course of work (see Part 2).

The criteria adopted during the analysis step are the following:

Comprehensiveness: concerns the dimensions of articulation work supported by the mechanism(s) of interaction incorporated in the considered application.

Malleability: concerns the degree of amenability to modification of the mechanisms of interaction by the users while in the process of work.

Visibility: concerns the extent to which the user can perceive the notation of the mechanism of interaction in its syntactic, semantic and compositional rules.

Propagation of changes: concerns the perception of the effects of the choice of the various options, especially during the modifications, and the extent to which the user can have (support in the) control on these effects.

Level of abstraction: concerns the adequateness of the basic primitives to express the actual features of articulation work.

Context maintained: concerns the extent to which the mechanisms of interaction maintain the organizational and cognitive context in which the articulation work takes place.

Degree of openness: concerns how the mechanisms of interaction are 'conceptually' and/or 'technologically' open to be integrated with mechanisms supporting other dimensions of the articulation work.

In the following, the considered CSCW applications are more or less presented along the following schema:

- a short genesis and description;
- the basic MOIs and their notations;
- the features of the mechanisms.

3.2. Coordinator, AWMS, DOMINO, UTUCS, WooRKS, CHAOS, Task Manager, and Lotus Notes

C. Simone, M. A. Grasso, A. Pozzoli

University of Milano

3.2.1. Introduction

This chapter presents an overview of a set of CSCW applications to identify and analyse the mechanisms of interaction (MOIs) they incorporate. These applications combine, in a more or less explicit way, a communication environment with an operational environment handling task definition and execution.

The considered applications can be partinioned into two parts. A first subset is identified by an explicit reference to the notion of 'conversation' and/or by a support to the workflow management. A conversation (in the sense proposed in [WF86]) is a structure of predefined speech acts exchanged by two interlocutors and is usually implemented on an asynchronous medium, basically an e-mail system. A workflow management system provides an integrated support to routine and non-routine work together with some exception handling mechanisms. Let us remark that, even if within workflows any model of cooperation can happen, the emphasis is in general on the asynchronous (in time and space) modality.

The systems belonging to the first subset are: The Coordinator, Action Workflow Management System, DOMINO, UTUCS-WooRKS, CHAOS and Task Manager. They differ in the way they represent tasks and their structure, in the way they integrate communication with the performance of tasks, and finally to what extent they maintain the organizational and cognitive context of communication and action.

The remaining applications are Lotus Notes and the proposal based on Sharing Processes. Both of them provide mechanisms for supporting work articulation in terms of tasks, agents and information resources coming from the opposite side: namely, instead to add something to communication, a communication support is added to the sharing of tasks and objects.

3.2.2. The Coordinator[1]

3.2.2.1. A short genesis and description

The Coordinator has been developed by Action Technologies [WF86] [AT88] as its first groupware product. It embodies the principles of the conversation paradigm, based on the universal vocabulary of Speech Acts (SA)[Sea69]. Its characteristic feature is to use bilateral conversational models.

The Coordinator provides its users with a support of cooperation through conversation management. The system addresses messages to specified people (both individuals and groups defined by individuals). Messages have a standard heading qualifying the type of SA and a portion of free text.

The management of conversations is realized by a set of tools provided to the users to create and maintain records of conversations. The system embodies an Agenda where the user can organize commitments and conversations in progress.

The Coordinator deals with articulation of tasks as it focuses on assigning and monitoring contract-based work activities.

3.2.2.2. The basic MOI

MOI to support communication

The MOI supplied for communication is the conversation. There are two predefined types of conversations: for action and for information. People build each message as a speech-act: specifically, they choose its illocutionary points from a set of alternatives and specify the propositional content. The receiver is defined upon the opening of a conversation, and is automatically recorded in subsequent message exchanges.

MOI to support action

The Agenda is the only support to record appointments, memos an conversations in progress. The only way users can coordinate their activities is through conversations, since there is no mechanism modeling the structure of an activity in terms of elementary actions. So there is not an explicit MOI for action. This limit is overcome by the Action Workflow Management System illustrated in the next section.

3.2.2.3. The Notation

The Coordinator uses a notation for modeling conversations and a notation for organizing information about conversations and, among them, the Agenda.

1) The notation used for describing how conversations introduce conventions in the communication is based on finite automata. Each

---

1 The Coordinator is a Trade Mark of Action Technology Inc.

conversation is composed of single messages that reflect the taxonomy of SA, as shown in Figure. 3.2-1.

Present State	Speech Act	Next State
()	A Offers	Offered
	P Requests	Requested
Requested	A Counter-Offers	Offered
	A Refuses	Rejected*
	A Accepts	Taken
	A Concludes	Concluded
	A Delegates	Delegated*
	P Modifies request	Requested
	P Withdraws	Deleted*
Offered	P Counter-Requests	Requested
	P Refuses	Rejected*
	P Accepts	Taken
	A Withdraws	Deleted*
Taken	P Counter-Requests	Requested
	P Cancels	Deleted*
	A Counter-Offers	Offered
	A Concludes	Concluded
Concluded	P Refuses	Taken
	P Evaluates	Satisfied*

* Terminal States; A = Active, P = Passive

Figure 3.2-1

2) The primitives available for managing the conversations can be derived from the main menu of The Coordinator:

Read	Compose	Agenda	Organize	File	Editing	Services	Quit	Help

They are quite self explanatory: specifically, Organize allows the user to define categories of conversations based on special keywords (e.g., the Partner, the Answer Time) to access them in a structured way; Agenda allows the user to insert new commitments and to manage their temporal conflicts.

3.2.2.4. The features of the mechanisms

Comprehensiveness

The structure of conversation for action and for information are quite general to handle negotiations for defining tasks and delegating their execution. However this type of conversation does not cover all the communicative situations: the proposers themselves have introduced [WF86] the notion of conversation for possibility, that is not implemented in the system. On the other hand, some aspects of work articulation are totally out of this approach, like for example any reference to the content of the communication or to its organizational context.

Malleability

The communication mechanism is not malleable, as it proposes either specific conversation models or an e-mail like ‘free’ conversation, and there is no possibility of defining new conversation structures.

Visibility

As for the notation, it is visible to the users as the finite automaton description shows the primitives, the syntactic and semantics rules. As for the visibility of the pragmatic effects, each operation performed to communicate and its effects on the communication environment are completely visible to the user. What is not visible is the pragmatic effect of communication, namely the actions and changes they generate in the activities and organizational context.

Level of abstraction

The conversation is expressed at a level of abstraction appropriate to the communication context, being the steps of the conversation based on speech-act theory.

Propagation of changes

Since the system is not amenable to modifications, there is no propagation of changes.

The context maintained

The functionalities for managing the conversations maintain the cognitive context of a message in terms of the sending and receiving of messages within the conversation it belongs to (the history of the conversation). Moreover the Agenda maintains the information concerning time and the timing relationships among on-going conversations and commitments registered by the user: this is implemented just as a visualization of the related information in the same frame of the Agenda.

The organizational context in terms of roles, rules, responsibilities or competences is totally disregarded. The system assumes that all this knowledge is kept by the user and the system does not help him in any way.

The degree of openness

The Coordinator generates text files which can be imported from the most common desktop publishing systems (Ventura, PageMaker, ...) using the ASCII format. Moreover, the Message Handling Service (MHS) used to deliver messages allows the communication with other applications, by setting some specific parameters through the MHS interface.

3.2.3. Action Workflow Management System

3.2.3.1. A short genesis and description

The pragmatic approach to communication underlying The Coordinator is the basis for a model of workflow that is implemented within the Action Workflow Management System [MWFF92],[Sey92]. AWMS shifts from traditional task structure to coordination structure. ...”this shift is analogous to moving from a view of a network as a collection of nodes to seeing it as a collection of links with shared nodes” ... .

The goal of AWMS is to track and monitor business processes (i.e. the work done by people in organizations) and workflows (i.e. the sequence of actions that happen in a business process), automating repetitive and cumbersome tasks.

Tasks are defined by requests and commitments expressed in the workflow loop (see §3.3). The workflow is composed of a set of different elements:

- People
- Procedures
- Information
- Tasks
- Management

The focus is on the activities for coordinating tasks more that on the tasks themselves. In fact, the structure of the workflow is defined by the language acts through which people coordinate, not by the activities that individuals must deal with.

Each workflow loop concerns a single task. Tasks can evolve in parallel or sequentially, as independent loops evolve simultaneously, while subordinate workflow loops must wait for the completion of subsidiary workflows.

Time is expressed through:

- deadlines
- follows-up
- time for completion

In order to improve task performance, AWMS aims at identifying, observing and anticipating potential breakdowns. Moreover, to keep processes moving along AWMS manages reminders, alerts and follows-up.

The system allows the definition of rules for routing documents, spreadsheets, forms, software programs, etc. among tasks.

One goal of the AWMS is to provide workflow capabilities to new and existing computer applications. The architecture of the AWMS allows interoperability among different applications and across diverse platforms, as shown in figure 3.2-2.

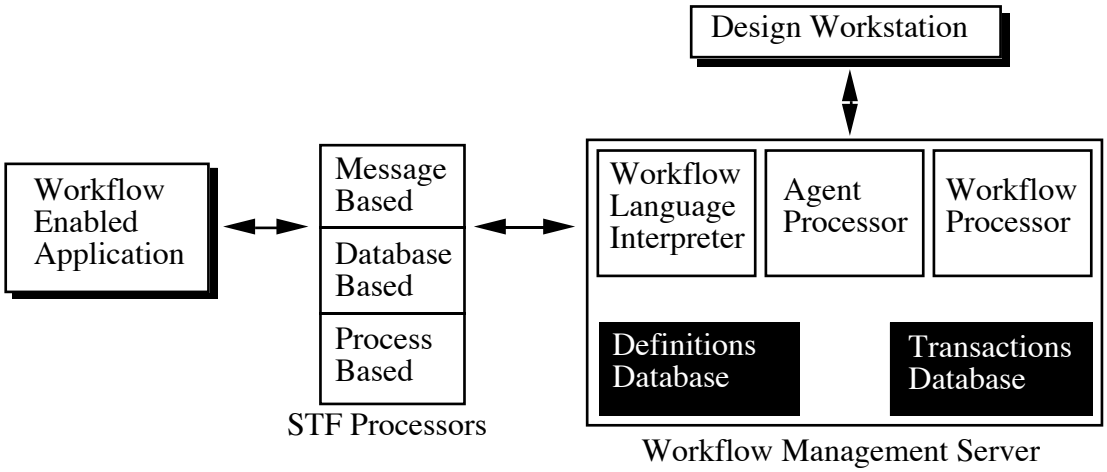


Figure 3.2-2. AWMS architecture

3.2.3.2. The basic MOI

MOI to support action

The MOI which supports action within AWMS is the business process that is made up of one or more workflows: in its turn, an organization is made up of one or more business processes. The business process is the articulation work done by people in organizations to make and fulfil offers to customers: other processes, as information and material processes, arise as a result of the more fundamental business process.

AWMS integrates existing or new computer applications; this possibility is referred to as *workflow-enabling*. This is done through the STF processors, which translate from the application native data format to the Standard Transaction Format of the Workflow Language Interpreter. So these processors give the users the opportunity to use tools and programs integrated in AWMS to complete their tasks.

MOI to support communication

The MOI used for communication is not described in the documentation at our disposal. It is evident that conversations are fundamental in the system, as the four phases of the loop are coherent with the conversation structure. We can easily suppose it being the one provided by The Coordinator.

However, as the emphasis is on facing recurrent breakdowns and on preventing them, the impression is that all communication is considered to fall within the business structure model. The apparently unanswered question is : which mechanism is used to resolve non-recurrent breakdowns that are not part of the network of workflows designed to represents the organization?

3.2.3.3. The Notation

The explicit notation used by the business process is the workflow loop (Figure 3.2-3). The loop is a sequence of four phases (or steps): request/offer, agreement, performance and satisfaction. Each loop deals with a particular action or task, that is completed by a performer to satisfy a customer's request. Each step can trigger the application of rules when one phase needs an input from another workflow.

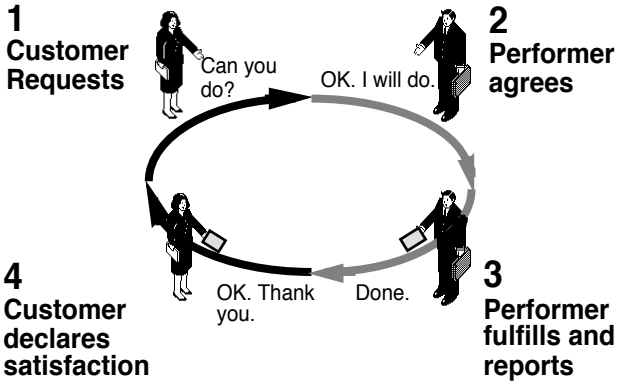


Figure 3.2-3. Workflow Loop

As for the use of STF processors, the notations are implicit in the involved application that can be accessed on a message-based, database-based or process-based way.

3.2.3.4. The features of mechanism

Comprehensiveness

The loop structure is applicable when there is coordination among people that can be modeled by a negotiation. Even if this is a quite diffused situation, it can be questioned whether this model covers all situations. Moreover, some criticisms formulated for The Coordinator apply to AWMS too, as they share the same basic principles.

Malleability

The system designer of the cooperating ensemble can generate, modify and maintain the definitions specific of his working group using a Windows-based graphical workflow creation tool from his Design Workstation. This tool permits to manage the MOI in a malleable but not distributed manner.

Visibility

As for the notation, it is visible to the users as the workflow loops and the rules to compose them are explicit. As for the visibility of the pragmatic effects, each user has the visibility of the mechanisms and each phase has its clear meaning in the overall context; users have an overview of where their tasks fit in the overall work.

Roles and actions are clear in one loop context: the roles are determined by the relationship between the customer and the performer. On the other hand, it is difficult to characterize roles when considering a network of interrelated loops, e.g. if user A is the customer of user B (the performer) within a certain loop and if user B is the customer of the performer C within another loop, it is not clear which is the relation between A and C. Moreover, the system does not consider how the organizational roles affect the control of the workflow loops and, consequently, does not allow to dynamically adapt the network to changes occurring in the organization.

Level of abstraction

The workflow loop is expressed at a level of abstraction appropriate to the work context. In fact the four phases of the loop are designed to face the completion of an activity by a performer on a customer's request, and this is a quite natural set of concepts for the users.

Propagation of changes

Since the only person able to modify the workflow network is the system designer, the system does not have to handle modifications made by different users. No consistency checks seem to be provided during modifications.

The context maintained

AWMS embodies an Action Workflow Reporter that provides an on-line monitoring capability as well as reporting functions and makes it possible to track cooperative work. With the reporter users can:

- graphically track any transaction in a given business process
- monitor the status of individual workflows and entire processes
- create a view of an entire process, with options to see either the present status or historical performances of individual workflows and performers
- create an historical profile of individual and group performances that includes all cycle times, number of actions late or on time, statistical assessments of customer satisfaction
- export process information to popular spreadsheets and graphical packages for further evaluation.

The degree of openness

AWMS is opened to be easily integrated with other action modalities by means of the STF processors, which allow message-based, database-based and process-

based applications to interact with the system. On the other hand, the system is closed to other communication modalities.

3.2.4. Domino

3.2.4.1. A short genesis and description

DOMINO is an office procedure system for modelling and monitoring structured office processes in organizations. It handles a variety of such processes which are specified in an application oriented language (CoPlan-S).

Cooperative office processes are modelled by coordination procedures that describe the information flow necessary to accomplish a common task within a group of persons. The process structure of coordination procedures allows for concurrent and alternative courses of activity.

A coordination procedure is performed step by step in a predefined manner. Each step (called *action*) is carried out by a certain person (called *actor*). The various actions of the procedures are assigned to *roles* responsible for their performance. The roles of a procedure stand for positions in the organization. Roles are assigned to certain persons when a procedure is started. One person can play several roles in different procedures; a role in a procedure can be played by different persons with different invocations of that procedure.

In general, an action needs information produced by other actions. As for the office, these pieces of information are called *procedure forms* (or *forms*). The execution of a procedure is started with the role assignment and the execution of the initial action. Subsequent actions can be carried out when all input forms have been produced. This process continues until all possible final actions have been executed: this terminates the procedure.

DOMINO mediates and controls communication related to tasks by notifying the participants about actions due, by providing them with the information needed, and by routing the results of such actions to the responsible parties.

The system (figure 3.2-4) consists of an automated agent, the *mediator*, and user components which communicate via e-mail using the CoPlan-X protocol.

The mediator is responsible for compiling, installing, executing and monitoring office procedures. It consists of the procedure compiler, the procedure control and the conversation monitor. Moreover, there is an organizational database used for role assignment during procedure execution.

Each user of the system has access to two components that support procedure processing: an interface module and the conversation monitor.

DOMINO runs on a network of personal computers and UNIX server machines.

• The system has evolved in time, so that relevant modifications have been made on the original release. In the following sections we will often refer to a demonstrator [KHKSW91].

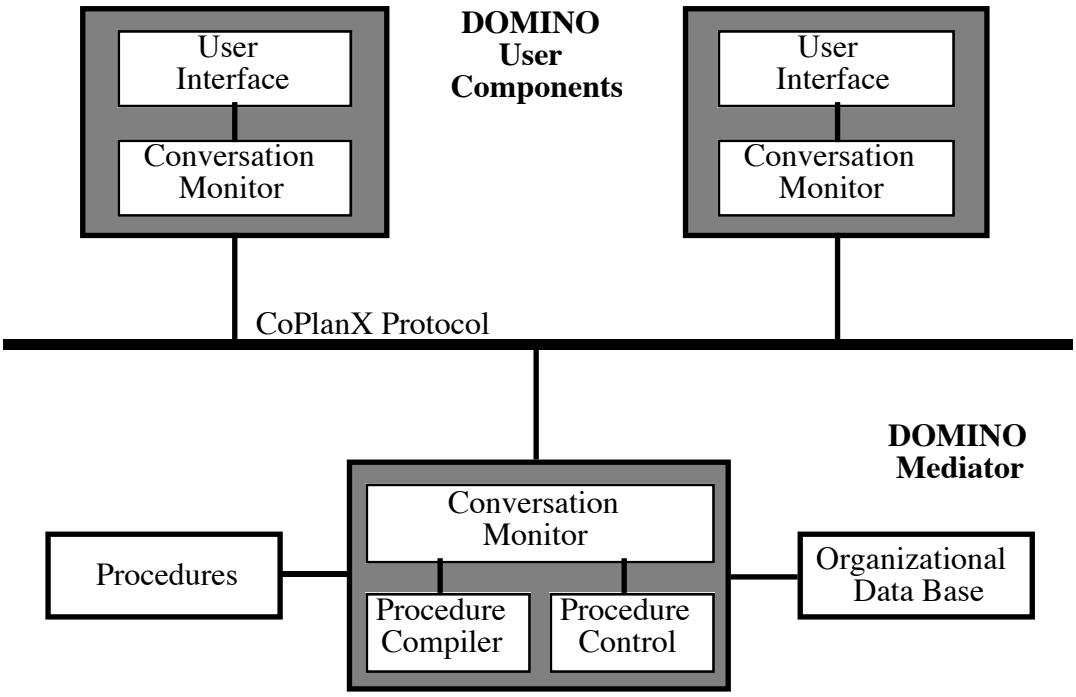


Figure 3.2-4. The DOMINO System Architecture

3.2.4.2. The basic MOI

MOI to support communication

DOMINO does not support direct communication among people. In fact, communication is mediated by the system since each conversation is just composed of messages (requests and responses) exchanged between the user and the system. There are 15 types of messages. In the following we consider the types used during the normal course of a procedure[1]:

- **execution** of an action (system request): an actor is requested to carry out an action. The message consists of the input forms needed and templates for the output forms to be produced;
- **completion** of an action (user response): an actor completes the execution or correction of an action as a response to a respective request. This message contains the alternative chosen (in case of a decision) and the produced output forms;
- **initialization** of a procedure (user request): a user wants to initialize a procedure. This message contains the procedure specification in the specification language CoPlan-S;
- **tracing** of a procedure (user request): an actor requests a record of the activities having occurred during the performance of a procedure;

1 The description of the various types of messages are directly drawn from the source documentation.

- **confirmation** of an action (system response): an action is confirmed, which means that the respective procedure has been successfully completed as a whole;
- procedure **record** (system response): this message is produced in answer to a tracing request, and upon termination of a procedure (for the initiator);
- **error** in a procedure specification (system response): the procedure system informs a user that the procedure specification of an initialization message contained errors. In this case, the procedure is not started.

In addition to the normal course, the processing of a procedure instance can vary in the sense that it may be set back by the procedure control system when one user complains about or when the initiator cancels the procedure. So the system provides mechanisms for exception handling in office procedures; these mechanisms are implemented as messages through which the system and the user communicate. The messages for exception handling are:

- **correction** of an action (system request): an actor is requested to correct a previously performed action. This message is caused by the cancellation of the action by the actor himself or by the objection of a subsequent actor to one of the forms produced by his action;
- **refusal** of an action (user response): an actor refuses the requested execution or correction of an action by specifying the reason for its refusal. When there is only a single actor assigned to the action, a refusal leads to the abnormal termination of the procedure.
- **delegation** of an action (user response): an actor delegates the requested execution to another person. The delegation message contains the name of this person;
- **objection** to input forms (user response): an actor objects to one or more of the input forms of an action. This message contains the names of the forms and the reason for the objection, and leads to resetting and restarting the procedure containing the actions that produced the objected input forms;
- **termination** of an action (user request): the initiator of a procedure requests the termination of an action the execution of which has been requested but has not yet been carried out. This message contains the name of the action to be terminated: all the causally dependent actions of the procedure do not take place;
- **cancellation** of an action (user request): an actor requests to cancel an action he had previously executed. This message leads to resetting and restarting the procedure with the action in question;
- **rejection** of an action (system response): it means that the procedure the action belongs to is abnormally terminated or has been reset. In the latter case, the execution of the action in question might be requested again;
- **excess of deadline** of an action (system response): the initiator of a procedure is notified that some actor has exceeded the deadline for the execution or correction of an action.

MOI to support action

The MOI which supports action within DOMINO is the cooperative office procedure. The execution of a procedure is started on request by a user who becomes the initiator of this procedure instance.

The actions of a coordination procedure are described by the input forms they need and the output forms they produce, i.e. by the way in which they communicate with other actions. The procedure does not tell how an action should be carried out, but rather what its result should be.

3.2.4.3. The notation

The notation for communication

The notation used in conversations to exchange messages is implicitly embodied in the CoPlan-X protocol. CoPlan-X consists of the different types of request and response messages described above. In addition, CoPlan-X messages have a *formal syntax*: it is defined as a certain regular expression of text lines of different format. The format of a line is determined by its first character: a Z-line contains a message type and a procedure identification and is always the first line of a CoPlan-X message, an A-line contains an action name, a B-line contains a user name, an S-line contains a procedure form name, a T-line contains arbitrary text, etc. The syntax can be interpreted by programs, such as the procedure coordinator and the user components.

The notation of the cooperative office procedure

The notation used by the cooperative office procedure is the specification language CoPlan-S, that allows for alternative and concurrent curses of action. An office procedure specification has three parts: the declaration of the roles, the declaration of the forms and the description of the information flow, which is a list of statements describing the input-output behaviour of the actions. Procedures are initiated and monitored by the procedure coordinator exclusively communicating with the participants' user components.

The information flow of a coordination procedure can be represented graphically (figure 3.2-5): actions are represented by boxes, and channels by circles. The input/output relation between actions and forms is represented by arrows connecting the respective boxes and circles. Notches in action boxes indicate that these actions produce alternative sets of forms and thus contain a decision.

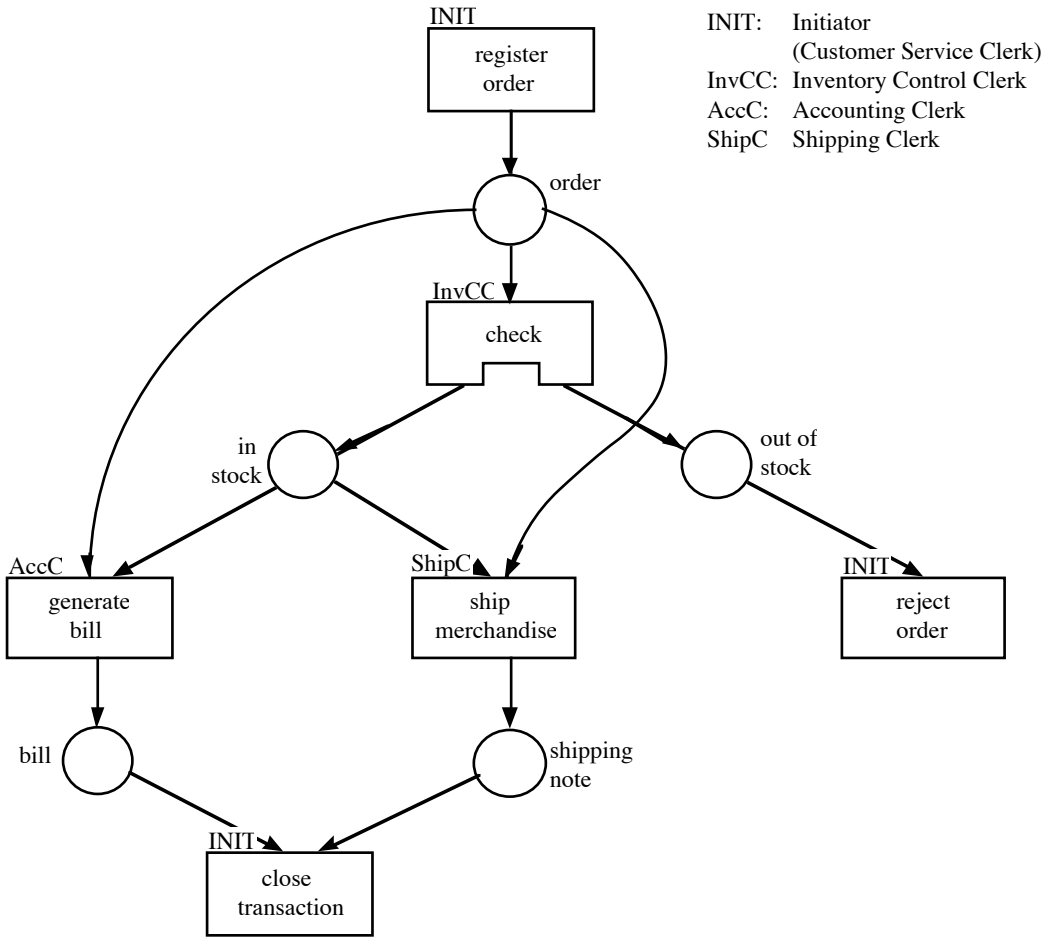


Figure 3.2-5. Example of a coordination procedure: Order processing

3.2.4.4. The features of the mechanism

Comprehensiveness

The specification language CoPlan-S permits to create office procedures tailored on the contextual needs in terms of tasks, agents, documents and their relationships. On the other hand, the mechanism seems to be tailored to working environments characterized by predefinable activities whose variations depend on the context of execution and not on flexible needs of the organization.

Malleability

Office procedures can be modified during execution, thus allowing for reaction to changing conditions or requirements. With this modification, the initiator may insert newly defined actions into the given office procedure.

On the contrary, the strict input/output relations between the actions of a procedure do not allow the data produced in one action to be changed in a subsequent action. So the mechanism is rather rigid, since the procedure has to be set back to the person who produced the data to be changed. Moreover, the environment for

working on office procedures lacks in tailorability, e.g. individual grouping of procedures or integration of personal tools.

Moreover, the exception handling facilities are not flexible enough. The experience of use showed there is a need for informal and free format communication.

Visibility

As for the notation, it is visible to the users through the above mentional graphic language. As for the visibility of the pragmatic effects, the tracing and report functionalities allow all participants to recover a consistent view of the procedure state by all participants.

Level of abstraction

The notations used within DOMINO are at a suitable level of abstraction. In fact the actions and the message exchanges correspond to the steps through which activities are performed in real world settings.

Propagation of changes

The above described malleability of office procedures can lead to inconsistencies which have to be taken care of. So consistency checks take place (procedure control, figure 3.2-5) whenever the plan is modified. These checks may concern, e.g.:

- unintended cycles in the causal net,
- deadlocks,
- possibility of a successful termination of the plan,
- availability of resources.

Modifications of the defined activities can be made on steps which are not in progress.

The context maintained

The main screen of the interface gives the user an up-to-date overview of the collaborative activity he is currently involved in and of work progress. A second element to maintain the context is the procedure form screen. It presents the information relevant to a procedure instance in a form like interface.

Experiences with the system have stressed that the organizational or group context was not adequately represented.

The degree of openness

Even if DOMINO is not meant to be an integrated office system, but rather as a tool for handling office procedures which may be used in conjunction with other tools like editors, data base systems, etc., however there is a lack of integration with other tools. DOMINO users complained particularly about two issues in this area:

- other programs, like e.g. a spreadsheet program, can only be integrated within a procedure processing via a not too comfortable copy and paste mechanism

- DOMINO messages are treated separately from ordinary electronic mail: the user has to switch tools to send an e-mail message concerning an office procedure.

3.2.5. WooRKS-UTUCS

3.2.5.1. A short genesis and description

The system has been developed within the Esprit II project ITHACA. In particular this system consists of a prototypal version that enriches the WooRKS (Bull, Massy France) library of objects [TLF88], [AS92]. This new library WooRKS-UTUCS is a framework for the development of workflow applications with communication functionalities [DG92]. The new module, which provides communication functionalities, is based on the UTUCS prototype developed by University of Milan [BDT91], [ADPT92].

The objective of the system is to support the development of workflow applications, where communication offers a way to handle exceptions that can arise during procedure execution. Whatever their role, office workers are involved in two types of activities: either they are doing some task on a predefined routine, or they are inter-acting with each other (communicating) in order to deal with an exceptional situation, i.e., they have to solve a problem that they or their colleagues have encountered.

A work process is a two facet object: that is, when work-flow follows the previously defined plans, it is made up of activities to be done by various persons (the routine dimension of the work process is emphasized); and when a breakdown blocks the planned work-flow, it becomes a subject of conversation among the same group of people (the creative dimension of work processes is emphasized).

WooRKS is a workflow framework implemented on top of an object oriented development system called HooDS (Highly objected oriented Development System), whose main components are the programming language CooL (Combined object oriented Language) and the database management system NooDLE (New object oriented Database system for advanced programming Language Environments). It provides nearly four hundred reusable object classes belonging to more than ten modules, a dedicated programming language to define activities and procedures, a compiler that translates them into CooL objects, and a kernel that supports coordination during procedure execution.

UTUCS (User To User Communication Support) is a system to support a group of persons interconnected in a communication network in handling the conversations carried on by means of different communication media: e-mail, face-to-face and group meeting. In the system the basic item of the communication is the conversation.

3.2.5.2. The basic MOI

The system implements two MOI that are two complementary visions of a work process: the action flow and the communication flow. In the first case only the performed input-output transformations are taken into account as predefined in a standard way (procedures); in the second case the abstraction takes into account the network of communications (for action, for possibility) and the sharing the knowledge created within the work process.

MOI to support action

Routine office work can be described as structured recurring tasks called Procedures, whose basic work items, called Activities, must be performed by various persons called Actors in the organization. The first MOI offered by the system to support office work is the workflow mechanism. It is used to start procedures and activities, to perform the activities and automatically schedule them between the actors.

WooRKS applications are composed by windows, each one of them containing office objects on which the user can act by the actions inserted in the menu of the window.

The *workflow basket* is a container with all the activities an actor is in charge of and all the procedures he is responsible for under his responsibility. The workflow baskets supports the user in starting an activity/procedure, abort, suspend or conclude the execution of an activity. Each actor can see the history of the procedure (the list of activities and their status, the list of the documents that have been produced). Each person responsible for a procedure can maintain an archive of ended procedures.

MOI to support communication

The communication mechanism used to establish mutual commitment (on an activity/procedure or not) is UTUCS. Like the Coordinator, UTUCS rests its foundations upon the Language/Action perspective, stressing the communication process, or more precisely its pragmatic dimension, as the primary aspect of office work.

UTUCS conversations are classified in four types, according to the commitment they concern:

- ‘Commitment to do’. This type of conversation subsumes all conversations in which people discuss a commitment for doing an action.
- ‘Commitment to be’. This category consists of conversations about possibility. Their effect is to transform the organizational setting.
- ‘Information handling’. This type of conversation can be opened to delegate someone to manage an information base.
- ‘Information providing’. This type of conversation can be opened to give or request some information.

As for integration with the action environment, when an exception arises while an actor is executing an activity, he needs to contact either one of the actors participating in the workflow procedure or the person responsible for the procedure. The activity that creates the problem is the subject of the conversation with its corresponding workflow procedure context, and the result of the conversation can be the missing information being provided, or an action by the procedure responsible solving the stalled situation.

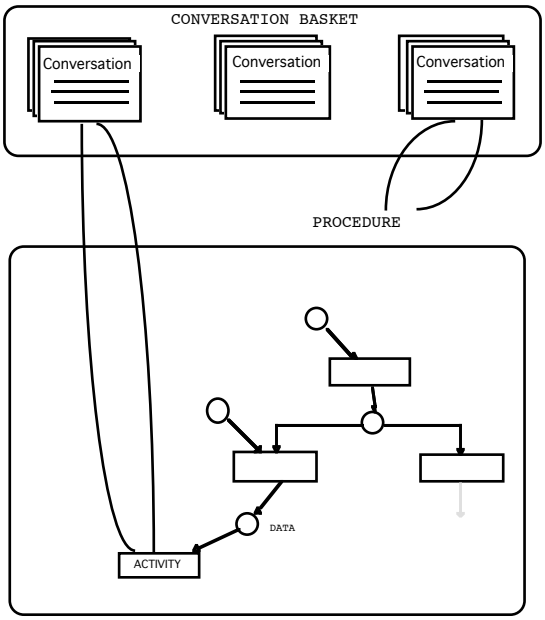


Figure 3.2-6. The MOI of WooRKS-UTUCS

3.2.5.3. The Notation

ADL: The Activity Definition Language for workflow applications

ADL is a programming language devoted to workflow purposes. Using this language, one is able to define a network of activities either connected by an object flow or directly connected using the control flow only.

For every activity it is possible to define a maximum duration and the responsibility of someone for this activity. An activity itself may consist of a sub coordination procedure.

The activities and procedures defined using ADL are dedicated to the COP (COrdination Procedure) kernel which follows the sequence defined in the procedures, and assigns activities to the specific actors according to the definition of activities.

In order to clarify the ADL role in the definition of workflow procedures, in the following the main design phases of a workflow applications are illustrated.

Once an application is identified, it needs to be described in terms of activities with necessary control flow to build a procedure. The input and output data for each activity should be clearly identified.

Each data item needs to be defined as a CooL object type. The object definitions can be selected from the various modules of WooRKS . The **organization** module describes object definitions for actors who will perform an activity under the capacity of the roles they can perform in various units of an organization. The **information** module describes the semantic attributes and contents of the information circulated inside a procedure. The **time** module describes basic definitions to represent time information regarding the activities. Each activity is described by an atomic **action.** The object definitions for generic interactive and non-interactive actions are in a library. This aids in building data objects for a particular application as well as customize the user interfaces for interactive actions.

In summary, the main design phases are the following:

- Definition of the procedure network and needed objects
- Identifications of re-usable objects inside the modules information, organization, time and agenda
- Identification of specializations needed in the modules information, organization, time and agenda
- Identification of re-usable activities and actions
- Identification of new activities and actions needed
- Procedure definition in ADL and code generation

In Figure 3.2-7 and 3.2-8 are given the synopsis of procedure and activity definitions.

In Figure 3.2-8 is illustrated the use of ADL code in order to get Cool code containing the procedure to load the net and the definitions of the procedure and activities types.

```
DEF_PROCEDURE procedure_name
[ROLE MethodCall | ROLE Dependency OF RoleTerminal ]
[duration]
[OBJECTS
     CONTEXT obj_var, ... : object_type [ :default_value ]
         ...
     [ VIRTUAL obj_var, ... : object_type : default_value ]
         ...
[INPUT  obj_var, ...]
[OUTPUT
     ALTERNATIVE  alt_name
        [obj_var, ... ] FROM
             [alt_name OF] activity_name, ..OR
        ...
] ]
ACTIVITY     activity_name   :    activity_type
     [ROLE MethodCall |ROLE Dependency OF RoleTerminal ]
     [duration]
     INPUT
        [CONTEXT obj_var, ...]
        [VIRTUAL obj_var, ...]
        [obj_var, ...] FROM
             [alt_name OF] activity_name, ... OR
             ...
     OUTPUT
        [ALTERNATIVE    alt_name]
             [obj_var, ...]
        ...
END_ACTIVITY
...
END_DEF_PROC
```

Figure 3.2-7. Synopsis of procedure definition

```
DEF_ACTIVITY activity_type
[OBJECTS
     obj_var, ... : object_type[INPUT
     obj_var, ... ]
[OUTPUT
     [ALTERNATIVE alt_name]
```

```
        [obj_var, ... ]
        ...
]
]
[PRECONDITION
     [BYPASS WHEN [NOT] method_name1 ( [ Param1.1 ... ] )]
     [EXCEPTION WHEN [NOT] method_name2 ( [ Param2.1 ... ] )]
     ]
[POSTCHECK
     [alt_name WHEN] [NOT] method_name3( [ obj_var3.1 ... ] )
     ...
]
ACTION Action_Method_Name |
SUBPROCEDURE SubProc_name SubProcInterface |
SUBPROCEDURESubProc_name SubProcInterface
     FOR_EACH_ELEMENT_OF VariableName |
SUBPROCEDURESubProc_name SubProcInterface FIRED_BY ActionCall
END_DEF_ACT
```

Figure 3.2-8 . Synopsis of activity definition

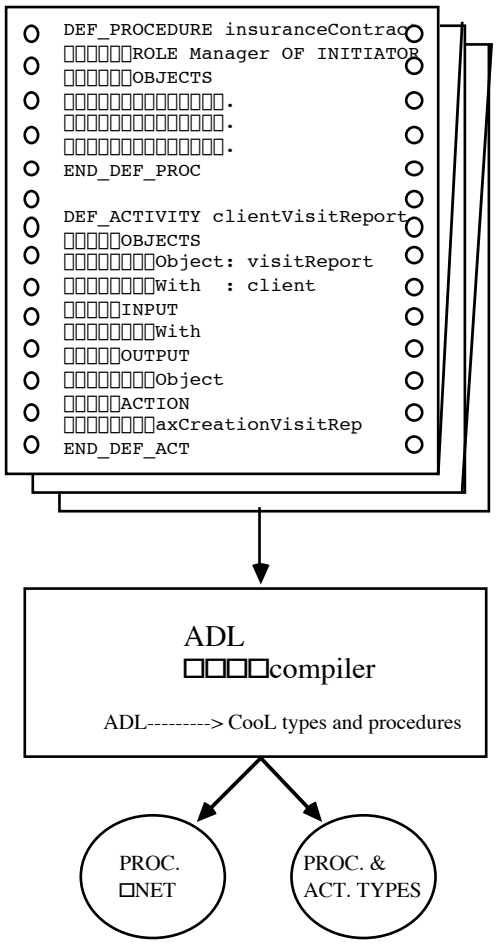


Figure 3.2-9. ADL usage

The transition automata of UTUCS commitments

In UTUCS each conversation is characterized by the possible states of its object and by the possible transitions that can be performed (i.e. the speech acts exchanged among the partners) from one state to reach another. In Figure 3.2-10 and Figure 3.2-11 the state-transition diagrams representing a ‘Commitment to do’ and a ‘Commitment to be’ conversations are depicted.

Present State	Speech Act	Next State
()	A Offers	Offered
	P Requests	Requested
Requested	A Counter-Offers	Offered
	A Refuses	Rejected*
	A Accepts	Taken
	A Concludes	Concluded
	P Withdraws	Deleted*
Offered	P Counter-Requests	Requested
	P Refuses	Rejected*
	P Accepts	Taken
	A Withdraws	Deleted*
Taken	P Counter-Requests	Requested
	P Cancels	Deleted*
	A Counter-Offers	Offered
	A Concludes	Concluded
Concluded	P Refuses	Taken
	P Evaluates	Satisfied*
	P Evaluates	Not Satisfied*

* Terminal States; A = Active, P = Passive

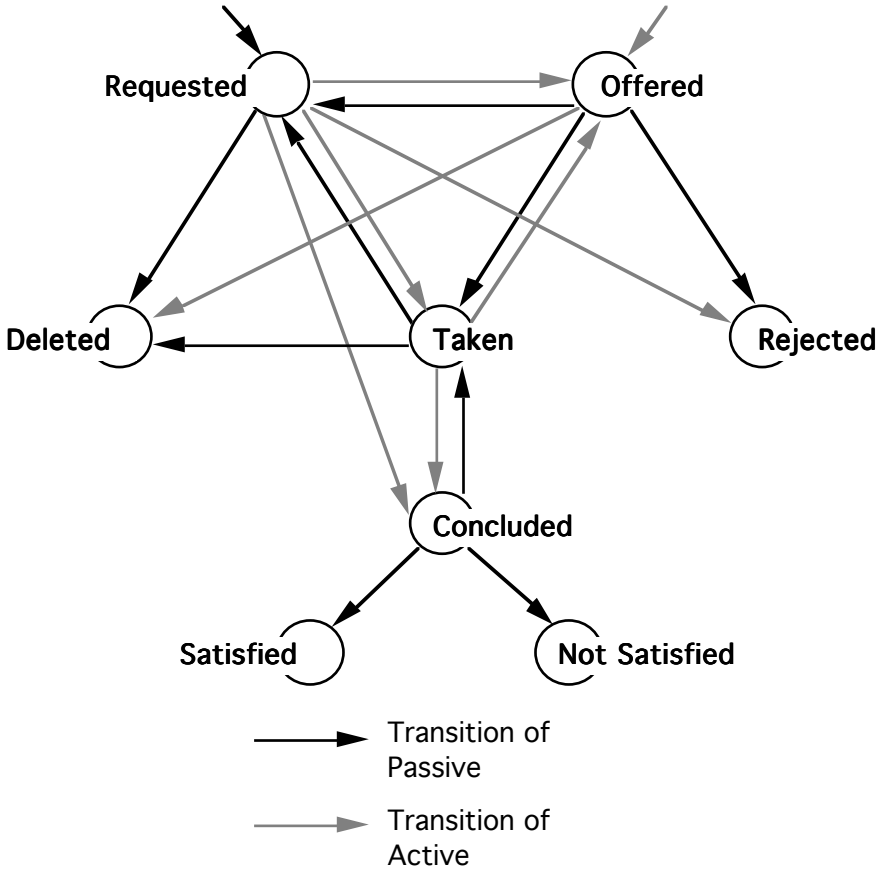


Figure 3.2-10. “To do” transition table and automaton

Present State	Speech Act	Next State
()	A Offers	Offered*
	P Requests	Requested*
Requested	A Negotiates	Offered*
	A Accepts	Accepted
	A Refuses	Rejected
	P Deletes	Deleted
Offered	P Negotiates	Requested*
	P Accepts	Accepted
	P Refuses	Rejected
	A Deletes	Deleted

* Initiating states; the other ones are all terminals

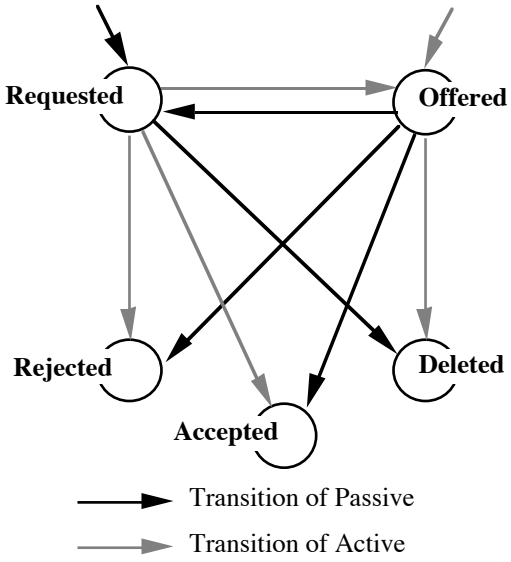


Figure 3.2-11. “To be” transition table and automaton

3.2.5.4. The features of mechanisms

Comprehensiveness

ADL is a dedicated programming language for office procedures, where the actions embodied in each activity of the procedure definition are either CooL object specializations of the class action of the module ActionLib, or sub-procedure calls. At the application engineer level the generality of the language is due to the possibility of redefining activities from the library of WooRKS (examples of action available are actions for creating, showing, revising, mailing and printing a document), adapting it to the specific activity.

As for the conversations, we can refer to the already presented considerations about this type of MOI.

Malleability and Visibility

WooRKS applications are developed using ADL notation by the application engineer. Each application is defined by a set of activities and their relations in terms of input/output and control flow. These relations can be redesigned by the application engineer, but are not accessible to users. The MOI devoted supporting communication provides a partial solution of this problem, since it provides a way to start a conversation about actions that have not been included in the procedure definition and to link these latter to the procedure.

Level of abstraction

ADL is a dedicated language for procedure definition, so it offers the way to define the flow of the objects in the network of activities, with their preconditions and postconditions, as well the duration and the role of each activity and procedure.

The level of abstraction of the notation underlying the conversations is based on the Language/Action perspective and offers an adequate set of speech acts with respect to conversation for action and for possibility.

The system WooRKS — UTUCS offers the way to combine the two mechanisms as the user needs and decides. This coupling is at user level.

Propagation of changes

Since the only person able to modify the workflow network is the system designer, the system does not have to handle modifications made by different users. No consistency checks are provided during modifications.

The context maintained

WooRKS applications hold the history of each procedure either under execution or already concluded. The history contains the state of each activity (STARTED, ABORTED, SUSPENDED or CONCLUDED) with temporal information and the list of all documents produced.

UTUCS is a system devoted to integrating steps of conversations independently of the communication media exploited (e-mail, face-to-face and group meeting). The information stored in the information base are conversations (CRR), talks and minutes (see Figure 3.2-12). Each conversation is a sequence of steps ordered in a temporal way and each step holds a link to the context that generated it (if the step is from a private talk or from a meeting).

WooRKS-UTUCS adds to these features the possibility of creating links between procedures and conversations. From a conversation it is possible to retrieve the activity/procedure that generated it, and from a procedure it is possible to see all the conversations about it. Context facilities are also provided when a conversation is opened on an activity/procedure: in fact, it is linked to the history of the procedure, as well as to the documents that have been enclosed to the conversations. Moreover, the system gives information about the responsible of the activ-

ity/procedure as well information about the net of activities that temporal preceded it.

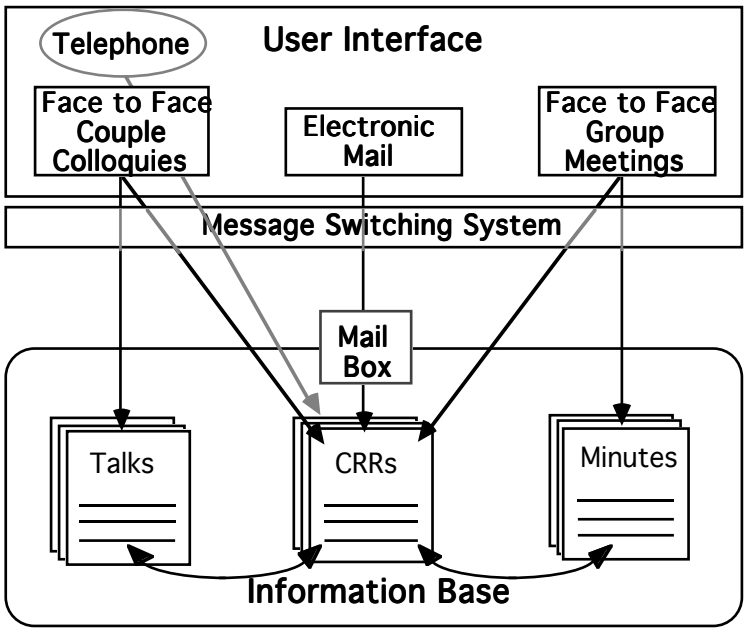


Figure 3.2-12. The architecture of UTUCS

The degree of openness

WooRKS-UTUCS is at the prototypal state. Some guide-lines for compatibility and interoperability between the system and the existing office tools have been identified, but at the moment this integration has not been fully implemented.

WooRKS has a module that is devoted to the integration of common office tools like wp, spread-sheet, fax in the actions that can be performed in a procedure. The module Operator provides these functionalities. The main objective is to allow the users the use of tools they already use in daily office work.

UTUCS offers some guidelines for technological openness by means of the concept of conversation step independent of the communicative media used.

3.2.6. Commitment Handling Active Office System

3.2.6.1. A short genesis and description

CHAOS (Commitment Handling Active Office System) is being developed at the Cooperation Technology Laboratory at the Computer Science Department of Milan University in the framework of a National Research Project [DSVZ88], [DDS89], [GS90], [B91], [SDOP92]. The conceptual framework of CHAOS is the language/action perspective to communication pragmatics. Since 1986, [DDSVZ86], its subsequent versions enlarge the set of functionalities provided to its users. The present version provides an environment integrating communication

(CH), activity management (GAM), organizational structure (GSM) and language (GLM). This conceptual framework is reflected also by the underlying system architecture, which is composed of four modules (figure 3.2-13):

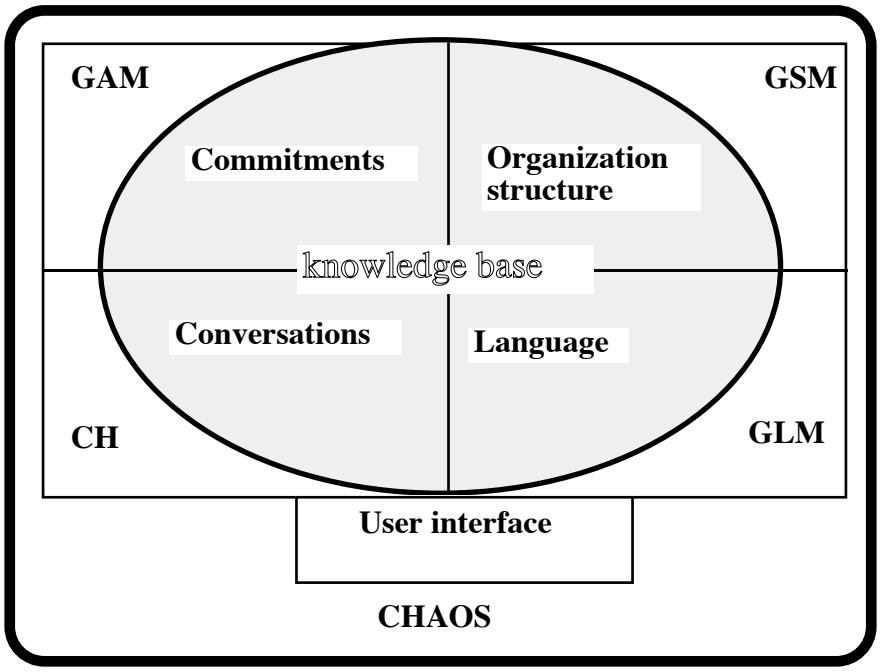


Figure 3.2-13. CHAOS Architecture

1) the Conversation Handler (CH), that maintains the knowledge about conversations by tracking their state and the relationships among messages;
2) the Group Structure Module (GSM), that maintains the knowledge about conversational models, experience and responsibility distribution;
3) the Group Language Module (GLM), that maintains the Lexicon and the Dictionary of the language used within conversations; the GLM contains also the User Models (UM), which are partial views of the world where the knowledge each organization member acquires in the communication and cooperation with other group members is maintained;
4) the Group Agenda Module (GAM), that maintains the knowledge about the commitments and their fulfilment.

The basic idea is that all the knowledge above can be acquired from handling conversation, by fully exploiting their semantic and pragmatic content. To this aim, messages within conversations show a structure also in the propositional content. Each communicative act or event triggered by the user propagates its effects on all knowledge bases to keep their content immediately updated. The main effort during CHAOS design and implementation was concentrated on the propagation of these flows of knowledge.

Within the integrated environment, conversations are viewed as the back-bone supporting communication, activity execution and role playing. In fact, each conversation is a structure of communication acts to define commitments and roles, and of activities to fulfil the commitments according to the rules characterizing the role of the involved people.

CHAOS is still under development: so many of the functionalities described in what follows are already implemented and some are at the specification stage. Other exist in separate pieces of software to be integrated.

3.2.6.2. The basic MOI

MOI to support action

The MOIs that support action live in the GAM. They are of two types: supporting each single user in handling his own commitments and supporting complex (non-elementary) activity design and execution. These latter are in the specification stage.

The first type of MOI are: the Agenda, the To-do List and a set of Structured Commitments.

The Agenda and the To-do List

The Agenda records instances of individual commitments inserted in it manually by its owner; it automatically records also commitments taken during a conversation, allowing the user to see how his time is occupied in a day, a week or a month. It graphically shows and provides primitives to manage overlappings of different commitments. The To-do list contains both communication actions and commitments. It reminds each user what are the actions he has to perform in the current day. It contains the commitments to be completed, those that have a deadline which has expired in previous days but have not been done yet and the conversations for which an answer is expected during the current day. At the present stage, both the Agenda and the To-do-list are not actually MOIs, as they support just individuals. Anyhow, they show some of the primitives that will possibly become available when the MOI supporting non-elementary actions will be designed.

Structured Commitments

Within a conversation, the propositional content, i.e., the commitment being defined, is expressed trough a semistructure that allows the system to draw the essential information. Commitments are organized in an IS-A hierarchy dynamically amenable to modifications by the users. In this sense it can be viewed as a MOI, as it induce a shared classification of actions to be performed.

MOI to support communication

The MOI used for communication is the conversation. It is the means by which commitments are mutually taken, both to accomplish tasks (Conversation for

Action, CfA) and to define the organizational structure (Conversation for Possibility, CfP).

MOI to support organizational structure

The structure of an organization is modeled in terms of Areas (i.e. units of organization) and of Activities (i.e. units of work) performed in these Areas. An area identifies the scope in which its Manager can make effective declarations (like, defining sub-Areas, “hiring” new people in the Area and defining their role).

Within a CfP involving a Manager the discussed commitment is the definition of a role that a person accepts to play in the Area managed by that Manager. A role is defined through the space of possibilities within CfA and CfP related to the Area under concern and to the on-going Activities. Hence a role in CHAOS is a set of rules governing future behaviours, defining rights and duties of a group member within Areas or within Activities. When a role is agreed upon, the related rules are passed to the GSM in order to keep trace of the modifications of the role definition, and to put them at work in subsequent actions.

CHAOS maintains a record of the successful fulfilment of commitments through the notion of Experience in Disciplines involved in that commitment. The degree of experience is determined by some (user definable) policy attached to the conclusion of a CfA in the state “satisfied”.

Being able to follow the evolution of the commitments and its impact on the organizational structure in terms of committed actions and user roles and experiences, allows the system to answer user queries about aspects of the organization they are not informed about by retrieving the most appropriate view of the domain. In the case of a query, the system answer is sought in the UM of the user ‘privileged’ with respect to the matter of the query under concern. In other words, if a user needs information on a specific action or object, the system will supply as an answer the information of the user that is currently responsible for it.

3.2.6.3. The Notation

The notation of commitments

Commitments are described by semistructures , i.e., by a set of attributes of the type: < name, value>, where the second can be either free text or formalized information. In both cases, they are active items allowing the navigation among the objects they represent. New types of commitments can be defined by the users by grouping sets of attributes whose type belongs to a predefined set.

The notation of conversations

The notation used to represent conversations is the finite automaton, as in the previous section. However, in CHAOS the space of possibility defined by the CfAs and CfPs show some small differences.

The notation of the Organizational Structure

Semantic networks are used to represent Areas, Activities, Disciplines and their relationships. Areas and Activities are represented by two kind of nodes: terminal nodes represent Activities. Each node is enriched with information about the involved people and their roles. All this information is active and allows one to access the appropriate knowledge base. Disciplines are organized in a graph whose nodes are the Disciplines and whose arcs represent two types of relations: *must* means "being contained" and *may* means "proximity". These relations are used to determine how the experience in a discipline propagates to the related ones.

Roles are represented as a set of rules whose basic items are: the Area/Activity, the actual Person, the Conversation State where it applies, the type of Control it triggers. The system allows for recording and sharing among its users patterns containing recurrent role definition.

The notation for the linguistic knowledge, and hence for the UMs, is frame-based.

3.2.6.4. The features of the mechanism

Comprehensiveness and Level of abstraction

As for the applications presented in the previous sections, comprehensiveness depends on the extent to what the Language/Action perspective and the approach to pragmatics it induces is considered general.

The notations used within CHAOS are expressed at a level of abstraction appropriate to their context. In fact the commitment is thought to formalize the features of a task, the conversation is designed to face the communication through which the task is negotiated and taken, and the handling of responsibility and experience deals with the notion of role.

Malleability

CHAOS is tailorable to the target environment, as Commitments, non-elementary Actions, Roles are user definable on the basis of the target organization.

The policy applied to the evaluation of people experience can be modified at any moment by a restricted set of users owning this capability (usually, the managers of the personnel).

There are two kinds of commitment hierarchies: global and local. Both are used to instantiate the commitments as propositional contents of the conversations. The global hierarchy that can be modified by an administrator. Each user can attach to its local copy local commitments. The receiver of a message containing a commitment belonging to the local hierarchy of the sender can decide to attach it in his local structure, improving shared representations. When a commitment is shared enough it is inserted in the global hierarchy by the administrator.

This centralized policy will be substituted by a structure of partial shared views connected both to the role definition and the information contained in the GLM about the group Lexicon and Dictionary. This new approach seems to be a good compromise between sharing and locality requirements.

Visibility and Propagation of changes

The effects of all mechanisms are totally visible to the user as the transfer of information and triggering of actions are documented and recorded through the various knowledge bases. The GSM and the GAM keep track of the modifications of the Area/Activity and Role and Commitments/Actions definitions as soon as they happen. Moreover, attention is paid to the relationship between local and global information. In any case, modifications are perceived when the modified items are accessed.

The context maintained

CHAOS keeps track of the evolving organization in terms of the dynamic definition and distribution of roles. The system keeps track also of the on-going actions in order to monitor planned deadlines and flows of information.

CHAOS has a dynamic organizational structure which records the role of each member in terms of responsibility and experience. This is one of the main advantages of the system compared to the others.

The cognitive context of the communication is maintained by the story of ended or in progress conversations. The user can access the story of a conversation to see the sequence of exchanged messages and to access the related information.

Finally, the Agenda and the To-do List maintains the information about the temporal dimension.

The degree of openness

The integration with tools supporting action execution is one of the main goals of the present evolution of CHAOS: the idea is to associate to each commitment an "active working space" which contains the resources, the execution modalities and the connection with other operational environments. This is part of the on-going design of the non-elementary commitment management.

3.2.7. Task Manager

3.2.7.1 A short genesis and description

The Task Manager (**TM**) [KHW93] is an asynchronous, geographically distributed prototype software system for specifying and managing cooperative activity. With its help, users may organize cooperative tasks, monitor their progress, share documents and services and exchange messages. The authors propose TM as a system overcoming some limitations of DOMINO, specifically the rigidity of

pre-defined procedures an not adequate exception handling, together with the isolation from informal communication and information sharing.

The TM distributes task specifications, attached documents and messages to the involved users in a consistent way. It provides:

- **support** of organization and **planning** of collaborative work
- up-to-date **overview** of collaborative activity and work progress
- **dynamic modification** of work plans during performance
- availability and **exchange of documents and messages** within groups of people involved in task performance

The TM organizes distributed work in tasks which have a person responsible, a deadline, other participants, the material needed for the task performance, and possibly subtasks.

The user interface of the TM is a hierarchically structured to-do list which displays all tasks a user is involved in and which allows direct access to the relevant information attached to a task.

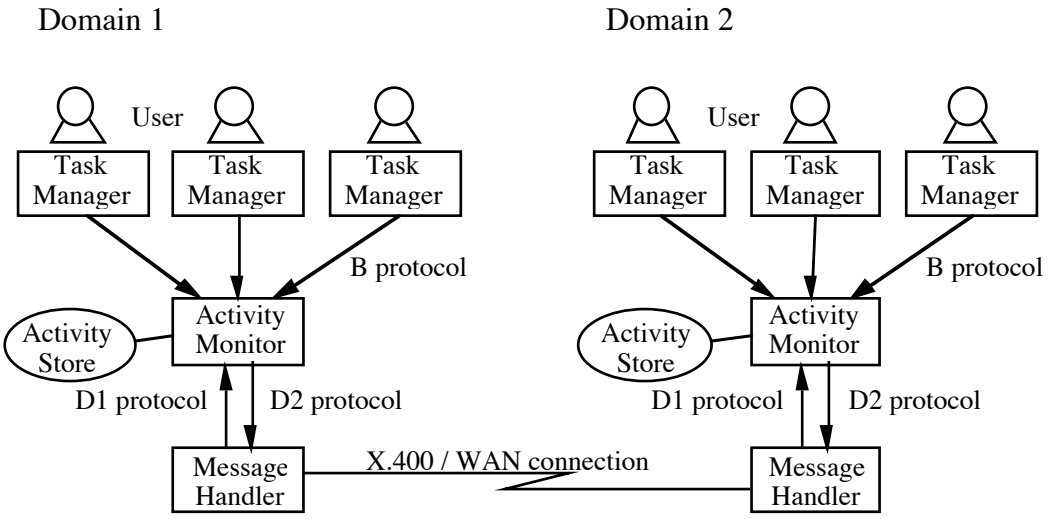


Figure 3.2-14. Software architecture of the Task Manager

The development of the TM began assessing two problems of DOMINO:

- the **rigidity** of pre-defined procedures
- the **isolation** from informal communication

The central notion of the TM is that of a task. To perform it, people use shared documents and/or services and communicate by sending messages to each other.

Figure 3.2-14 shows the TM software architecture. It shows two domains linked by standard X.400 store-and-forward message transfer. Within a domain, a client-server approach is used to update a shared data structure, while X.400 messages are used to distribute changes to other domains. Every user of the system is located in a specific domain and uses his own instance of the TM. Using remote procedure calls, the TM talks to the activity monitor to get up-to-date data

and to request changes to the domain-wide shared activity store. In each domain, there is one instance of the Activity Monitor that serves multiple TM instances.

3.2.7.2 The basic MoI

MoI to support communication

TM supports communication through messaging. People freely exchange informal electronic mail messages within the context of a task. Implicit communications are supported via sharing documents and services attached to tasks. Then, TM does not contain a real MoI to support communication.

MoI to support action

The MoI of the TM to support action is the **shared to-do-list** that contain tasks to be executed individually or in cooperation. The notion of task has diverse aspects. It can be intended as a common goal of a set of people (*result-oriented*). On the other hand, a task can be composed of various sub-tasks and dependencies can be defined among them and their documents (*procedure-oriented*). A task can also be used as an unstructured folder, which is used as a container of subtasks, documents, services and messages (*information-sharing-oriented*).

Tasks are specified by a set of attributes:

- the **title**, which identifies a task to its participants and gives a description of its goals; it is the only mandatory attribute;
- the **state**, which can be *finished* or *not-finished* (set by the user); *pending* or *not-pending* (set by the system); *acknowledged* (set by the responsible);
- the **deadline**;

Beside these, there are other attributes to specify in more details how a task should be performed: time-related data, data that describe causal dependencies between tasks, personal data, etc.

Operations are either invoked by the user or by the system as a result of pre- or post-conditions that have become true. Users can create tasks and subtasks, dependencies between tasks and between tasks and documents. Persons responsible for a task may refuse responsibility and they may reassign it to another user.

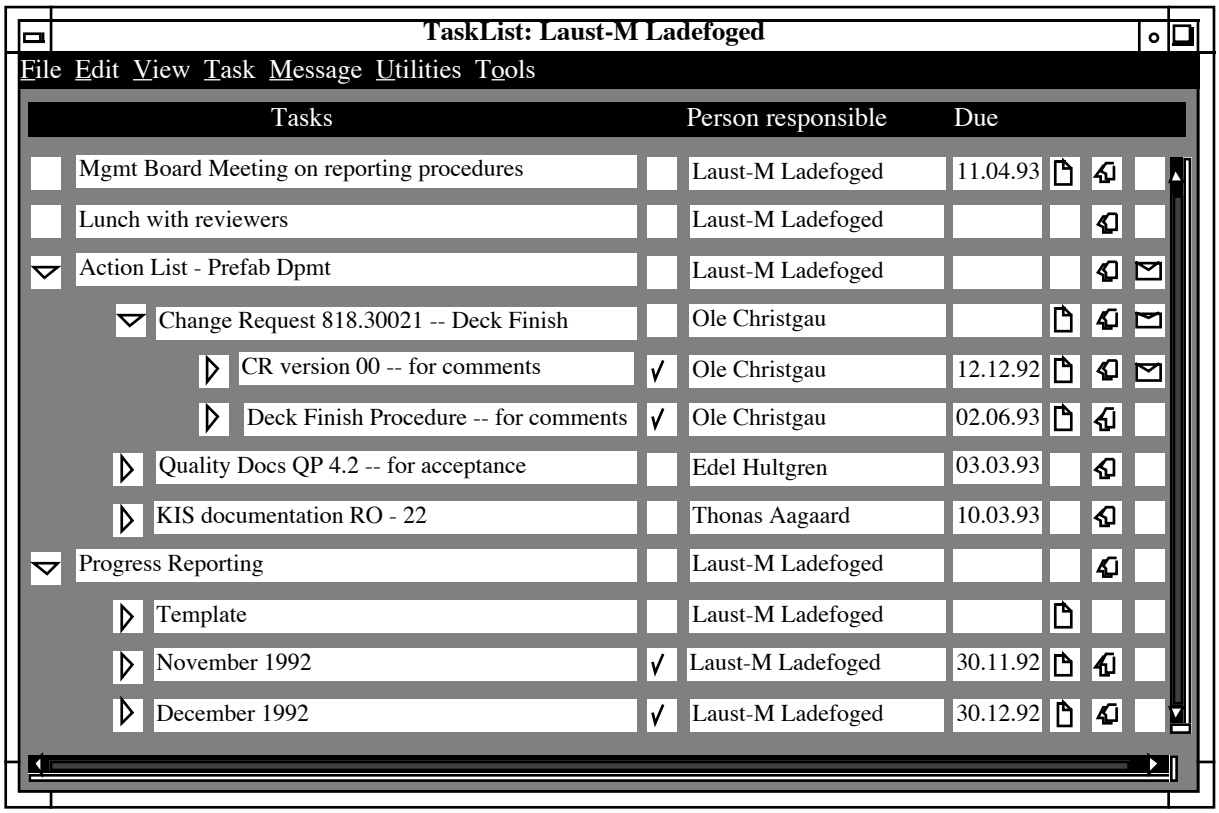


Figure 3.2-15. The Task List

The set of tasks a user participates in is presented as a *Task List*. (figure 3.2-15). The Task List gives an overview of the hierarchically ordered tasks along with a condensed view of the most important attributes: *title*, *person responsible*, *deadline*, *documents/services*, *participants*, *state*, and a list of *messages* that have been exchanged within that task.

More detailed information is provided by the *Task Editor* (figure 3.2-16), which may be used to view and edit all the attributes. The *Dependency Editor* represents groups of sub-tasks and documents in a net-like way; it serves the purpose of graphically displaying and editing dependencies between tasks and resources. The dependency structures are similar to those used in workflow systems, but are interpreted differently: instead of driving a workflow with a strict sequencing of steps, dependencies in the TM represent recommendations which may be changed or overridden by the users at any time.

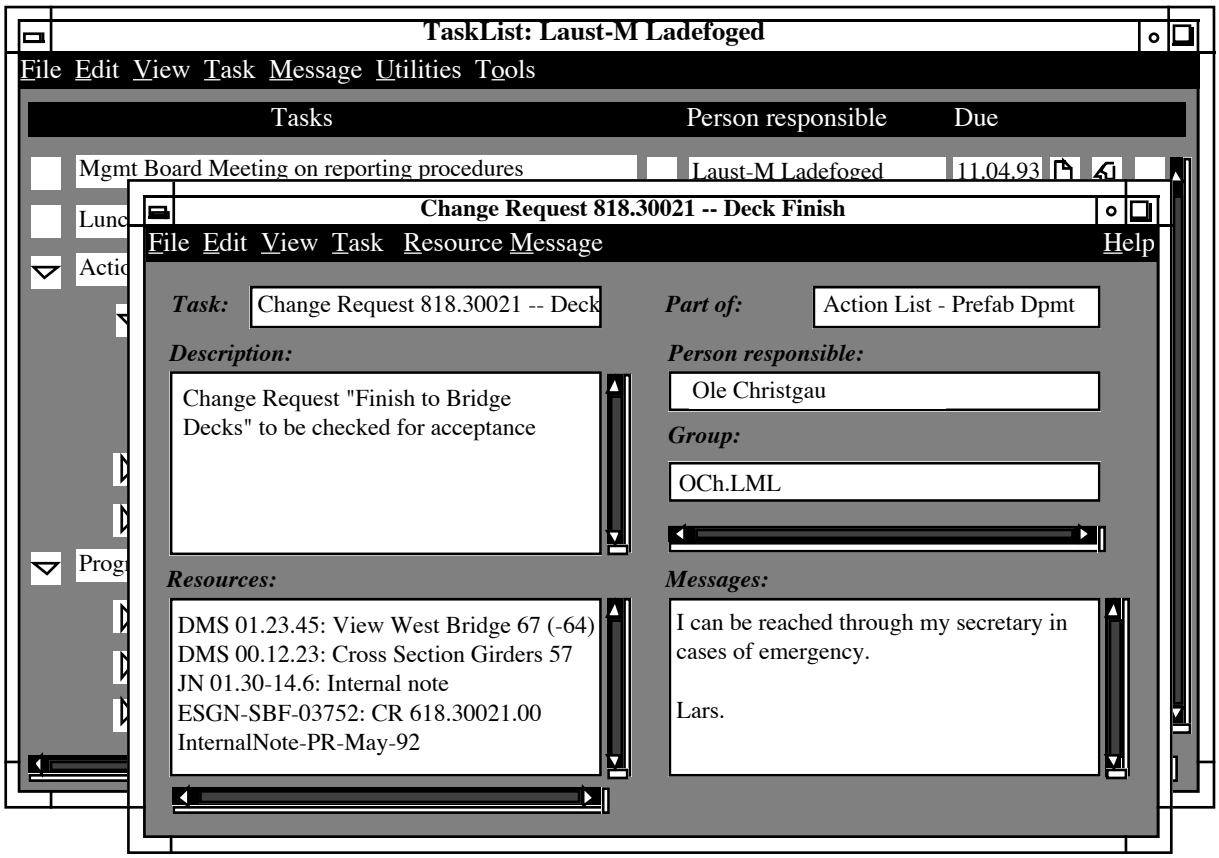


Figure 3.2-16. The Task Editor

The TM distinguishes among various levels of participation in a task:

- the **participants**: it is the set of the involved people that all have equal access rights to the attributes of a task;
- the **person responsible**: he has exclusive write access to some of the task attributes, and only he may reassign the responsibility of the task to another person; when a user creates a task, he automatically is its responsible;
- the **observers**: they have only read access to information and they have the right to participate in the informal message exchange associated with the task.

3.2.7.3 The notation

TM does not use any notation for communication. As for the action, the structure of the to-do-list implies a hierarchical way at looking tasks as they can be decomposed into sub-tasks. Their mutual dependencies are established via exchange of data/information and by the timing constraints.

3.2.7.4 The features of the mechanism

Comprehensiveness

The dimensions of work articulation considered by TM are: agent, task and information resources. TM can be used in various ways, and not only for the management of clearly defined tasks. In its present form, the TM is a simple tool which can be applied to any kind of collaborative activity, because it hasn't been tuned to any specific application domain. The TM is fundamentally an asynchronous system, but it doesn't forbid a synchronous use. For instance, a user may realize that another user is working on a task at the same time he does, and he could decide to discuss with him a group of subtasks while both are looking at the task list.

Malleability

Tasks may be copied and pasted or moved around freely. This may be done at any time, thus allowing for dynamic modification of the work situation at run time.

There are some typical forms of usage which demonstrate the versatility of the TM. They are: personal to-do lists, brainstorming and conferencing, meeting preparation, project planning and monitoring, project execution, repetitive tasks and bulletin boards. The various usages of the system are not necessarily to be kept separate from one another, but can dynamically develop from one kind to another; for instance, a task in a personal to-do list can be turned into a brainstorming item by adding participants; people involved in a planning activity can drop out of the task and new participants can substitute them, and so on.

A shared task may be structured in various degrees of refinement and causal dependency, and all attributes are dynamically changeable. The only limitation to flexibility is the rather egalitarian model of task sharing: all participants share the same view.

Level of abstraction

Users of the TM may organize their collaboration as they like, the only prerequisite being the installation of the tool in the users' community. This is made possible by the simplicity of the tool, which seems to be at the appropriate semantic level.

Visibility and Propagation of changes

Each user has instant access to the tasks he is involved in. The system produces a consistent view on tasks for each participant. Users can modify tasks by changing their attributes and the modification is perceivable by the other users.

When a user modifies a specific task or the task structure, the TM sends an operation request to the Activity Monitor, which executes the operation in the activity store and via the Message Handler broadcasts the request to the remote domains involved. When an operation request arrives from a remote domain, the

Message Handler passes it over to the Activity Monitor which then updates the corresponding task objects in the Activity Store accordingly.

The protocols defined within the software structure (see figure 3.2-16) enable the Activity Monitor to detect concurrent updates of task attributes and to resolve conflicting assignments by assuming a linear order on those events.

The propagation of changes seems to disregard any control about the consistency of the new task configuration (e.g., in terms of time or exchange of data).

The context maintained

The documents and services which are attached to a task contain a history of who did what and then. The system offers monitoring and task tracking at execution time as well as report generation after completion of a task. The Task List gives an overview of the hierarchically ordered tasks along with a short view of the main attributes. The *Logging Tool* gives a complete history of events pertinent to the tasks.

The degree of openness

Various kinds of resources are needed to achieve the goal of a task. The main resource is the sharing of documents. But there are also other resources that are handled by services that are implemented outside of the actual TM, but may be referred to and shared by the participants of a task.

The concept of documents and services associated to a task causes the system to be “open-ended”. It supports documents of any kind as long as their associated tools, like word-processors or graphical editors, running on the underlying UNIX operating system.

3.2.8. Lotus Notes

3.2.8.1. A short genesis and description

Notes [L89] is a customized multiplatform product that allows the user to take no care of the hardware and operating system used. It is a form-oriented client-server system based on a satisfaction-oriented approach. It comes as an answer to two main motivations: faster information access and synchronization of activities among the central and the peripheral locations.

Its scope is to support cooperation within a virtual workgroup, i.e. geographically distant people with different knowledge, whose structure and members evolve in time.

Lotus Notes is proposed for three types of basic applications:

1 — Disseminating information

2 — Routing information

3 — Interactive applications

Once connected to a server, there are two ways to work:

- *interactively*, on shared databases on the server. Working interactively is working in real time over an active connection; changes are made directly to a shared database while being connected;
- *locally*, on databases stored on one's workstation hard disk. Consistency is then maintained updating the shared databases on the server by means of the replication mechanism, which will be described in the following. Working locally is working without maintaining a constant connection.

Notes databases differ slightly in structure and terminology from traditional ones. A database consists of any number of records called notes. A note is a general structure that Notes uses to hold a variety of information. Each record consists of any number of fields called items; each item is composed of three elements:

- a field name
- a data type
- data.

Views, forms, icons and policy documents are all stored as notes; documents are also a specific kind of note, a data note. A Notes database is a collection of related documents stored under a single name.

Two Notes core subroutines are responsible for the control of databases: the Notes Storage Facility (NSF) and the Notes Indexing Facility (NIF). As indicated in figure 3.2-17, the NSF and NIF subsystems work independently of the operating systems and of LAN hardware and Software and are used by the Application Programming Interface (API). Notes API is a low level application that allows a software developer to implement ad hoc applications.

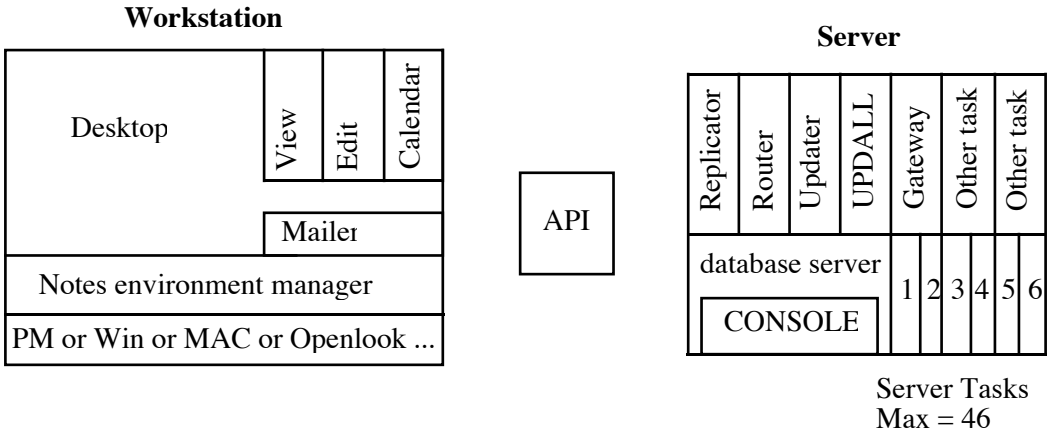


Figure 3.2-17. Notes System

The NSF subsystem is responsible for the following tasks:

- assuring that every note can be identified
- maintaining note content
- storing databases

The NIF subsystem is responsible for all the tasks which keep data notes ordered within each database view. Specifically, the NIF subsystem is responsible for:

- opening and closing databases
- updating the index
- locating notes within and index
- locating index entries
- updating collections

Distributed management of documents

In order to handle documents, Notes uses the following elements:

1. **Database.** Each database resides on different servers with the same identification and can be opened by a specific icon in Notes work-area.
2. **Documents.** Notes documents contain any kind of datum: text, number, graphic information. Users can classify documents by categories, they can

link them within the databases and they can use filters to access and modify them.

3. **Forms.** They include the layout for specific kinds of documents and specify the way in which the user can insert, view and print information. **Fields** are areas defined within forms to contain specific information, e.g. names or postal addresses. This information can be given by the user explicitly or by a formula.
4. **Views.** Notes allows users to see the information contained in a database from different perspectives called views. They can be based on creation time, author, keyword, topic, and so on, and permit the user to classify or select the information of the database. Even though the largest part of Notes views are public, i.e. accessible to every authorized member, the user can create also private views for personal usage; these are not stored in the shared database.

   Database creators handle privileges to access documents by means of ACL (Access Control List). There are seven access levels: manager, designer, editor, author, reader, depositor and no access.
5. **Replication.** Information is shared geographically replicating databases among servers. This procedure is executed periodically as specified by the system administrators. Replication merges modifications made on databases allowing users to access information without being affected by the database managing mechanisms. Notes handles propagation of changes by this mechanism.

Notes users running on DOS or OS/2 platforms can create links between their documents and other applications files by means of the application-linking protocol DDE (Dynamic Data Exchange). DDE is a protocol that allows coresident Windows applications to pass data and commands back and forth and use each other [W91], so that the functionality of one application can be added to another. There are three ways to exploit DDE functionalities. The first is to use DDE functions within code written in the macro language of an application. The second is to use the pull-down menu selection called “Paste Link” contained in the Windows applications that support DDE; in this way changes made in one application are automatically reflected in the other. The third way consists in embedding commands within spreadsheet cell formulas; in this way, users can employ remote reference formulas to link worksheet cells to data in other applications.

Moreover, Notes users can create compound documents via embedding as well as automatically update information via linking by means of the OLE (Object Linking and Embedding) protocol. OLE is implemented at the applications level and allows linked data to “remember” their origins. When users click on a linked information, OLE can automatically call up whatever application was used to compute the original value.

3.2.8.2. The basic MOI

MOI to support communication

The communication within Notes is mainly through data(base) sharing. Moreover, each user has his own Mail Database, in which he receives mail and from which he can compose new messages. No specific MOIs are provided to support work articulation.

MOI to support action

Notes can be considered more as an application development environment than a workgroup computing system [O92]. In this sense Notes supports several types of activities: sales management, development of products and services, customer assistance, etc. It is the database designer who describes how the system supports the action in the various instances. Each user has his own workspace with a 3-D representation of database icons. The workspace is like his desk: every activity is performed using instruments which are placed on it.

The DDE/OLE protocol is the mechanism of interaction between Notes and other applications. Applications can engage in multiple DDE/OLE protocol at the same time.

3.2.8.3. The notation

The notation of communication

The notation used for communication based on database sharing is the query.

Communication between Notes documents and other application files uses the DDE protocol. The program making the request is defined as the DDE client, the program supplying the information is the DDE server.

As already mentioned, OLE is a superset of DDE that provides improved linking capability. It is a compound document[1] protocol in which a piece of data produced in one application becomes a linked or embedded object in another application. When an object is embedded, a copy of the data is made from the source, or OLE server, and duplicated in the destination, or OLE client. There is no connection maintained between the copy and the original. Thus, when a user edits the embedded object, the original source data is unaffected.

The notation of the MOI for action

The notation for action used within Notes is the one commonly adopted by information systems based on distributed database systems.

---

1 A compound document is a document that contains data in any format that is not native to itself. Any kind of data may be incorporated into a compound document. A Notes compound document might include a graphic created with Freelance for Windows with voice annotation.

3.2.8.4. The features of the mechanism

Level of abstraction

The analysis starts by considering the level of abstraction of Notes: in doing that, it is useful to compare the system with Oval. The two systems share some similarities.

Like Oval, Notes can be tailored for many different applications. The templates in Notes are equivalent to object types in Oval; the databases in Notes are equivalent to folders in Oval; and the views in Notes are equivalent to views of collections of objects in Oval. Even though Notes allows very knowledgeable users to do certain kinds of sorting and filtering using views, it does not appear to be designed to make this easy for end users, and it does not have active agents like those in Oval.

Unlike Oval, Notes is based on a database replication algorithm which approximates "live sharing" of documents. It includes even some features such as access controls, free text fields, and calculated fields. On the other hand, even though these features are not currently part of Oval, they would be consistent with its overall user interface paradigm and useful for many other applications [MLF92].

The similarities to Oval lead us to state that the primitives of the notation are not at the appropriate level. That implies that the following considerations are to be interpreted as associated with a development environment rather than an application incorporating MOIs for the work articulation.

Comprehensiveness

Notes provides its users with total generality. Initially the system provides the users with an empty database; then the system administrator creates the forms necessary for the specific environment of the organization.

Malleability

Notes reflects the dynamic nature of workgroups by giving a great flexibility to define them. There is not any geographical or organizational barrier to join a group. Access management for individuals or for groups is handled by the database administrator. A single database can contain shared design elements from multiple design templates. Thus a single database may share some fields with a corporate design template and some with a group-level design template. Automatic updates or unscheduled ones can be made by managers while the work is in progress.

Visibility

The user looses visibility of the role of the actions he performs within the overall procedure. As in common database applications, Notes allows the user to know only who is the sender and whom he has to communicate the results of his work to. The context may be learnt in other ways, e.g. through e-mail messages.

Propagation of changes

Notes uses the process of database replication to distribute and update copies of the same database, stored on both a workstation and a server (or servers). Replication is the process of exchanging data between the local database on one's workstation and the shared database on the server. Notes allows even the remote user to manage the replication.

Working locally means working on local replicas of databases, then replicating the local database with the server database. This propagates the changes back to the server, and allows to receive changes made by others.

Design templates offer database designers a way to maintain consistency of design across databases. A database linked to a design template will be automatically updated by a server task, but a database manager can perform unscheduled updates when they are required.

The degree of openness

Notes users can employ the word-processor, the spreadsheet and the graphical application they like by exploiting Notes capability to import and export the files used to integrate data created with those programs in Notes databases.

DDE allows Notes users running on DOS or OS/2 platforms to create links between Notes documents and other applications files. With a DDE link, changes to the information in the original file made in the source application will be automatically reflected in the target application. OLE is layered on top of DDE. It takes data integration a step further in application integration by allowing users to create real compound documents via embedding, as well as automatically updating information via linking. The applications running on the operating system use the DDLs; this implementation provides a flexible system open to future, non-application-dependent data formats.

API allows the development of specialized C-language written interfaces to support importation/exportation from non-standard environments.

3.2.9. Sharing processes

3.2.10. A short genesis and description

Sharing processes [JMR92], [RJGMN91] are designed to document and structure events. Events drive the evolution of a sharing process and can trigger notifications. Events are caused by agents.

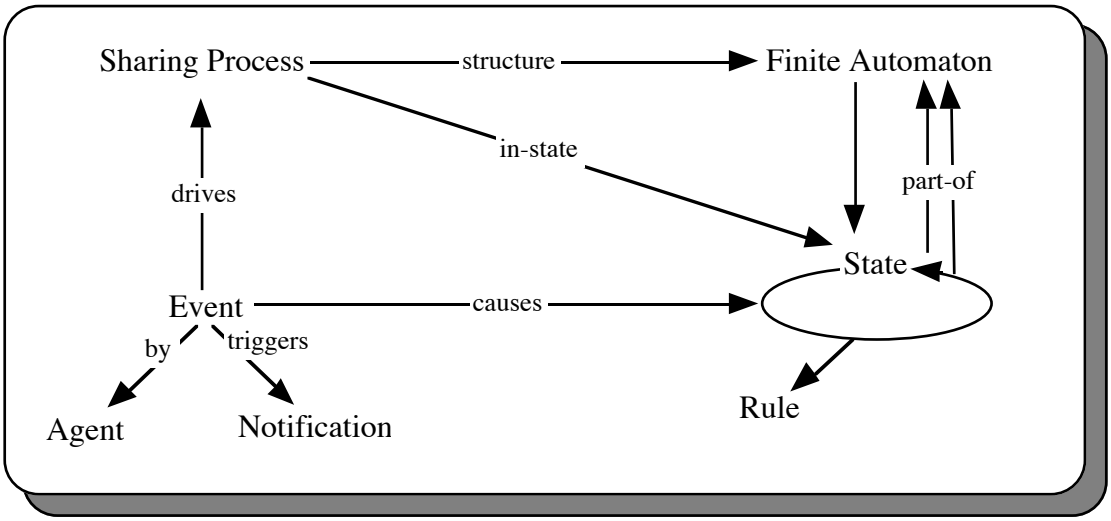


Figure 3.2-18. Sharing process formalism

The mechanism developed is similar to those considered in the previous sections, but in this context a sort of communication support is added to the sharing of tasks and objects. For example, when one user has the task to modify an object originated in a public workspace, the system creates o copy of it in his private workspace which can be modified; but when the new version has to be reconciled in the public domain, the system creates a communication to ask permission or to inform other users. So communication is a support to operations on data.

The novelty of Sharing Processes is that not only usual kinds of resources (design objects) are shared, but also ideas and tasks.

3.2.11. The basic MOI

2.9.2.1. MOI to support action

Task sharing protocols define the cooperative handling of tasks depending on the work setting. One of such protocols is the negotiation, that follows the pattern of conversation for actions proposed in [WF87].

Object sharing allows parallel access to copies of objects distributed across workspaces. Object sharing protocols offer four functions:

- make design objects available in workspaces
- define relationships among workspaces
- define a history of object versions within and across workspaces
- control concurrent access to objects

2.9.2.2. MOI to support communication

The Argument Editor is used to discuss and to take design decisions.

The Contract Assistant is used to assign tasks to people and monitor execution.

3.2.12. The notation

2.9.3.1. The notation of the sharing processes

The notation of the sharing process is a triple <E,N,S> where E is a set of events, N is a nondeterministic finite automaton (NFA), and S is a set of states. The relationships among these components are shown in figure 3.2-19.

A similar notation can be used to represent task sharing and object sharing, by specializing the structure of the automaton. Beside the above mentioned case of conversation for action, we show an example concerning object sharing.

A possible instance of concurrence control specified as an object sharing process is given in Figure 3.2-19; it follows the *copy-merge* paradigm offered by SUN's Network Software Environment.

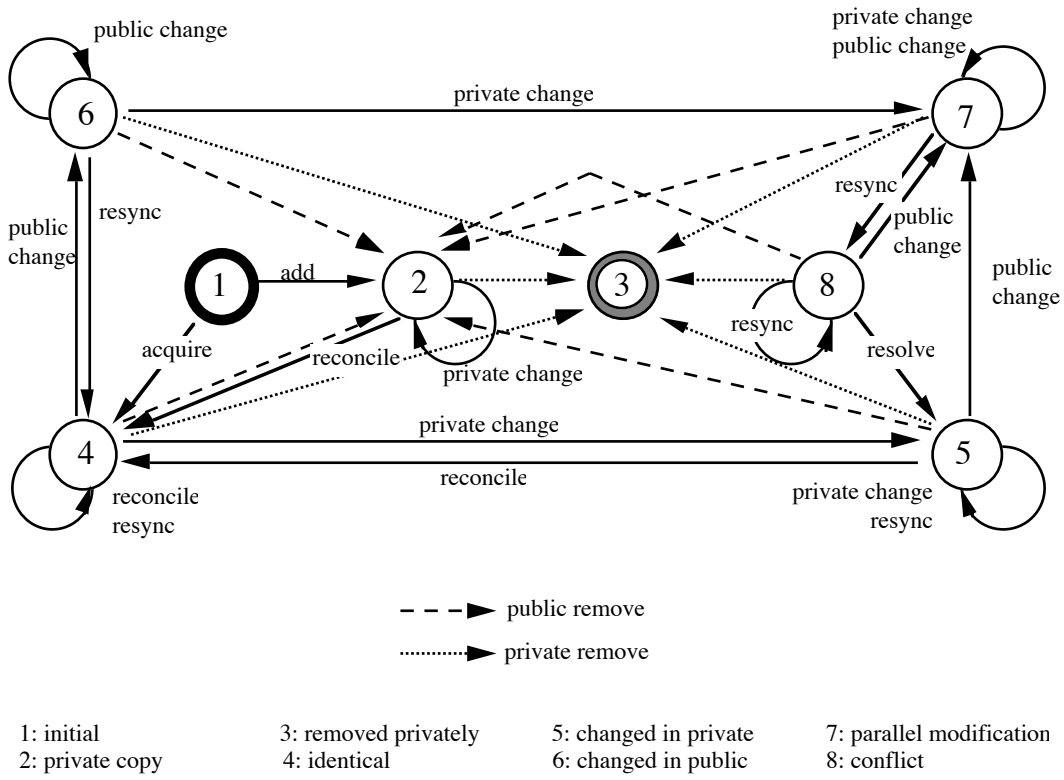


Figure 3.2-19. Finite automaton for object sharing following a copy-merge paradigm

The documentation at our disposal does not allow any precise assessment of the above mentioned MOIs and notations, especially for what concerns the level of abstraction. However, if and when more detailed documentation is available, this application could be of some relevance to the aim of identifying and combining different notations to deal with the different aspects of work articulation.

3.2.13. References

[ADPT92] Agostini, A., G. De Michelis, S. Patriarca and R. Tinini: A prototype of an Integrated Coordination Support. UM-DSI Technical Report, Milano, 1992

[AS92] Ader, M. and K. Srivaths.: WooRKS Programmers' Guide. ITHACA Technical Report ITHACA.Bull.92.U4.#23.3.0, Bull S.A., France, 1992

[ATI88] The Coordinator Version II: User's Guide, Action Technologies Inc., Emeryville CA,1988

[BDT91] Bignoli, C., G. De Michelis and R. Tinini: UTUCS: a Support for Synchronous and Asynchronous Communication. In: Gorling, K.; Sattler, C. (eds): International Workshop on CSCW. Informatik Informationen Reporte, IIR Berlin, Apr. 9-11, 1991, Berlin,1991, S. 74-84

[BS91] C. Bignoli, C. Simone, AI Techniques for supporting human to human communication in CHAOS, in J.M. Bowers and S.D. Benford (eds.), "Studies in Computer Supported Cooperative Work — Theory, Practice and Design", North Holland, Amsterdam, 1991

[DDS89] G. De Michelis, F. De Cindio, C. Simone "Groups in a Language/Action perspective", in Proc. of the 2nd European Meeting on Cognitive Science Approaches To Process Control, Siena 1989

[DDSVZ86] F. De Cindio, G. De Michelis, C. Simone, R. Vassallo, A. Zanaboni "CHAOS as a Coordination Technology", in CSCW86. Proceedings of the Conference on Computer-Supported Cooperative Work, Austin, Texas (1986)

[DG92] De Michelis, Giorgio and M. Antonietta Grasso: "Routines and Conversations", to appear in Structured Programming, Springer-Verlag

[DSVZ88] F. De Cindio, C. Simone, R. Vassallo, A.Zanaboni "Chaos: a knowledge-based system for conversing within offices", in W.Lamersdorf (ed.) "Office Knowledge: Representation, Management and Utilization", Elsevier Science Publishers B.V., North -Holland, 1988

[GS90] P. Gasparotti, C. Simone "A User Defined Environment for Handling Conversations" in S. Gibbs and A.A. Verrijn-Stuart (eds.) "Multi-User Interfaces and Applications, IFIP, North-Holland, 1990

[JMR92] Jarke Matthias, Maltzahn Carlos, Rose Thomas: "Sharing Processes: Team Coordination in Design Repositories", Aachener Informatik-Berichte, Nr. 92-5

[KHKSW91] Kreifelts Thomas, Hinrichs Elke, Klein Karl-Heinz, Seuffert Peter, Woetzel Gerd: "Experiences with the DOMINO Office Procedure System", in CSCW '91. Proceedings of the Conference on Computer-Supported Cooperative Work, Amsterdam, The Netherlands, September 24 to September 27, 1991, ed. by L. Bannon and M. Robinson, Kluwer Academic Publishers, 1991, pp.117-130

[KHW93] T. Kreifelts, E. Hinrichs, G. Woetzel: "Sharing To-Do Lists with a Distributed Task Manager", ECSCW'93

[KPV91] Kreifelts Thomas, Pankoke-Babatz Uta, Victor Frank: "A Model for the Coordination of Cooperative Activities", International Workshop on CSCW, Berlin, 9-11 April 1991

[KW87] Kreifelts Thomas, Woetzel Gerd: "Distribution and Error Handling in an Office Procedure System", Office Systems: Methods and Tools, Elsevier Science Publishers B. V., North-Holland, 1987

[L89] Lotus: "Lotus Notes Users Guide", Lotus Development Corp., Cambridge, MA, 1989

[MLF92] 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, ed. by J. Turner and R. Kraut, ACM Press, New York, 1992, pp. 289-297

[MWFF92] Medina-Mora Raùl, Winograd Terry, Flores Rodrigo, Flores Fernando: "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, ed. by J. Turner and R. Kraut, ACM Press, New York, 1992, pp.281-288

[O92] Orlikowski Wanda: "Learning from Notes: Organizational Issues in Groupware Implementation" in CSCW '92. Proceedings of the Conference on Computer-Supported Cooperative Work, Toronto, Canada, October 31 to November 4, 1992, ed. by J. Turner and R. Kraut, ACM Press, New York, 1992, pp.362-369

[RJGMN91] Rose Thomas, Jarke Matthias, Gocek Michael, Maltzahn Carlos, Nissen Hans W.: "A Decision-based configuration process environment" *Software Engineering Journal*, September 1991, pp.332-346

[SDOP92] Simone Carla, De Cindio Fiorella, Omodei Salé Giuseppe, Pozzoli Alberto: "Integration of CHAOS modules: The User Interface", Technical Report, N° 7/95

[Sea69] J. R. Searle "Speech Acts: an Essay in the Philosophy of the Language", Cambridge University Press, 1969

[Sey92] Seybold Patricia B.: "Business Process Design", *Office Computing Report*, Vol. 15, N°9, September 1992

[TLF88] Tueni, M., Li, J. , Fares, P.: AMS: a knowledge based approach to tasks representation manipulation and organization. In: Proceedings COIS-88, Palo Alto, 1988

[W91] L. Wood: "Navigating Windows's Dynamic Data Exchange", Datamation, vol. 37, n° 9, 1991

[WF86] T. Winograd, F. Flores "Understanding Computer and Cognition. A new foundation for design", Ablex Publishing Corporation, Norwood, 1986

3.3. Aspects, Collage, Active Memory, and OVAL

Hans Andersen, Peter Carstensen, Betty Hewitt, Carsten Sørensen

Risø National Laboratory

3.3.1. Introduction

This document contains a state of the art survey of a number of CSCW tools. The tools are reviewed to identify and describe mechanisms of interaction embedded and supported in the tools.

A Mechanism of Interaction as described in 'Evaluating Computational Notations of Mechanisms of Interaction', (Schmidt et al., 1993) is a mechanism for reducing the complexity of the articulation activities necessary for co-operative work. It is a standardized symbolic artifact which is publicly available and publicly perceptible. It is in a standard format which imposes constraints on the users of it and it can be manipulated independently of the field of work. In many cases it can be seen as a 'plan of action'. Examples would be; timetables, schedules, flight strips, common diaries, some shared computer applications or parts of them etc.

A Mechanism of Interaction can be described in terms of notations. A notation consists of a set of symbolic primitives, syntactic rules and semantic rules. A Mechanism of Interaction has features or characteristics associated with it. These are to do with the conventions surrounding the usage of it and the artifact itself. Features of a mechanism include control, visibility, generality, modifiability, maintainability of context, degree of openness, level of abstraction and its ability to propagate changes.

The framework used for analyzing the different CSCW applications in this paper consists of three major elements: (1) The general function and purpose of the application; (2) The characteristics of the application; and (3) The Mechanism of Interaction embedded within the application and its features.

In Section 2 Aspects and Collage are analyzed. Section 3 presents an analysis of Active Memory, and Section 4 discusses Oval.

In terms of computer based mechanisms of interaction, FTP tools such as Fetch (Matthews, 1992) merely provides the basic functionality for establishing shared data-storages, and is therefore excluded. Designers Note Pad (DNP) (Monk, 1993) is in it's current shape a structuring tool used by a single user, for displaying the same information on more than one screen, thereby allowing several users to edit the same diagrams. It is not relevant to analyze the single-user version with respect to computer based mechanisms of interaction supporting ar-

ticulation in cooperative work. The multi-user version can be regarded as implementing a shared space with no restrictions as to who is doing what, when. We have, therefore, also excluded DNP from this analysis.

3.3.2. Aspects & Collage

From the viewpoint of supporting articulation work, Aspects and Collage are quite similar. Collage contains a subset of the basic functionality provided by Aspects. We, therefore, in this section analyze Aspects, and when appropriate relate to Collage. Unless Collage is explicitly mentioned, all the features analyzed belongs to Aspects.

3.3.2.1. Function and Purpose of Aspects & Collage

Aspects (Group Technologies Inc., 1990) is a simultaneous conference application for the Macintosh. It will allow up to sixteen people, using a network of Macintoshes, to share the writing, drawing and painting activities necessary for creating and modifying documents. The main objective of Aspects is to 'Hold a Conference' for doing a particular co-operative task. To 'Hold a Conference' the participants need to be able to create, join and leave a conference, and conduct a conference.

Before being able to hold a conference using Aspects some articulation work has to be done. All potential participants need to know the time the conference is going to begin, the password if necessary and who is going to be the moderator, as this person needs to be the one to create the conference. The interface tools available for co-operating over Aspects are the Conference Control Window, the Conference Tools Palette, the Chat Box, the menus, the actual documents being worked on and the dialogue boxes generated by Aspects in response to the activities of the users. It is recommended that users of Aspects also have a conference call telephone link, so that all participants can talk to each other. The telephone will therefore be used for much of the articulation activity.

Collage (NCSA, 1992) is also a conferencing tool, but it is particularly aimed at supporting conferences among scientists cooperating on presentation and analysis of scientific data:

> "This tool provides image display and analysis, color table editing, and spreadsheet display of floating point numbers ... *[it]* provides the capability to distribute most of these data analysis and visualization among a number of users." (NCSA, 1992)

Collage consists of image display functions[1], a shared whiteboard, a shared text editor, and a spreadsheet function. The application provides the possibility for conference participants to perform joint analysis and presentation of data. As in Aspects, participants can start a conference, join a conference, participate in a conference, and quit a conference. In Collage each participant decides when a

1 Collage supports manipulation of objects of the following types: 8-bit raster, 24-bit raster, color table, floating point data set, 3D spread sheet, and movies.

given object is suitable for being broadcast to other participants. The tool only supports the distinction between private and public objects. There is no support for turn-taking protocols, moderated conferences, or other types of restriction on access.

3.3.2.2. Characteristics of Aspects and Collage

Both Aspects and Collage allows users to create, leave and join a conference and also supports some of the articulation work needed to conduct a conference.

Creating, joining and leaving a Conference

There are three choices for creating a conference in Aspects, off line, network and point to point. The network option is the most general, so this is the one being considered here. If the network option is chosen, and a conference is not open, then a user can create a conference. They then become the moderator of the conference and will be the only participant of that conference entitled to change the mediation level. Before creating a conference, a user can set the mediation level for a new conference, and decide whether a password is set for the conference or not. Figure 3.3-1 shows the state transition diagram for creating, joining and leaving a conference if a password is needed and if the moderator has asked to be informed of new participants wishing to join.

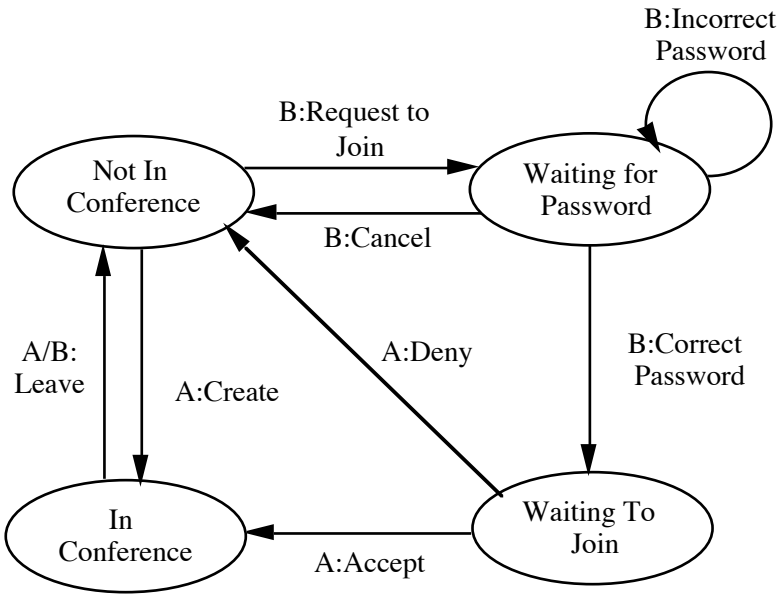


Figure 3.3-1: Creating, Joining and Leaving a Conference

Once a conference is created then other users can join in that conference. If they choose to join and a password has been set by the moderator for a conference, then the person attempting to join the conference must input the correct password. If the password is incorrect then they can just keep trying ad infinitum. If the moderator has asked to be informed when a user wishes to join a conference, then the user wishing to join has to wait for acceptance or denial from the moderator.

If one of the participants leaves the conference then all of the other participants are informed of this. The list of participants will be changed on all remaining participants screens, so that all participants will know who has left and who is still participating. If the moderator leaves the conference then the first person to join the conference after it was created will become the new moderator and all participants are informed of this.

In Collage there are no restrictions regarding a moderator accepting or rejecting a request to join. A participant either starts a conference or invokes the listen command, which will make Collage listen for a conference on the network.

Conducting a Conference

Having started a conference in Aspects the functionality for managing that conference includes changing the mediation level, sending messages, control of editing and pointing, linking and unlinking views, and sending, receiving opening and closing of documents.

Changing the Mediation Level of a Conference

The ability to change the mediation level of a conference uses the Conference Control window, the Changed mediation level dialogue box, and the Conference Tools Palette. Changing the mediation level of a conference has no effect what so ever on the field of work, i.e. the documents being edited.

The moderator can set the mediation level of a conference before creating it and can change it during the conference. They are the only participant allowed to change the mediation level of a conference. The mediation level can be free for all, medium mediation or full mediation. Free for all means that all participants can edit a document at the same time, medium mediation allows only one participant to edit one document at any one time, but several participants can each be editing different documents within the same conference. Full mediation means that there is only one editor at any one time per conference and the moderator assigns edit control. To change the mediation level of a conference, the moderator, can choose one of the three options. In each case the other participants are informed of the change to the mediation level.

The only indication of the mediation level of a conference is by looking at the conference tools palette. Depending on the level of mediation only some of the primitives within the conference tools palette are available. Because of the similarity of the primitives available within the conference tools palette, it is not always easy to distinguish between full and medium mediation.

Sending Messages

The Chat Box, as shown in Figure 3.3-2 is a floating window visible over all other conference documents and is used by participants in a conference to send messages to each other. It is flexible in that any text message can be sent. Each participant has control over opening and closing of the Chat Box which can sometimes cause problems. If Aspects is run and a new conference created, then if

any participant chooses the Chat Box from the Window menu, it will be opened on all participants screens. This is fine except that each participant can close the Chat Box at any time, so that any subsequent opening and closing of the chat box is an individual activity. One participant could be sending messages to another who has their chat box closed, and will subsequently not receive them until they open their Chat Box.

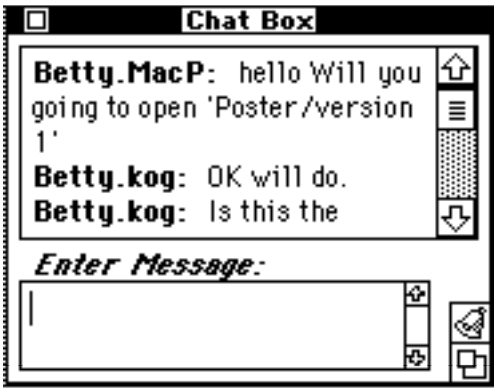


Figure 3.3-2: The Chat Box in Aspects

Control of Editing and Pointing

In full mediation the moderator has control over who can edit. Figure 3.3-3 shows the state transition diagram for mediating editing control when the mediation is full. The moderator can remove or assign edit control, but ordinary participants can request to edit or remove the editing control away from themselves and back to the moderator. In medium mediation, if no one is editing then anyone can grab edit control. If someone is editing another participant can request edit control. If the person currently editing releases edit control it will automatically go to the first person who requested edit control. In free for all, anyone can edit.

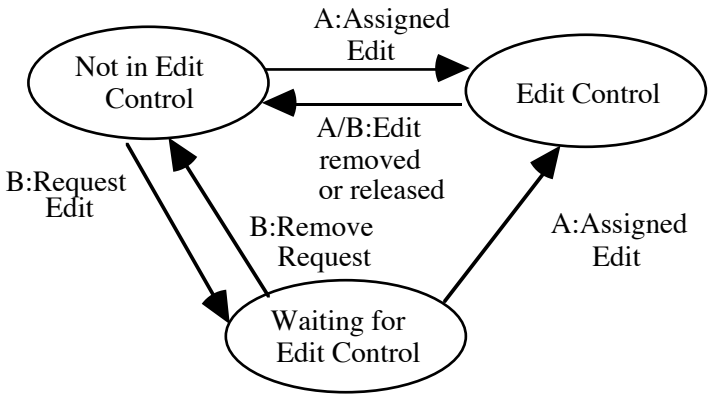


Figure 3.3-3: State transition diagram for editing if full mediation

The Conference Tools Palette shows the editing status of each person. This enables the participants to see who currently has edit control, and who is waiting for control. In medium mediation the first person to request edit control will automatically get it when the current editor releases edit control. The moderator can see who is requesting edit control as the raised hand within the Conference Tools Palette blinks.

Pointing is a useful means of articulation, in that each participant can choose from the select pointer dialogue box, which icon they will use as a pointer (see figure 3.3-4). This pointer is then the one that is used when pointing within an open document. Problems can occur because the choice of pointer icon is not exclusive (i.e. more than one person can have the same pointer icon) and there is no means of showing who has chosen which pointer icon. For a pointer to be a useful articulation tool decisions have to be made before or during the conference concerning who has which pointer. As changing a pointer icon is an individual activity, no one else in the conference will necessarily know that someone has changed their pointer icon.

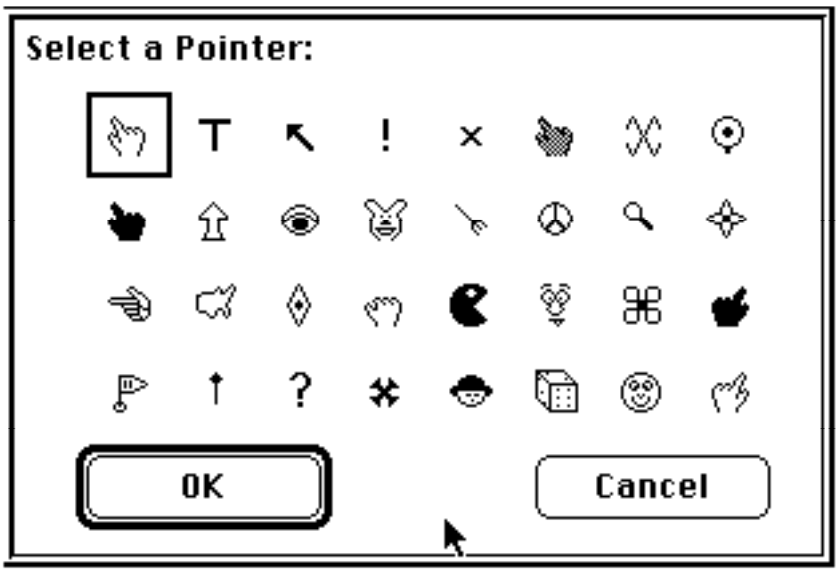


Figure 3.3-4: Pointer Palette in Aspects.

Collage does not offer different pointer icons for the different participants, except on the shared whiteboard, where participants can choose different marker colors.

Linking and Unlinking Views

Having a linked view means that scrolling a shared document or changing the active window can be universal on all linked participants screen's. Only the moderator can choose to link all participants views, but individuals can choose to link with one other person or join an already existing linked subgroup. One problem is that participants cannot make a linked subgroup of more than two people as a single activity. Participants can only be in one linked subgroup at any one time. If participants choose to link views, the icon in the corner of the open document

window which is to be linked, is changed from a closed eye to an open eye. With linked participants, the menu will change from ‘view with’ to ‘who is viewing’, thus showing who is linked. If a person unlinks their view, the other members of the linked subgroup will not know unless they look at their menu. Participants cannot tell whether there are other participants with another linked subgroup. If only one document is linked the other shared documents will have a half closed eye in them. If the Conference Control Window is the active window all document windows will show a half closed eye.

Opening, Closing, Sending and Receiving Documents

Any participant can open new documents or previously edited documents, regardless of the mediation level of the conference or whether they have editing control. They will only be able to edit that document if the mediation is open for all or, unless they request it or if it is free. These newly opened documents will only be sent to other participants if they have made that choice before the start of a conference in Aspects. Whether to accept newly opened documents from other participants is an individual activity. There is no way of finding out which documents each of the participants have unless they are asked. Participants can open and close documents and windows at will and will not know which documents anyone else has open.

3.3.2.3. Mechanism of Interaction and its features

Having looked in some detail at what constitutes ‘holding a conference’ within Aspects and Collage, we now need to extrapolate the Mechanisms of Interaction and their properties. The only candidate for a Mechanism of Interaction within Aspects and Collage is the list of participants of a conference. It is a symbolic artifact that is publicly available and publicly perceptible. All other articulation work carried out within and around Aspects and Collage is not mediated by a Mechanism of Interaction.

Notation

Mechanisms of Interaction have an associated notation, and in Aspects this notation consists of a list of names and the functions that can be performed on it. These functions are, creating a list, adding to the list, deleting from the list, annotating the list and changing the contents of the list. Not all of these activities can be done all of the time. Figure 3.3-1 shows the state transition diagram for creating, joining and leaving a conference. In terms of the Mechanism of Interaction this is the same as creating a participants list, adding to the list and deleting from the list. The state transition diagram shows some of the semantics of the notation, but does not capture the time critical features of it, which is an important part of a Mechanism of Interaction, and needs to be formally specified in order for Mechanisms of Interaction to be made more explicit and used in the design of future CSCW applications.

The contents of the list of participants can be changed as each person has the ability to change their name at will. If a conference consists of three people, Bob, Fred and Melanie, and Fred changes his name to Olivia, then that change is reflected in the Mechanism of Interaction. Depending on the mediation level of the conference, each name has attached to it the type of control they have over the field of work (i.e. the documents being shared). Figure 3.3-3 shows the state transition diagram for managing the change in edit control. This change is reflected in the mechanism of Interaction by each element of the list being flagged as to whether the person concerned has edit control, is not editing, or is waiting to edit. The syntax of the notation is tightly coupled to the interface objects available. For example, to request edit control a participant will double click on the closed pen icon within the conference tools palette. A symbol of a raised hand will appear next to their name in the list of participants within the conference tools palette, and can be seen by everyone and the Mechanism of Interaction will be flagged.

3.3.2.4. Features

Several features are associated with Mechanisms of Interactions: such as visibility, control and level of abstraction, as described in Part 2.

Visibility

The notion of visibility concerns whether the users of the mechanism of Interaction can perceive the notation in its entirety i.e. whether the symbols, semantics and syntax of the notation can be seen and whether they can see the effects of their actions. □The primitives of the notation i.e. the actual names of the participants in the participants list can be seen from the Conference Control window, the Conference Tools Palette and within the ‘send document’ menu choice. The Conference Control window just lists the participants of the conference, but within the Conference Tools Palette, if the mediation level is full (or medium) then it is possible to see who has editing control, who is not editing and who is waiting to edit. It is however not very obvious whether the mediation level is full or medium, and changes to the mediation level of a conference are not very apparent. For example, it could be changed by the moderator and unless the participants are aware at the time of the change, there could be some doubt as to which mediation level is currently being employed. In the ‘send document’ menu choice it is possible to see who has or has not a copy of the document that is going to be sent. It would be useful to know who the moderator of a conference is, however the mechanism of Interaction does not show it anywhere in the list of participants i.e. it is not visible although the system knows about it. As mentioned above the syntax is tightly coupled to the interface so the usage of the mechanism of Interaction is fairly obvious, however the semantics is not shown and the users do not necessarily know what is going to happen when for instance they request edit control. Using Aspects over time will allow the users to learn what happens, but to a novice user there is no mechanism for letting them know what is going to happen.

Level of abstraction

For a Mechanism of Interaction to be useful for articulation of the work its level of abstraction needs to be considered. Within Aspects the list of participants can be seen as a fairly general Mechanism of Interaction, which could be and is (cf. Collage) used within many different applications. However the level of abstraction does not help in articulating the work and most of the articulation work has to be done either within the field of work itself (i.e. pointing within a document), or using different modes of interaction that do not have a symbolic artifact connected to them, such as the telephone, the use of the different mediation levels, the chat box and the different ways of being able to view documents. Perhaps the ability to build Mechanisms of Interaction within a conferencing system for a particular domain would promote the ability to articulate the work better.

Control

There are two ways of considering the control over a Mechanism of Interaction. Control over whether and how it can be altered to suit the needs of the users and the domain, and control of its usage. In Aspects there is no way the users can adapt the list of participants to suit particular circumstances although as stated above, the names of the participants can be changed. As the list of participants is a central feature of Aspects, it is not possible to circumvent it and use another means of contacting a participant. For example it is not possible to have a chat box which is shared between only two people within the conference.

The amount of control participants have over the mechanism of Interaction is limited to being able to change their own name within the list of participants delete or add themselves to the list. No one can add or remove anyone else from this list. These changes are propagated to all participants. The notion of being able to make lasting changes to a Mechanism of Interaction, within the context of Aspects does not make sense, as the list of participants is only a temporary mechanism for use within one conference. It is not possible to save the context of a particular conference and use it and the associated list of participants again. So information concerning who was in a particular conference, what was done during the conference, when a document was collaboratively edited etc. is lost when a conference is closed.

The first person to create a conference becomes the moderator. However this participant is in charge of the list of participants only in so far as they are able to stop other people from joining a conference. During a conference there is no explicit mechanism for changing the moderator. If the moderator leaves during a conference, the first person who joined the conference after it was created becomes the new moderator. This is probably not a problem if the number of participants of the conference is small, but if there were many participants, being able to choose who has control over the conference might be a necessary prerequisite for particular conference set ups.

3.3.3. Active Memory

Active Memory (Denizon, 1990) is a calendar and planning application. It provides general facilities for defining activities and deadlines, and distributes these among group members. Active Memory is intended to be used both as a tool for individuals and groups, and is run on a networks of Macintosh computers. The architecture is distributed (there is no dedicated server) and all involved computers are used both as servers and as clients. The facilities for information sharing are based on the existing Macintosh file sharing facilities (system 7). These architectural decisions imply that all set up, etc. can be done individually by the users themselves without involving some kind of organizer or system administrator.

3.3.3.1 Function and Purpose

The overall idea of Active Memory is to support activity planning, both regarding individual activities and activities involving several people. Afterwards these deadlines, etc. can be checked. The planning also covers day-to-day planning, e.g. it is possible to look up in the calendars and see when a given group of people could have a meeting without conflicting with other things.

Active Memory handles a “memory” (a plan file) for each user/actor. Here planned tasks, activities, etc. can be defined by means of types of activities, contents, period of time, and participants. Each activity can be prioritized. Furthermore, deadlines for the activities and the frequency for repeating the deadline warnings (at the receiver’s side) can be defined. The users themselves define how and when reminders are activated. Each activity is shown and visualized as a line in a spread-sheet like table (the planning panel). The textual information on each activity (line) can be viewed as a whole in a pop up window. Figure 3.3-4 illustrates the lay-out of the planning panel.

The user can identify up to sixty four other Active Memory users (or groups of these) to whom information defined in the task planning are sent. To send information, an activity (a line) is defined, containing type, information, deadline, involved persons, priority, etc., and is transmitted to the involved actors. The receiver is then notified and can either refuse to accept, return the activity or accept it. If accepted by the receiver it becomes a part of his or her “personal” plan.

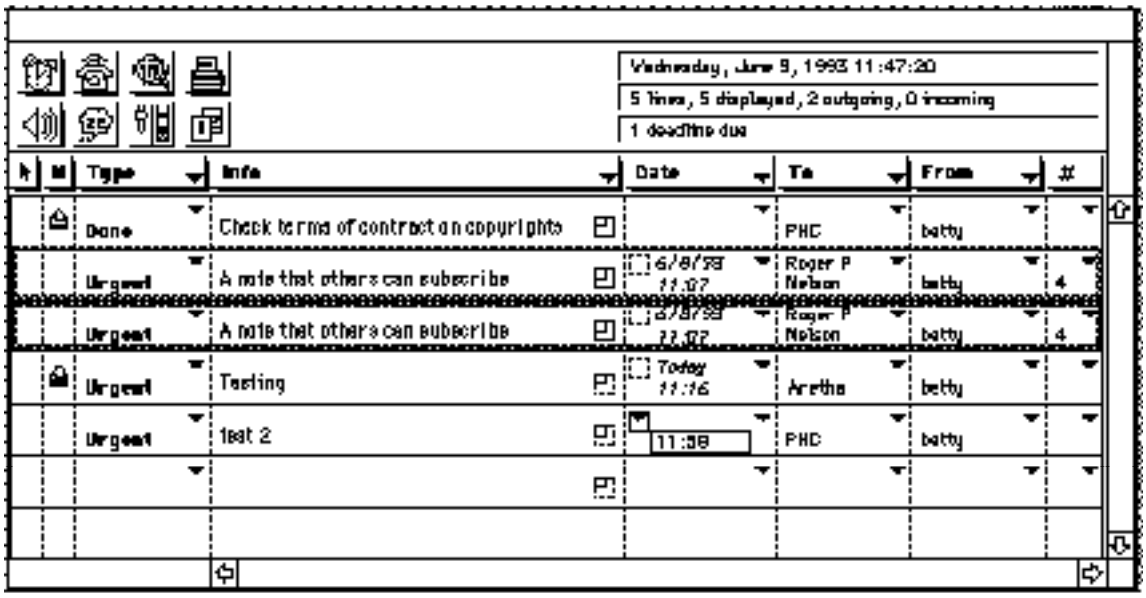


Figure 3.3-4: The planning panel in Active Memory. In this each user can view all the activities he or she is involved in. He can see who has initiated them, who is involved, their deadline, and their priority. If one of the other users sends a note (e.g. informing the user to perform a given activity) it will be shown as a line in the planning panel.

Active Memory can be used for publishing different types of information. Every user in the network can offer and maintain a publishing item. Other users can then subscribe to these items. Each item a user subscribes to is represented as a line in the planning panel. Whenever the owner of a publishing item updates the information the subscribers are notified. This makes it possible to establish close-to real time synchronous communication between a group of users.

As illustrated in the figure below each publishing item is represented as a normal Macintosh file on the owner's computer and the other users can (via their Active Memory session) subscribe to these.

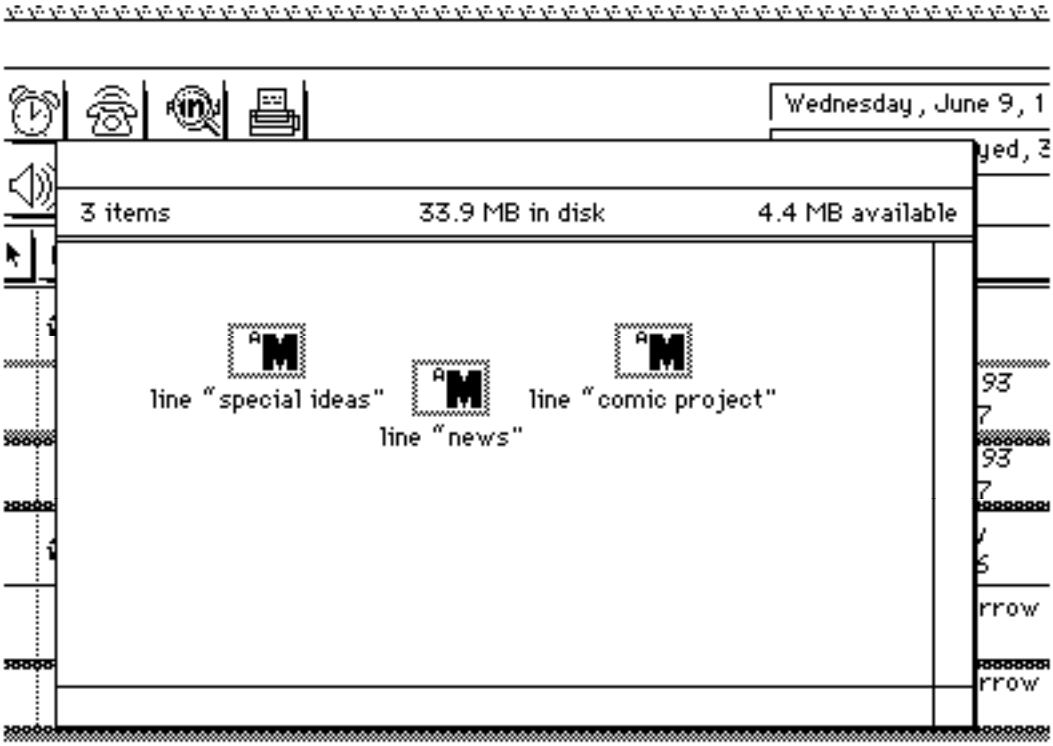


Figure 3.3-5: Published information items are handled as ordinary Macintosh files to which others can connect.

When the user views the planning, etc. they either use the planning panel or a calendar window. To see information concerning a specific day, the day-field is activated and activity lines with relations to this day are shown in the planning panel.

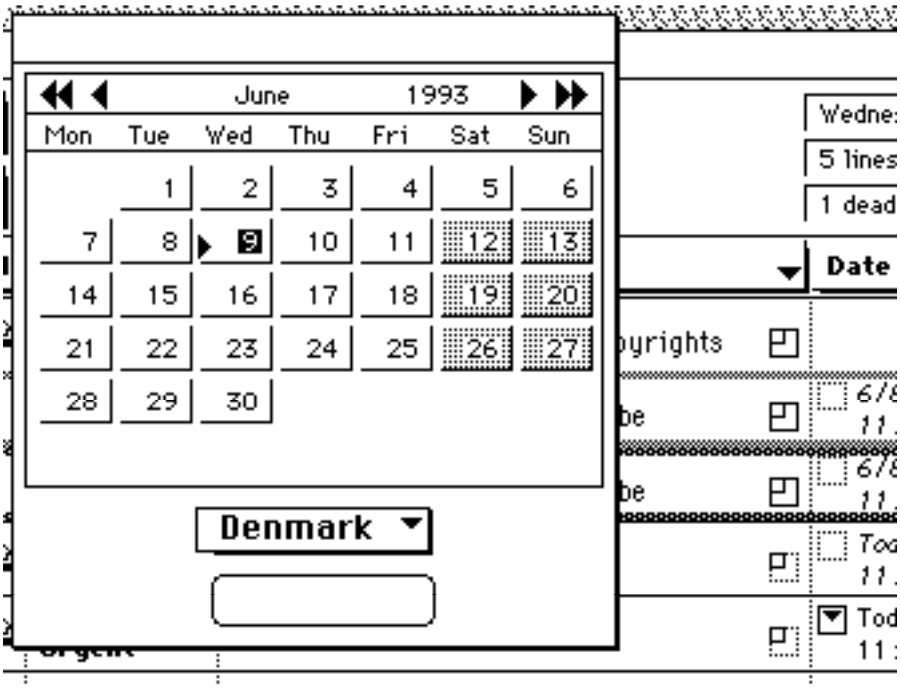


Figure 3.3-6: The day-to-day calendar used when looking for free time or getting an overview of activities of specific dates.

3.3.3.2 Characteristics of Active Memory

Most of the functionality in Active Memory regards personal planning and personalization of how the planning panel will be viewed, e.g. selecting criteria for which activities are shown in the planning panel, defining how deadline reminders must provide alarms, defining sorting order for activities, etc. As this section is concerned with facilities and mechanisms for supporting coordination and articulation work, these aspects will not be addressed in this paper.

Establishment of users, groups, and interconnections

Establishing who are members of the cooperative groups, and to whom a specific user is connected, is controlled and handled by the individual user. There is no support for central administration, set-up, and control of user naming, etc. A user can therefore send information such as activities (cf. next section) to a list of receivers (the “to-list”), without any control or feedback on whether those users exist or not.

To support the maintenance of the user groupings and connections between them Active Memory offers the following functionality:

- Change user ID.

  When the user starts up his or her computer, Active Memory informs the network that an Active Memory user having a specific name is now running the application. The name can at all times be changed by the user. Other users are not notified except if they actively ask who is connected. When

starting Active Memory for the first time the user name is equal to the Mac user name distributed on the network (Appletalk) unless the user changes it.

- Define search area.

  Each active user can define which network areas (Appletalk zones) should be searched when a message is send to a specific supplier. This regards only when sending information. When receiving, the sender's set up is used, i.e. if user ABC and HIJ are placed on different zones, user ABC might be able to "see" and send messages for user HIJ whereas user HIJ cannot reach ABC.

- Ask who is connected.

  A user can ask who is connected at the moment. The specified zones will be scanned and the names of all Active Memory users will be returned.

- Define names for "to-users".

  The user can add and delete names to the internal list of Active Memory users that messages (information on activities, cf. figure 3.3-4) can be sent to. The list is used as a pop-up list when defining the receiver of a message. The names can be either user names of distribution lists. Each user has his own internal list containing up to 64 user or distribution list names. The user has total freedom to define the names of potential receivers and there is no control of whether or not the names correspond to actual Active Memory users.

Any coordination needed for the setup and administration of Active Memory is entirely in the hands of the users. There are only few attempts to introduce a standardized format that provides affordances for the aspects of coordinating and administrating articulation work (Schmidt et al., 1993).

The objects in this part of the application are only addressing the structure of the work arrangement, namely information on names of actors and their computers, and information on whether they are actively running the application or not at a given time.

Definition and description of activities, and distribution of these

In Active Memory the central feature regarding articulation work, is the support for defining activities. Each user can from the planning panel (see figure 3.3-4) describe an activity using a predefined format (a line in the panel). The description can then be used as a personal entry in the user's panel as part of his work plan, and can be distributed to one or more Active Memory users. Each of the users receiving the activity description (message) can then decide whether they will accept, refuse or return the message. If the user accepts the message it is integrated into his or her planning panel.

To support these activities Active Memory offers the following functionality:

- Create an activity.

  Having the planning panel active the user can define new activities or redefine activities already existing in the planning panel. The activity contains a

type-field, an info-field, a deadline field, a to-field, a from-field, and a priority-field:

* The type-field can either be selected from a pop-up menu or described in free text. The entries in the menu are defined by the individual users, i.e. there is no central register.
* The same goes for the to-field and the from-field. The default in the from-field is the current user name.
* The info-field is a free text field. The first line is shown in the panel.
* The contents of the priority-field are either empty or 0 to 9.
* The deadline can be defined either as a relative date (e.g. 3 working days from now) or an absolute date (e.g. Monday 5. June). Also a specific time of the day can be specified. Furthermore the user can define whether exceeding a deadline causes an alarm to be sounded at the receivers end. The sound of the alarm and the repeating frequency can be defined. All these parameters are either selected from menus or entered by the users themselves.

All fields can be edited and changed at any given time. Activities defined by other users, received, and integrated into the user's planning panel can be edited, e.g. the individual user can change the priority or the deadline of an activity in her panel.

- Distribute activities.

  After an activity is defined it can be sent to the Active Memory users defined in the to-field. If the to-field is empty the user is requested to fill out the field. The activity is then transmitted to the specified users. The user can then afterwards see the status of the activity (described below). The activity is marked as outgoing on the originator's panel.

  If the originator afterwards edits a transmitted activity the changes are automatically distributed.

- Recall or cancel activities.

  The originator of an activity can cancel the activity. This implies that it is removed from the planning panels it was integrated into (it is not clear how the protocol works if users are switched of when this function is activated).

- Receive activities.

  The user is notified when an activity has been transmitted and now is received. The user can then see the description and decide on one of the following:

  * Accept the activity. The activity is then integrated into the user's planning panel. The activity is marked as ongoing.
  * Refuse to integrate the activity. The originator can then see this via the activity status.
  * Edit the activity and return it to the originator.

- See the status of a transmitted activity.

The originator of an activity can at any time see who — among the users having their computer running — has received the message, looked at it, and accepted it (it is not clear how the protocol works if users are switched of when this function is activated).

The definitions of activities are based upon a standardized format providing — at least to some extent — support for the articulation activities. These activities can be manipulated individually by the users, but only if this information is altered by the originator will other users be notified.

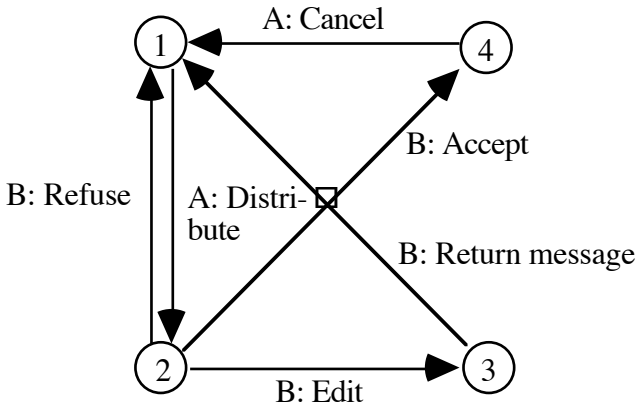


Figure 3.3-7: A simplified diagram of the possible states and activities in an interaction between two Active Memory users. A is distributing an activity and B is receiving it.

The objects provided and handled in this part of the application are addressing several relevant aspects of articulation work. Information on activities, agents, classes of activities (types and priority), and some aspects of the time dimension are modeled. However, the relations to the actual domain and work arrangement are not addressed at all and there is no support for globally defining structures relating to the domain or the work arrangement.

Publishing information items and subscription

The third mechanism supported by Active Memory regards facilities for the users to publish information that other users can subscribe to. All users can define an activity field as published information. Other users can then subscribe to the information. Every time the originator changes the contents of the information field the subscribers are notified and can immediately read the updated content.

To support these activities Active Memory offers the following functionality:

- Define publishing information. This is identical to defining an activity.
- Make the information available.

  The originator defines the activity as being a publishing information field. He is then requested to define where the file containing the information must be placed. This has to be in a folder which other users are allowed to address via the file sharing facilities of the Macintosh. A publishing infor-

mation field is represented as an activity line in the originators planning panel.

- Subscribe to the information.

  All users who have access to the shared folder can request to subscribe to a publishing information file. The file is selected through the normal finder on the Macintosh. The publishing information field is represented as an activity line in the subscriber's planning panels. He can then at any time expand the information field and read the information.

- Edit the information.

  The originator can change the contents of the information field. When this is done the subscribers are notified by a small flag that is set in the actual information field in the planning panel.

- Cancel publishing

  The originator can at any time cancel the publishing. The file is then deleted and the relevant activity line is removed.

- Cancel subscription.

  The subscriber can at any time cancel his subscription. The relevant activity line is removed from his planning panel.

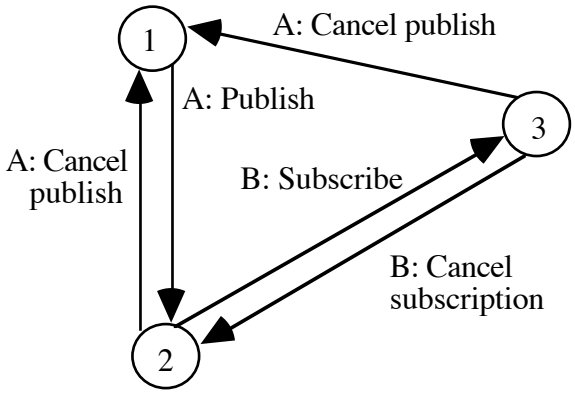


Figure 3.3-8: Diagram illustrating publishing and subscription. A is publishing and B is subscribing.

This mechanism only supports articulation in a very informal manner. There is no standardized format for the information and no support for defining who receives the information, how, and when. The structure does not reflect typical structures in the work arrangement or the domain.

3.3.3.3 Mechanisms of Interaction and its features

Several of the aspects of Active Memory can be interpreted as mechanisms of interaction supporting articulation work. The entries (lines) in the planning panel can be interpreted as a mechanism of interaction on an activity task level. The entries in the "joint calendar" can also be interpreted as a mechanisms of interaction

on planning activities, i.e. on an organizational level. Finally the publishing and subscribing facilities can be interpreted as a mechanism of interaction. When discussing mechanisms supporting articulation work we also need to discuss the associated features.

One important aspect is the domains in which the application could be useful. Active Memory provides some very general — but also very primitive — facilities for planning activities and exchanging this information among the group members. All types of activities, etc. in the planning items are user defined. This increases the generality and the application can probably be used in many domains. There are however no, or very little correspondence between the structures supported and domain specific items. This connection — which is an important requirement in articulation support tools — must be specified outside “the tool”.

As mentioned in (Schmidt et al., 1993) the two major properties to address when discussing mechanisms of interaction are visibility and control.

Visibility

Presuming the user has activated Active Memory, he or she is actively informed when relevant changes and messages occur. Regarding the status of the other users and similar information the user are not informed at all.

Active Memory handles items on a rather high level of abstraction. The concepts fit quite well into the natural division of tasks, and the user can define the abstraction level of tasks and activities as needed.

We must conclude that both the structures and the manipulation of these mechanisms of interaction are easily visible and accessible to the user. Also the content of the structure is easily accessible. However, status and administration information are totally left out.

Control

As mentioned, the types of activities and information distributed are defined by the users. So at the level of content of the activity description the users are in total control, as long as they stick to the defined structure.

The structure can, however, not be changed and the central functionality and mechanisms in Active Memory cannot be modified.

Furthermore, the administration and control of the setup (e.g. use of distribution groups, conventions for activity definitions, conventions for use of publishing items, etc.) are not supported at all. The control of these is placed in the hands of the individual users. There are no support of the coordinating activities and, as mentioned above, changes in status, user names, etc. are not forwarded to the other users.

3.3.4. Oval

3.3.4.1 Function and Purpose of Oval

Oval[1] (Fry et al., 1992; Malone et al., 1992) is a new version of the Object Lens software from the Center for Co-ordination Science. MIT Oval is a 'proof of concept' demonstration system rather than a finished application. Its main purpose is to show how the basic building blocks in the system work together to enable integration of many types of information and applications, and at the same time provide users with the ability to tailor this integration to their actual needs. In this way it can be seen as an example of an end user information management programming tool. The system claims to allow users to create and modify co-operative work applications for themselves without always requiring the help of professional programmers.

Oval integrates features of object-oriented databases, hypertext, electronic messaging, and rule-based agents. As Oval combines different applications into a single integrated environment, people can use one interface for reading mail, querying databases, creating applications, etc.

The system aims to allow people to keep track of and share knowledge about messages, people, tasks, projects, companies, meetings and other things with which they work. The system lets people create various kinds of programmable agents to help them organize and respond to this knowledge. For example, people can use hypertext links to represent relationships between a message and its replies, between people and their supervisors, and between different parts of a complex product. They can also use programmable agents to find electronic messages in which they are interested, to notice overdue tasks, and to notify people of coming deadlines.

3.3.4.2 Characteristics of Oval

Oval is based on four key elements to be used in creating a variety of customizable applications. These primitives are Objects, Views, Agents and Links.

Objects can be characterized as template based data sheets ordered in a hierarchical manner. In the data sheets it is possible to add and manipulate fields and field values, and perform various kinds of action upon fields. Throughout the tree structured hierarchy new subtypes of objects inherit fields from previous objects. The field information can be in the form of free text and/or links to other objects.

Views can be characterized as semi-editable tables in which it is possible for the user to present objects in different ways. The user can select to view objects in edit boxes, tables, calendars, emacs editors, networks, matrixes and templates. Furthermore the user can decide upon what information to show and how to sort it according to object type, keyword, text and name. Views can be customized to display various kinds of information in any object field.

---

1 Trademark of Massachusetts Institute of Technology, Cambridge, MA.

Agents can be characterized as rule-based programs. The user can program a set of production-rules (IF-THEN rules) to automatically perform actions upon series of objects at specified times or events. When a rule is applied to a set of objects, each object is first matched against the description in the IF field of a rule. If the match succeeds, the action in the THEN part of the rule is executed. Possible actions are: Move, copy, copy item, cut, add value, show, send, send item and run script.

Links can be characterized as representing hypertext like relations between objects. Knowledge represented in links can be used by rules and in creating displays.

The objects, views, agents and links, constitute a very general notation that can be used for constructing a wide variety of different applications. A number of systems and applications has been emulated and created[1]:

Emulation of the function of existing systems

- Argumentation support (gIBIS)
- Conversation and task management (The Coordinator 1.42)
- Semi-structured information management (Lotus Notes)
- Rule based mail sorting (Information Lens)

Applications created

- Software design support system (3-C Demo, 1.1)
- Management support (Cereal Demo)
- Integrated problem solving (Car Door Design 1.1)
- Support of vendor selection (Vendor Selection 1.2)
- Internal personnel expertise directory (Contact Point 1.3)
- Time sheet routing (Paper Work Flow 1.2)
- Project tracking (Project Time 1.2)
- Purchasing approval support (Order Approval 1.5)

3.3.4.3. Mechanisms of Interaction and its features

New primitives can be developed on different semantic levels using the basic primitives. Developing new primitives at a correct level can be considered as building Mechanisms of Interaction. However as it stands, the basic primitives in the notation are not expressed at the appropriate semantic level regarding the work context i.e. the basic building blocks do not incorporate Mechanisms of Interaction. Objects, views, links and agents do not seem to be a natural set of concepts for a co-operative ensemble trying to co-ordinate distributed co-operative work activities.

---

1 Extracted from the Oval Applications Catalogue delivered together with the Oval application package.

Since users themselves are able to construct and modify Mechanisms of Interaction the malleability of these is quite obvious. In the emulation of The Coordinator, for example, users can manage the notation to define new conversation structures and rules. Though Oval supports multiple users to modify Mechanisms of Interaction it does not support the co-operative process of doing so. Of course this gives rise to some problems regarding control of propagation of changes and management of consistency. The underlying model of the applications created and the incorporated Mechanisms of Interaction are completely visible to members of a co-operating ensemble.

It is possible to save a collection of created objects that can be opened by other people later on. Furthermore it is possible to mail a collection of objects back and forth. Oval also provides some kind of version control, so the facilities are there. But in what way changes in the applications created using Oval, will be propagated and agreed upon throughout the co-operating ensemble is not clear. Perhaps changes will be propagated following the principle of "survival of the fittest" via informal communication channels — "I have heard from someone, who told me he knew of someone who have heard....." Certainly some kind of control mechanism is needed, but again such a mechanism has to be malleable to allow manipulation and management by the co-operating ensemble.

Organizational context in the form of relations between roles, rules, responsibilities and competence, as well as tasks, messages and meetings can be presented and maintained via Links from various fields between different Objects in a hypertext like manner. The maintenance can be performed by manipulatable Agents, and users can create customized Views to display the context. But the maintenance of this mixture of features seems to be quite complex in itself and surely will require some kind of control mechanism to produce a reasonable result.

Files generated by Oval can only be opened from within Oval. It is possible to use Cut and Paste regarding text-strings. The email function uses SMTP and requires a UNIX Pop-server and Techmail installations. It is not clear whether it is possible to create, for example, text files in applications created by Oval, that can be imported by other systems, but in the Coordinator emulation made in Oval this feature is not supported.

As a 'proof of concept' tool Oval must be said to fulfill its purpose. It is possible to create demonstrators of computational Mechanisms of Interaction. Navigating through Oval is a difficult task. The structure is quite complex and it would be advantageous to have giant screens to handle the structure and a good sense of how to simultaneously manage multiple windows. To use the conceptual framework in Oval to create new types, end user programming tools might seem a fruitful idea. But it must be stated that users will have to be given a huge amount of time to tailor systems to the work at hand. Instead of doing work users will program systems. They will sit there tailoring and tailoring and tailoring and tailoring and....

3.3.5. References

Denizon, Christine: *Active Memory*, ASD Software, Inc., Montclair, California, 1990.

Fry, Christopher, Kum-Yew Lai, and Thomas Malone: *Oval*, v. 1.1, Massachusetts Institute of Technology, Cambridge, Massachusetts, 1992.

Group Technologies Inc.: *Aspects — The First Simultaneous Conference Software for the Macintosh*, Group Technologies, Inc., Arlington, Virginia, 1990.

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*, ed. by J. Turner and R. Kraut, ACM Press, New York, 1992, pp. 289-297.

Matthews, Jim: *Fetch*, v. 2.1, Trustees of Darthmouth College, Darthmouth, 1992.

Monk, Simon: *Using mini-DNP on the MAC*, Lancaster, 1993.

NCSA: *NCSA Collage (Collaborative Numerical Analysis and Visualization Environment)*, v. 1.0, University of Illinois at Urban-Campaign, 1992.

Schmidt, Kjeld, Peter Carstensen, and Betty Hewitt: *Evaluating Computational Notations of Mechanisms of Interaction: Notations, Dimensions, and Features*, Risø National Laboratory, Roskilde, Denmark, May, 1993. [COMIC-Risø-3-8].

3.4. Quilt, gIBIS, Answer Garden, and The Maintenance Support Tool

Tom Rodden

University of Lancaster

3.4.1. Introduction

These notes focus on a class of CSCW application which exploit the concepts within shared hypertext to support a range of different co-operative activities. These systems are analysed with a view to highlighting the various mechanisms of interaction embodied within the cooperative applications. Many CSCW systems developers suggest that shared stores which offer hypertext facilities will form a central component of future CSCW applications. As a result of their features of persistence these systems are increasingly been referred group or organisational memory.

The systems considered within this document each exploit different features of shared hypertext technology. The term *hypertext* describes any system employing non linear structuring of text, graphics, and other media. Hypertext systems normally form linked network structures with data (usually text or graphics) in the nodes and occasionally typing information on the links (see (Conklin 87) for an overview of hypertext systems). Hypertext documents resemble nets of connected nodes with each link between nodes denoting an association between the information held in the nodes (Figure 3.4-1).

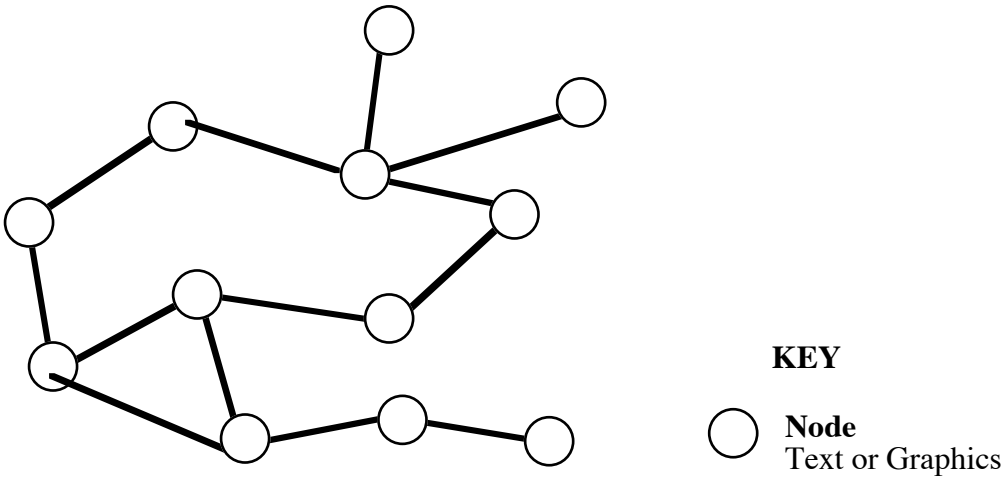


Figure 3.4-1 A typical hypertext network

Hypertext systems are successful in structuring information in such a manner that the user's ability to process it is enhanced (Perlman 90) A general view of the co-operative systems considered here is as multi-user hypertext systems, where the hypertext document (or network) is constructed by a number of users adding nodes to the network in an independent manner. However, the provision of tools which deal explicitly with the interaction arising in collaborative hypertext settings are relatively few. Two general classes of systems can be highlighted as been based on shared hypertext systems co-authoring systems and argumentation or design rationale systems. Design rationale systems have recently been highlighted for their ability to provide a group or organisational memory. However, shared hypertext systems have also been used to represent shared memory without the use of an explicit rationale notation. In the following, the applications will be considered in terms of a number of related issues:

- a short description of the systems
- the basic MOI;
- the notation al features of the system
- features of mechanisms , and among the others
  - the context they maintain
  - the degree of openness (in the sense above).

A large numebr of multi-user hypertext systems have emerged, each with their own individual peculiarities. These include Intermedia (Catlin 89), Matrix(Jeffay 92), KMS (Yoder 89)and Notecards (Halasz 87). Considerable overlap in functionality exists across these applications. Consequently, within this document shared hypertext systems will be considered under three different catagories of use, Co-authoring systems, design rationale and group memory. Each of the catagories will be considered by describing an example system. Finally, these different systems are complemented by a consideration of a more general shared hypertext system developed at Lancaster which supports the work of maintenance engineers.

3.4.2. The Quilt Co-Authoring System

3.4.2.1. Description

Co-authoring systems are a class of systems which apply the principles of hypertext technology in a cooperative setting. Co-authoring systems exploit a model of cooperation irrespective of the physical proximity of the users. Co-operation is assumed to occur over an extended time period with group members working in a time independent manner. The Quilt co-authoring system (Fish 88,Leland 88) developed at Bellcore is representative of the general principles used by most co-authoring systems.

Quilt is aimed at supporting the work of existing authors who have an established pattern of cooperation. The premise behind the application appears to be

that Quilt is used after some form of planning/ brain-storming meeting which establishes the nature of the cooperation between the authors. Thus Quilt acts as a framework to support the co-ordination needed for the group to author the document.

3.4.2.2. The basic MOI

Quilt is designed to be editor independent and provides a framework for the sharing of comments and annotations. A document in Quilt consists of a base and nodes linked to the base using hypertext techniques. The aim is that these nodes act in a similar way to paper notes, Post-its, and margin comments in paper documents.

The general principle of cooperation in Quilt as in most co-authoring systems is that the users read through a publicly available document annotating the document to reflect their comments. To achieve this the designers of Quilt believe that a general hypertext graph is unnecessary and a tree structure is sufficient for supporting most cooperative authoring. This restriction on the more general hypertext linking techniques imposes a particular style of cooperation and interaction. The basic mechanisms used to co-ordinate the cooperation occurring is the definition of different access to document nodes. These access rights delimit a set of social roles within the system.

3.4.2.3. The Notation

The major notational feature of Quilt is the association of different type information with different nodes in the hypertext network representing the co-authored tree which is the focus of the group work. At any time a Quilt comment tree will consist of :-

- A current base document consisting of the text and other materials the authors consider publicly available.
- Revision suggestions, which are available in a form where users with appropriate permissions can swap them with existing document paragraphs.
- Comments with an associated set of permissions which in turn distinguishes them to be one of three different types of comment.
    1) *Private Comment* visible only to the creator.
    2) *Public comment* can be read by anyone with permission to read the parent
    3) *Directed message* whose existence is displayed only to named individuals or groups. Directed messages may make use of a notation facility to inform users of their existence.

Quilt provides facilities for making the document comment tree more active by the attaching triggers which can cause a range of possible commands to be executed. A possible use of triggers is maintaining deadlines for document produc-

tion. Quilt also provides facilities for the tracking of user actions and a message system to support their co-ordination.

Quilt supports six different types of nodes within the hypertext network. In addition, to the base document creator Quilt distinguished three social roles which delimit the particular access rights for reader, commentator and co-author. These different roles are used as means of determining the operations which can be used on different nodes within the Quilt document tree. The following table summarises the access permissions within the system.

	Base Document	***Suggested Revision***	***Public Message***	***Directed Message***	***Private Comment***	***History***
Create	Co-author	Co-author	Commenter	Commenter	Reader	Co-author
Modify	Co-author	Creator Co-author	Creator	Creator	Creator	
Delete	Co-author	Creator Co-author	Creator	Recipient Creator	Creator	
Attach Revision	Commenter	Commenter	Commenter	Creator Recipient	Creator	
Attach Comment	Commenter	Commenter	Commenter	Creator Recipient	Creator	
Attach Message	Commenter	Commenter	Commenter	Creator Recipient	Creator	
Attach Private Comment	Reader	Reader	Reader	Creator Recipient	Creator	
Read	Reader	Co-author	Commenter	Creator Recipient	Creator	Original Permissions Apply

Figure 3.4-2 Access Rights in Quilt

The permissions outlined in Figure 3.4-2 delineate the cooperative activity of users Quilt in the collective development of a document. These permissions are fixed and both the identified social roles and types of comment are fixed and not readily alterable by users.

3.4.2.4. The features of the mechanisms

Generality

The principle of representing a co-authored document as a hypertext structure has general applicability and Quilt provides an editor independent framework to support the generation of a shared document. The system is developed on top of a standard storage facility and can be used by a number of different applications.

Malleability

The general network is malleable in that it is freely editable given the appropriate set of access rights. However, these access right control the ability of participants to modify the network and need to be initially configured by the cooperating group. Neither the types of comment or the social roles identified can be ammended.

Visibility

The features of the system is made visible to user through a set of interfaces which allow users to directly alter access rights to different nodes on the quilt comment tree.

Level of abstraction

No level of abstraction is specified for the cooperation or the notation. A node in the tree can contain different levels of information from simple messages to complex diagrams and alternative suggestions of text.

Propagation of changes

Given that all users share the Quilt document tree changes are propagated quickly in that alterations are immediately available to other users.

The context maintained

The system assumes the use of roles as a means of determining different relationships between users and the shared document. These roles may be closely related to defined organisational roles. Quilt makes no semantic claim for these roles and merely uses them as a place holder for defining access.

The degree of openness

Quilt is designed to be open in that it assumes no particular text editor and offers a general framework for constructing shared documents. In addition , Quilt uses the ORION database for storage to increase the degree of openness offered by the system.

3.4.3. gIBIS Design Rationale System

3.4.3.1. Description

Argumentation systems support the structured development of multi-party arguments and negotiation. The gIBIS (Conklin 88) system supports the argumentation process undertaken by system designers. The system is based on a well known and simple model of design deliberation called Issue Based Information System, or IBIS (Rittel 73).

3.4.3.2. The basic MOI

The IBIS method is based on the principle that the design process is a conversation between a number of protagonists each holding a specific viewpoint. The IBIS model focuses on the Key *Issues* in the Design problem. Each Issue can have many *Positions*, where a Position is a statement or assertion which resolves the Issue. Each of an Issues Positions, in turn, may have one or more *Arguments* which either support that position or object to it. Thus each separate Issue is the root of a tree, with the children of the Issue being Positions and the children of the Positions being Arguments. The application of the IBIS model within gIBIS allows a small set of legal rhetorical moves to be constructed, the legal gIBIS moves are shown below in Figure 3.4-3

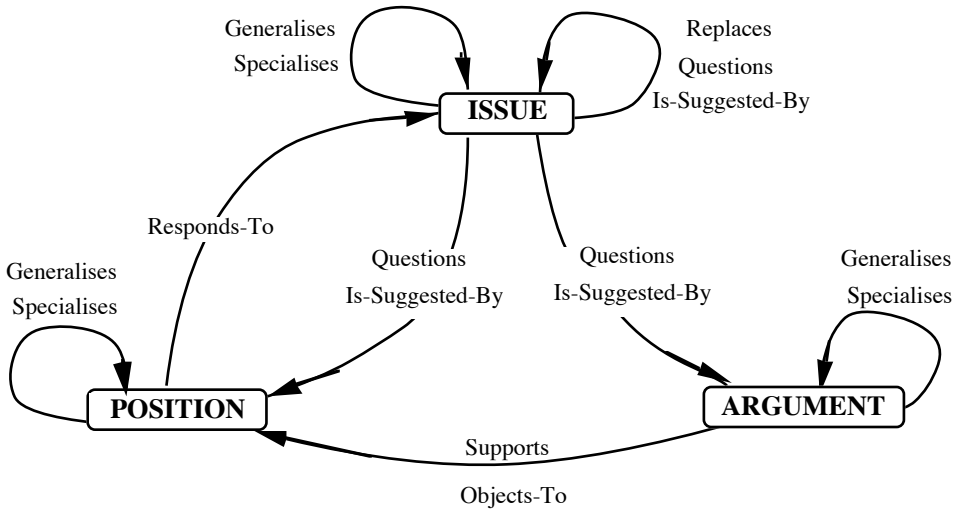


Figure 3.4-3 The set of legal rhetorical moves in gIBIS

3.4.3.3. The Notation

The gIBIS tool supports simultaneous access and update of issue groups by multiple users on a common local area network. It provides the necessary concurrency control, locking, and update notification to allow real time interactive network construction by teams of cooperating designers.

The explicit notation used by gIBIS is the typing of both nodes and arcs within the Hypertext system to support the development of an argument. The hypertext network is type checked to enforce the rhetorical moves allowed by the IBIS method. The notation is displayed as different coloured and shaped nodes linked by different labelled arcs. General gIBIS network follow a pattern of Issue , Position and Argument (IPA). The network tends to grow as a tree and the graphical tools provided with gIBIS support the display and growth of this network

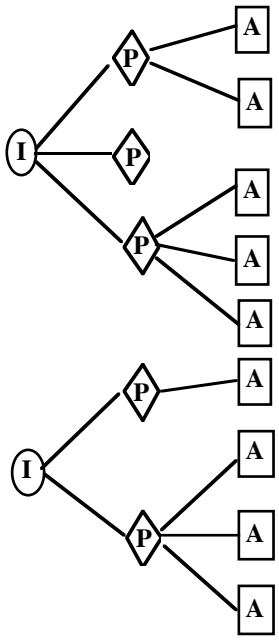


Figure 3.4-4 . A gIBIS network

gIBIS represents a particular solution which supports a specific approach to collaboration. As such the tool is limited in that it supports only a single perception of collaboration and argumentation within the design process. However, the developers of the method and the tool claim it has proven to be successful (Yakemovic 90) with its underlying model easily applicable to the design process. Other approaches to design rationale have been proposed which alter the different type provided. These include SYBIL(Lee 90) which is electronic supported and QoC(MacLean 89) which is predominantly theoretical. The concept of typing of the hypertext network is also supported in general purpose tools such as Oval(Malone 92)

3.4.3.4. The features of mechanism

Generality

The developers of gIBIS claim that the IBIS method is generally applicable for the representation of any argument. However, in practise users have the system have found the structure not to be expressive enough to capture their particular design discussions.

Malleability

The system presents the small set of vocabulary within gIBIS directly to the user and allows the construction and subsequent alteration of arbitrarily complex argumentation networks. However, the vocabulary is fixed in that only the given types of node are allowed within a gIBIS network.

Visibility

The tools supporting gIBIS are designed to allow direct visualisation and manipulation of the argumentation network by different users. Thus gIBIS makes the use of notation explicitly visible to users.

Level of abstraction

No suggestion is given as to the level of abstraction supported by gIBIS. However, the underlying philosophy is hierarchical in that large issues are decomposed into a set of positions which are then supported or refuted by arguments which in turn raise more detailed issues. This decomposition continues until small scale design decisions are recorded

Propagation of changes

The emphasis on sharing ensure that propagation of change takes place. A more recent real time version of gIBIS called rIBIS(Rein 91) ensures that the propagation of change takes place as quickly as possible.

The context maintained

The gIBIS notation does not explicitly represent users or their relation to the network. The focus is on the development of an argumentation network which represents the rationale surrounding a particular artefact.

The degree of openness

Openness is severely limited in gIBIS in that all design rationale must be expressed in terms of the given vocabulary of discourse.

3.4.4. The Answer Garden Organisational Memory

3.4.4.1. Description

The Answer Garden system developed by Thomas Ackerman(Ackerman 90) at the centre for co-ordination science at MIT is a general hypertext system intended to support the development of an organisational memory. Ackerman characterises the problem he is addressing as an inability for people within an organisation to have questions and queries correctly answered by appropriate experts within an organisation.

> " The Answer Garden system helps an organisation solve these problems by providing a database of answers to commonly asked questions that grows *organically* as new questions arise and are answered. " (p 31, emphasis as original)

The collecting together of this information is termed an "*organisational memory*" and Ackerman makes the point that while there is little theoretical consensus as to what constitutes an organisation's memory the metaphor is attractive.

3.4.4.2. The basic MOI

The Answer Garden captures answers to questions that would otherwise be lost and provides tool support for the subsequent retrieval of answers. Three key concepts are identified as central to the Answer Garden.

1) A branching network of diagnostic questions which help users find the answers they want.
2) New questions from users are automatically routed to appropriate experts and then inserted along with answers in the network.
3) Experts can modify the diagnostic branching network in response to users' problems.

The primary way in which users locate answers in the Answer Garden is by following a branching series of multiple choice questions. These are presented as multiple choice questions which the users answers until they find the appropriate question and answer to satisfy their needs.

3.4.4.3. The notation

The core of the Answer Garden is a notationally simple linked network which users traverse. The vast majority of this network is structured as a series of diagnostic questions of a form similar to expert help systems. These questions might include those of the form" does the disk light come on?". More expert users can circumvent the need to traverse this hierarchy by directly acessing the network through a graphical display of the tree,

The key to the Answer Garden is the way in which the memory grows. Users can attach comments to any part of the branching network to allow it to be extended. When annotations are attached to a branching questions an email message is directed to the expert responsible for that part of the network. The expert can then collate these questions for later addition to the network. Tools support is provided to inform experts of new questions and statisticsof use. These are then used by experts to focus answering the actual questions of users rather than esoteric issues.

3.4.4.4. The features of the mechanism

Generality

The Answer Garden make use of a simple structuring of answers with linked questions which in turn can raise questions. This structure is generally applicable and has been used to support the location of expertise in a number of different domains.

Malleability

The Answer Garden can be altered by all users and experts alike and is thus highly malleable. Limited support for controlling or co-ordinating the update is

provided and the assumption is that social conventions will suffice for managing the growth of the network.

Visibility

Answer Garden relies on a set of interfaces which make the network visible to users, either as a set of questions to guide the user through the network or as a graphical display of the network to allow direct access. Both are essential to the use of the system.

Level of abstraction

A variety of levels of abstraction are supported in terms of the granularity of the questions and the linking between questions and answers. Questions from a very general level can lead to very specialised points as answers and vice-versa.

Propagation of changes

Changes to the network take place over extended time periods and while experts are informed of added questions other users are not informed of growth of the network.

The context maintained

Little context is maintained about the users of the network other than to distinguish them as experts with particular expertise associated with different portions of the network

The degree of openness

The Answer Garden network is open in that anyone can add answers and questions to the network. In addition, the information is stored as flat UNIX files which allows it to be run on any UNIX system which supports X-Windows.

3.4.5. The Maintenance Support Tool

3.4.5.1. Description

The use of information sharing as a means of supporting cooperation has also been used to develop a tool to support software maintenance at Lancaster. Initially, software maintenance appears to be primarily an individual task with little group work in evidence. However, as software systems have grown in size and complexity the task of maintaining them has increasingly become the shared responsibility of maintenance teams. Maintenance tasks are undertaken by the maintenance engineers individually or in loose cooperation with others. However, when this process is viewed over the life-time of the system, maintenance is a group endeavour requiring the combined effort of the maintenance team. Importantly, due to the extended life-span the collaboration required to do this is inherently long-term. This perspective of cooperative work is characteristic of

many organisational domains and it is important that CSCW systems are sensitive to the issues involved.

As a software system is maintained by a maintenance team a set of experiences and expertise emerge which is of potential value to future maintenance activities. The development of this understanding is a time consuming task, much of it involving the incremental gathering and processing of information gained from examining different system components. However, this maintenance rationale is seldom recorded in any systematic manner and is lost when a programmer ceases to maintain that particular system. When systems are maintained by a team, an agreed understanding of this system rationale becomes crucial if the co-ordination for effective maintenance is to occur.

Documentation plays a vital role in promoting the understanding necessary for effective maintenance of any system. However, in many cases systems reach the maintenance phase without adequate or appropriate documentation. Even when documentation from the design phase is available, it quickly becomes outdated, and is usually abandoned in favour of the source code. This is generally due to the unsuitability of much of the documentation available for maintenance. Maintenance rationale needs to be far richer than traditional documentation and reflect the views and experiences of different maintenance engineers. This reflects the additional kinds of information that maintainers may want to capture, such as informal notes and details which may be of a tentative nature. In this respect, the emerging body of maintenance rationale forms a superset of traditionally available documentation, and can be considered to be the documentation for the system as in the traditional sense.

The approach adopted is support for collaboration through *documentation by annotation*. By using hypertext style links maintenance rationale can be associated with the appropriate part of the system being maintained and tools can allow this information to be easily browsed and amended. In a similar way to document review tools and co-authoring systems, the source code is treated as a *base document* to which the documentation is linked in the form of *maintenance comments*.

This method of documentation by annotation allows system documentation to be easily extended on a day to day basis by any member of the maintenance team. In this way, the communication is centred about the system, growing and evolving as information is discovered. Furthermore, documentation will only be added when and where it is needed, targeting the information to parts of the system requiring most explanation. This close integration of the rationale with the artefact enables the most crucial aspects of the discussion to be situated where it is most influential.

In addition to being able to create and link comments to the source code, comments can be added to other comments and additional links can be added to annotations which already exist. This informality, coupled with a flexible comment typing mechanism, provides an unconstrained approach to documentation. For example, tentative or incomplete notes can be added quickly during program comprehension which, in the light of new understanding can be amended.

Maintainers can similarly add comments on possible improvements or additions as they are noticed.

3.4.5.2. The basic MOI

The system takes the form of an *interactive program editor* through which the maintainer can browse and edit the evolving information space. Comments are attached to the source code through the user interface (see Figure 3.4-5). Links to comments are displayed by highlighting the text to which the rationale is linked; for example Figure 3.4-5 shows a section of source code where a maintainer has attached a simple textual comment to a variable.

The maintainer can freely edit the source code and references to annotations are kept consistent.

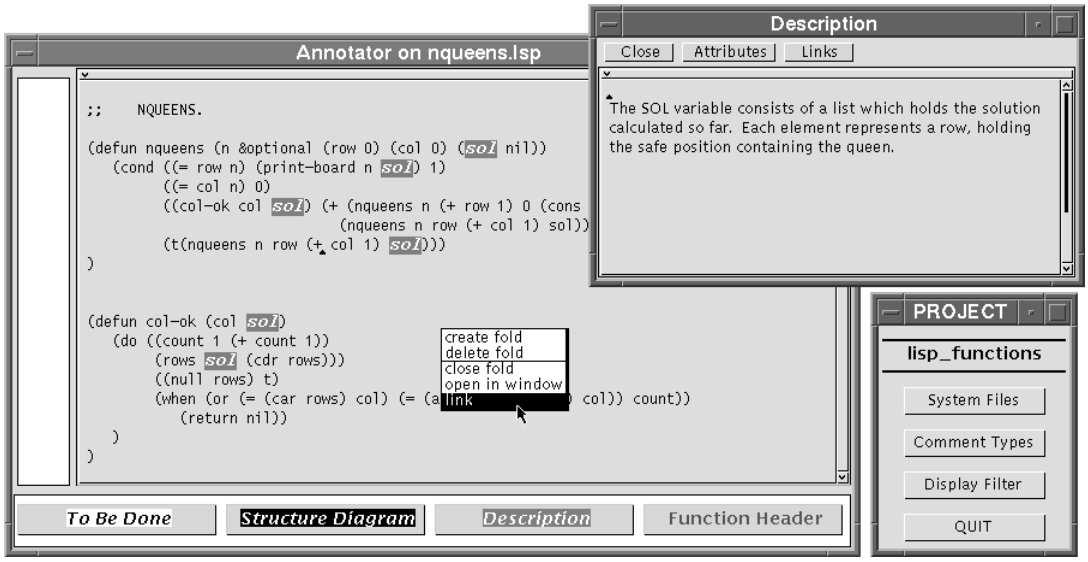


Figure 3.4-5 The Maintenance System

A number of important views can be seen in figure 3.4-5, the *main view*, the *key view* and the *margin*:

The main view: This is the largest portion of the interface and displays the program source code. In addition to allowing comments to be added and examined it acts as a full screen editor.

The key view: This panel acts as a key for all the currently defined comment types. This is achieved by highlighting the comment type buttons in their associated display style.

The margin: The margin lies to the left of the main view and displays iconic information concerning the comments attached to the system source code.

3.4.5.3. The Notation

The aim of the maintenance system is to allow the knowledge space to evolve over time into a richly structured network of inter-related discussions and information centred about the software. Rather than impose a particular structure for representing this information a set of mechanisms is provided which allows maintainers to define entities which reflects the diversity of knowledge present. This freedom of expression allows members of the maintenance team to create comment objects that represent the different kinds of information. For example, a comment could be defined to be of type ‘Bug Report’. This would be used to describe the nature of errors found during routine maintenance work.

In addition to being able to represent the “type” of the information, system is also able to allows particular comment type to be based on one of a three base representations: free text, graphics and structured textual forms. Complex comment types can be defined by maintenance engineers using a graphical definition facility. This allows the different elements of the structured comment to be added and positioned with text labels in the complex comment. The graphical comment and structured comment definition facility are shown below in figures 3.4-6 and 3.4-7:-

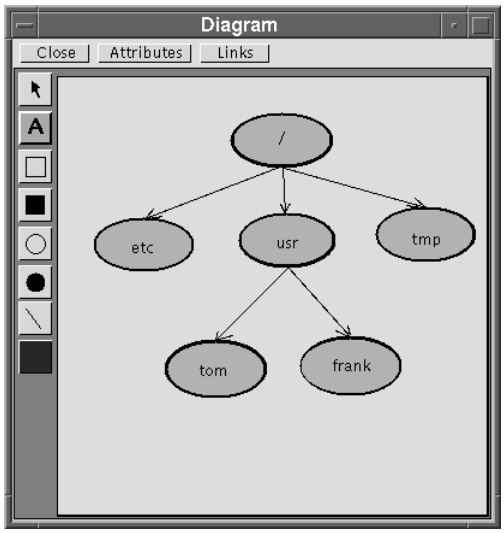


Figure 3.4-6 Basic Graphical Comment

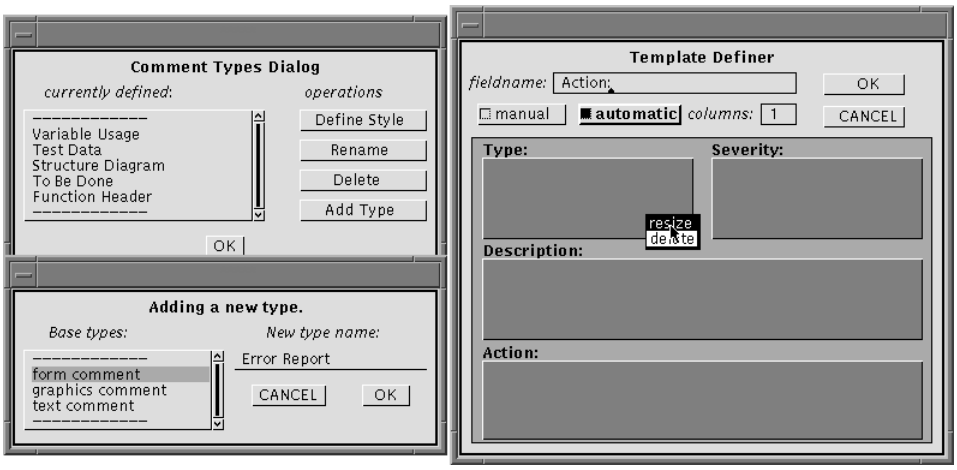


Figure 3.4-7 Defining Structured Comments

The sense of rationale used in maintenance is less contentious than in the case of design, the emphasis is on describing and explaining the nature of the artefact rather than an argumentative process of constructing the artefact. As a result the system focus is on the ability to support the “freedom of expression” necessary to allow effective descriptions to be recorded. Integrating the discussion with the software system supports indirect communication between different maintenance engineers working on the system over time. To support co-ordination between these maintainers it is necessary to also record information about the context in which the deliberation occurs. Each comment instance captures this in a number of attributes: *who created it, when it was created,* when it was *last modified* and *who modified it*. These attributes provide important contextual information to users for deciding how to interpret the comment, such as how recent it is and the skills of the maintainer involved. These attributes also facilitate browsing, filtering and report generation of the structured space. Thus a maintainer can see all the comments added before or after a given date or by a selection of team members. A set of filters can be defined within the system to allow a maintenance engineer to select a subset of comments of particular interest to her.

3.4.5.4. The features of the mechanism

Generality

The system is language independent and imbeds the annotations as comments within the source language using the appropriate formatting instructions for the language or tool

Malleability and Visibility

A set of tools are provided to allow users to associate particular colours and fonts with the different type of comments they wish. This set of annotation types can be freely extended by the maintenance team.

Level of abstraction

No particular level of abstraction is assumed and the system has been used to comment on source code in any programming language. Because of its focus on the detail of source code it suffers in its consideration or more widespread issues associated with large scale systems

Propagation of changes

The changes are made directly to the source code. However, the current editor does not support simultaneous real-time sharing across a group of maintainers and the assumption is that cooperation takes place over an extended time period.

The context maintained

Little context is maintained other than the names of maintainers and the time and date they added annotations. Additional contextual information can be added in a number of free slots provided to describe different users of the system.

The degree of openness

The annotations are encoded using a human-readable mark-up language which shows the annotations as comments within the source code. This allows the source code to be accessed through a wide range of editors.

3.4.6. Summary and Conclusions

This paper has examined the use of shared structured information as a means of supporting cooperation. The general structuring technique explored was that of nodes joined by links to form hypertext structures. These structures allow a range of different information to be shared across a community of users and the management of this sharing to be used to mediate the cooperation which takes place. An examination of this kind allows the notational exploration undertaken within the COMIC project to consider properties of systems based on shared objects networks. This has direct consequences for the consideration of shared objects within the shared object server been developed in strand 4.

Two principle notational techniques are used to support cooperation within systems based around the concept of shared hypertext. The addition of type information to the hypertext network and the definition of access permission to distinguish different members of the cooperating group. Access permissions are most often used to distinguish different social roles while type checking is used to enforce the way in which information is structured and represented. Both these aspects combine to define the particular mechanistic behaviours of these different cooperative systems.

3.4.7. References

Ackerman M.S., Malone T.W. (1990) , ‘ Answer Garden: A tool for Growing Organisational Memory’, in Lochovsky F.H.(ed) COIS90 Proceeding conference on Office Information Systems, April 25-27, 1990 , Cambridge, Mass.

Catlin, T., P. Bush, and N. Yankelovich. (1989) ‘InterNote: Extending a Hypermedia Framework to Support Annotative Collaboration.’ ACM Hypertext’89 Proceedings. Applications in Writing. Pages: 365-378.

Conklin, J. (1987) ‘Hypertext: An introduction and survey.’ IEEE Computer Vol: 20 No.: 9, Pages: 17-41.

Conklin, J., and M.L. Begeman. (1988) ‘gIBIS: A Hypertext Tool for Exploratory Policy Discussion.’ Proceedings of ACM CSCW’88 Conference on Computer-Supported Cooperative Work. Structured Communication Technologies. Pages: 140-152.

Fish R.S., Kraut R.E., Leland M.D., Cohen M. (1988) ‘ Quilt: A colloborative tool for cooperative writing’, in Allen R.B.(ed) COIS88 Proceeding conference on Office Information Systems, March23-25,, Palo Alto, California.

Halasz, F.G. (1987) ‘Reflections on NoteCards: Seven Issues for the Next Generation of Hypermedia Systems.’ ACM Hypertext’87 Proceedings. Issues. Pages: 345-365.

Jeffay, K., J.K. Lin, J. Menges, F.D. Smith, and J.B. Smith. (1992) ‘Architecture of the Artifact-Based Collaboration System Matrix.’ Proceedings of ACM CSCW’92 Conference on Computer-Supported Cooperative Work. CSCW Architectures. Pages: 195-202.

Lee, J. (1990) ‘SIBYL: A Tool for Managing Group Decision Rationale.’ Proceedings of ACM CSCW’90 Conference on Computer-Supported Cooperative Work. Supporting Structured Communication. Pages: 79-92.

Leland, M.D.P., R.S. Fish, and R.E. Kraut. (1988) ‘Collaborative document production using Quilt.’ Proceedings of the Conference on Computer-Supported Cooperative Work (CSCW ‘88), Portland, Oregon, September 26-28. ACM Press, Pages: 206-215.

MacLean, A., R.M. Young, and T.P. Moran. (1989) ‘Design Rationale: The Argument Behind the Artifact.’ Proceedings of ACM CHI’89 Conference on Human Factors in Computing Systems. Issues in Interface Design Methods. Pages: 247-252.

Malone, T.W., K.-Y. Lai, and C. Fry. (1992) ‘Experiments with Oval: A Radically Tailorable Tool for Cooperative Work.’ Proceedings of ACM CSCW’92 Conference on Computer-Supported Cooperative Work. Emerging Technologies for Cooperative Work. Pages: 289-297.

Perlman, G., D.E. Egan, K. Ehrlich, G. Marchionini, J. Nielsen, and B. Shneiderman. (1990) ‘Evaluating Hypermedia Systems.’ Proceedings of ACM CHI’90 Conference on Human Factors in Computing Systems. Panel. Pages: 387-390.

Rein, G.L., and C.A. Ellis. (1991) ‘rIBIS: A real-time group hypertext system.’ International Journal of Man Machine Studies Vol: 34 No.: 3, March, Pages: 349-368.

Rittel H., Webber M. (1973) ‘ Dilemas in a General Theory of Planning’, Policy Sciences, Vol 4.

Yakemovic, K.C.B., and E.J. Conklin. (1990) ‘Report on a Development Project Use of an Issue-Based Information System.’ Proceedings of ACM CSCW’90 Conference on Computer-Supported Cooperative Work. Supporting Structured Communication. Pages: 105-118.

Yoder, E., R. Akscyn, and D. McCracken. (1989) ‘Collaboration in KMS, A Shared Hypermedia System.’ Proceedings of ACM CHI’89 Conference on Human Factors in Computing Systems. Tools for Collaborative Work. Pages: 37-42.

3.5. COSMOS, AMIGO Advanced and MacAll II

John Bowers

Manchester University

3.5.1. Introduction

In this paper, three early European CSCW projects are described and analysed. These projects represent some of the earliest research endeavour in CSCW in Europe — being developed in the period 1985-1989. The projects to be dealt with are:

- COSMOS
- AMIGO Advanced
- MacAll II

Aside from being early attempts, these projects have a family of features in common. COSMOS and AMIGO Advanced, for example, are strongly based around notions of activity and roles played within activities. MacAll II is an implementation of some of the notations developed within AMIGO Advanced together with some preliminary recognition of the importance of supporting organisational context.

These projects did not all yield working CSCW systems. Working prototypes within the COSMOS and MacAll II projects were built and a full specification for an (unbuilt) revised version of COSMOS is available. In contrast, the work of the AMIGO project remained theoretical although they have formed the basis of subsequent working systems (particularly ongoing work at GMD).

Each of the projects will be discussed in the following manner. First, a brief overall description of the project and its history will be given. Following this, the basic mechanisms of interaction (MOI) recognised in the projects will be isolated and described. Next, any notations developed within the projects will be described. Finally, the MOIs will be analysed further in terms of the features used by Simone, Grasso and Pozzoli (1993).

It should be noted that these projects have been reviewed together before. Hennessy, Benford and Bowers (1989) review them in a different, though complementary, style to that developed below. It is worth comparing the current treatment with that earlier paper.

3.5.2. COSMOS

3.5.2.1. A Short Description

COSMOS was a multidisciplinary research project which attempted to integrate technical and social scientific perspectives in the development of an advanced communication system. The project ran from 1986-1989, was funded under the auspices of the U.K.'s Alvey initiative in information technology and involved the collaboration of a number of U.K. companies and universities, including two of the participating institutions in the COMIC project (Manchester and Nottingham Universities).

The central concern of the COSMOS project was to develop a COnfigurable Structured Message Oriented System, hence the acronym. An X.400 based prototype system was built in 1988/1989 which expressed some of the central ideas of the project and the project closed with the specification of a revised design, which remains to date unbuilt. Unfortunately, then, working demonstrations of some of the project's ideas are not available for scrutiny. However, the central notions that computer supported cooperative work should be regarded as *structured* and *configurable* is demonstrated in the prototype system and amplified in a series of documents and publications (e.g. Bowers and Churcher, 1988; COSMOS, 1989; Dollimore and Wilbur, 1991).

In theoretical work, the project regards cooperative work as organised around *practices* which involved the coordination of different actions by different participants in the production of some entity. Bowers and Churcher (1988) describe how their perspective on cooperative work can be thought of as an extension of language/action perspectives such as Winograd and Flores' (1986). Rather than draw inspiration from the formalised account of speech act theory of Searle (1969) as do Winograd and Flores, Bowers and Churcher describe how work in Conversation Analysis can be used to inspire the design of CSCW systems which support both conversational and non-conversational interaction. So, while — like Winograd and Flores — Bowers and Churcher acknowledge the importance of locally managed conversation to cooperative work, they feel that this ought to be complimented by a perspective which recognises the non-conversational, global management (or structuring) of practical activities.

In the COSMOS environment, working practices are represented as *Communication Structures* (CSs) using a formal language called SDL (for *Structure Definition Language*). A CS is an abstract representation of a working practice, independent of any particular occasion of use, and the COSMOS environment is regarded as containing libraries of CSs which can be instantiated in the support of cooperative work. An instantiated CS is referred to as an *activity* in COSMOS.

3.5.2.2. The Basic MOIs

COSMOS can be regarded as consisting of at least three mechanisms of interaction. However, only one of these is well specified in the COSMOS literature.

Support for globally structured activities

These are supported through CSs notated in SDL. In principle, SDL should be accessible by any COSMOS user. A CS contains the following components:

- The **name** of the CS. Every CS must have a name. A CS's name can be referred to in another CS so that, for example, related CSs can exchange messages.
- **Actions** — These are divided into *exchanges*, which are elementary communicative acts involving at least two roles (a notional sender and one or more recipients) and at least one object, and *encapsulated actions* (EAs) which are specific to just one role (e.g. creating or modifying objects).
- **Rules** — These are production-rule like formalisms which assemble one or more related actions together on their RHS and define the conditions under which the actions can occur on their LHS.
- **Roles** — These are the agents who are capable of action. A role may be instantiated at run time by an individual user, or a group of users or by an automated process. The mapping between physical users and processes and abstract roles is entirely flexible as are the relations between roles within the CS definition, being open to dynamic change and so forth.
- **Objects** — These are informational entities which can be created, modified, embedded within each other etc. in the execution of a CS.
- **Conditions** — These are part of the LHS of rules which are resolvable to true or false and determine when the RHS of rules are potentially executable.
- **Temporal order constraints —** These state constraints on the order in which actions can occur. These can be of a logical nature (an object can only be exchanged once it has been created) or can relate to the sequential organisation of interaction (a request for an object will precede its response). Temporal order constraints provide further mechanisms for specifying the order of events over and above the LHS/RHS structure of SDL's production rules.

It should be clear from this account that SDL is primarily a procedural language for representing patterns of communication in working practices.

Support for locally structured conversations

Bowers and Churcher (1988) argue that CSs notated in SDL capture the overall organisation (or global structure) of a working practice. A CS will not necessarily specify or support the actual execution of EAs or determine, say, how many messages between two actors are required for them to exchange a document. The realisation of a CS at the level of actual messages/turns in interaction need not be di-

rectly notated in SDL. In this way, Bowers and Churcher wanted to provide support facilities for cooperative work without overly constraining users in the exact execution of their activities. However, this leaves open the question of just how to support the locally managed conversations through which a CS will be realised. This is noted as an issue in the COSMOS literature and the final COSMOS report suggests that — amongst other possibilities — Coordinator-like support might be offered at this level (cf. Winograd and Flores, 1986). While ingenious solutions are offered in this document to the problem of how to inter-relate locally managed conversations with the global structuring that a CS provides, little substantive is said about exactly how conversation-like interaction could be supported in COSMOS.

Support for departures from globally structured activities

It is recognised that users may have to depart from the structure provided by a CS to handle the exigencies of situations as they unfold at run-time. An MOI called SPACE (for Structure PeripherAl Communication Environment) is provided for this purpose. COSMOS (1989) gives details of the kinds of facilities SPACE would provide. For example, in SPACE, an action or an action sequence could be declared as done if it was no longer relevant. Similarly, a condition could be forced to be true or false. Some of the facilities of SPACE are comparable with routine exception handling and contradiction resolution mechanisms for production systems. More interestingly, COSMOS (1989) discusses the possibility that users may be supported during run-time in the editing of fragments of SDL. In this way, re-defining parts of the CS while an instantiation of the CS runs is supported. This is a step towards understanding CSs as self-organising and user-modifiable even at run-time. In an extension of this idea, Bowers (1990) describes a demonstrator system called TRACE in which editing CSs in SDL and acting within an instantiated version of the emerging CS are exactly cotemporaneous processes — that is, CSs are designed and redesigned ‘from within’. However, fully functional implemetations of SPACE and TRACE are yet to appear. In the COSMOS prototype, CSs were not modifiable at run time and exceptions to the actions represented in a CS could only be conducted through using a background, standard email facility.

3.5.2.3. The Notation

SDL has been outlined above. Below is a CS in SDL based on the practices by means of which reports are written in a U.K. local government organisation (from Bowers and Churcher, 1988):

REPORT_WRITING_CS

REPORT_WRITING_CS **->**
REPORT_WRITING_JUNIOR**,**
REPORT_WRITING_ SENIOR

REPORT_WRITING_JUNIOR **[**Author **evaluate** "The Author is not a Senior Management Team member"**] ->**
Author **create** *report,*
Author **allocate** Line_manager**,**
OBTAIN_COMMENTS**,**
CHANGE_REPORT**,**
LINE_AUTHORISATION**,**
REVISE_REPORT**,**
SUMMARY_FOR_DIRECTOR**,**
BACKGROUND_DOCUMENTATION

REPORT_WRITING_ SENIOR **[** Author **evaluate** "The Author is a Senior Management Team member"**] ->**
Author **create** *report,*
OBTAIN_COMMENTS**,**
CHANGE_REPORT**,**
SUMMARY_FOR_DIRECTOR**,**
BACKGROUND_DOCUMENTATION

OBTAIN_COMMENTS **->**
can (Author **allocate** Interested_others **& invitation [for** *comments***]** *report* Author Interested_others**),**
can (Interested_others **create** *comments* **from** *report***;** *comments* Interested_others Author)

CHANGE_REPORT **->**
Author **create** *report* **from** *report comments***..,**
LINE_AUTHORISATION **[**Author **evaluate** "Line authorisation is now required"**] ->**
report Author Line_manager**,**
(Line_manager **create** *rejection* **&** *rejection* Line_manager Author)
or (Line_manager **create** *authorisation* **&** *authorisation* Line_manager Author**)**

REVISE_REPORT **[on** *rejection* Line_manager Author**] ->**
CHANGE_REPORT**,**
LINE_AUTHORISATION

SUMMARY_FOR_DIRECTOR **[**Author **evaluate** "Report requires a summary for the Director"**] ->**
Author **create** *report_summary* **from** *report***,**
report_summary Author Director

BACKGROUND_DOCUMENTATION **->**
Author **create** *backgound_documents_list* **from** *report***,**
backgound_documents_list Author Proper_Officer

REWRITING_REPORT **[on** *rejection* Senior_Management_Team **/CS:**
SMT_AUTHORISATION_CS
/ Author **or** *rejection* Senior_Management_Team Line_manager**] ->**
REPORT_WRITING

Author **create** *report***,**
Author **allocate** Interested_others
< **request [for** *comments***]** *report* Author Interested_others
< Interested_others **create** *comments* **from** *report*
<*comments* Interested_others Author
< Author **create** *report* **from** *report***;** *comments***..**
<*report* Author Line_manager
< Line_manager **create** *rejection***,**
Line_manager **create** *authorisation*
< *rejection* Line_manager Author**,**
authorisation Line_manager Author**;**

Author **allocate** Line_manager
< *report* Author Line_manager**;**

Author **create** *report_summary* **from** *report*
< *report_summary* Author Director**;**

Author **create** *backgound_documents_list* **from** report
< *backgound_documents_list* Author Proper_officer

3.5.2.4. The Features of the Mechanisms

This section will concentrate on the relevant features of the MOI which provides support for globally structured activities as it is this which was actually built into the prototype COSMOS system and most fully specified in the project's final report. The features as outlined by Simone et al. (1993) are used below.

Generality

In principle, SDL was intended as an entirely general notation for representing working practices. However, the COSMOS literature contains a fairly circumscribed set of examples: collaborative paper writing and only a few others. Thus, the plausibility of this claim has not been effectively tested.

Additionally, as SDL is concerned to represent working practices in their overall, global organisation, local structuring and the exact mechanisms by which negotiation of changes takes place are not supported through explicit notations.

Malleability

In principle, SDL was intended to be available to any user of the COSMOS system. That is, any user could author CSs in SDL, register them in the COSMOS environment, and instantiate them for use. As such, the overall functionality of the COSMOS environment is in principle fully end-user configurable. This was a central aim of the project, though exactly how in practice this would work out is a little unclear.

Visibility

Once a CS had been instantiated, a variety of kinds of information about the activity would be visible to a user playing a role in that activity. For example, the following would be available in different user interface 'contexts' (see COSMOS, 1989, chapter 4 for details):

- lists of users playing roles
- the SDL definition of the CS
- CS library contents
- the temporal relationships between rules
- the rules and their current states (a rule-based view)
- the definition of roles (a role-based view)
- the definition of actions (an action-based view)
- a list of the currently available actions which a user could next perform

etc.

In short, a variety of views were available to users each organised around a key SDL concept. Additionally, some of these would dynamically change as the activity progressed. However, in terms of the visibility of the information objects in an activity, it was generally assumed in COSMOS that information became visible through explicit exchange, i.e. one role sending information to one or more others. Indeed, this leads Hennessy et al. (1989) to contrast the procedural approach of COSMOS with the information sharing approach of some other projects.

Level of abstraction

A CS is an abstract representation of a working practice, intended to capture the 'normal', 'non-accountable' conduct of that practice. In a sense, a CS is a motivated ideal of the practice it stands for. This means that it is not intended to capture every possible way in which the practice could actually be realised (nor could any representation). Also a CS abstracts away from the detailed turn-by-turn texture of the local organisation of work (see above). While communicative actions exist at the level of granularity of speech acts, any particular exchange in a locally

managed sequence can be a composite of several speech acts or, indeed, any particular act might be distributed over a number of turns (messages) in a sequence. Exactly how speech acts are realised in locally managed sequences is not modelled in SDL.

Propagation of changes

How changes to CS definitions were to be propagated and how the management of different CS versions in the COSMOS CS library was to be organised were issues that were not definitively settled in the COSMOS project. Similarly, changes made to CS definitions during activity run-time, how these were to be propagated and how notice of activity in SPACE was to be distributed were all undecided issues.

The context maintained

Rule, role and activity contexts are supported by the views discussed above. That is, any action can be seen in relation to the rule it is part of, the roles involved in it, the activity it is part of etc. Also, any specific dependencies relating this action to its antecedents or to those following could be highlighted.

The degree of openness

The COSMOS system was built upon X.400 standards and hence was intended to be open to other X.400 compatible messaging systems and services. Additionally, as roles could refer to automated processes in SDL, a CS could call a process or some external function/routine in another environment and send 'objects' to it and receive 'objects' from it. In this way, in principle, the notion of exchange includes the passing of a value to a function where the exchanged 'object' is taken to be the value.

COSMOS interfaces were prototyped for green screen, NeWS and Macintosh environments. These interfaces supported the integration of COSMOS with basic services in those environments: e.g. the Macintosh interface supported cutting and pasting to/from the Clipboard.

3.5.3. AMIGO Advanced

3.5.3.1. A Short Description

The AMIGO Advanced project, funded by the COST 11-ter program of the European Communities, was aimed at investigating the requirements for group communication tools and developed a model of group communication, the AMIGO Activity Model (AAM), based on *activities* representing group communication processes. The project published a monograph (Pankoke-Babatz, 1989) which traces the origins of the perspective developed in the project, gives details (including a full syntax) of AAM and demonstrates some examples of AAM in use to model some specimen activities, specifies the architectural requirements of

potential AMIGO-based systems, discusses alternative approaches and future developments. From all this detail, it is the AAM which is of greatest relevance in the current context.

In what follows, it is important to bear in mind that the AMIGO Advanced perspective on group communication was especially influenced by the literature on office procedures and their automation (see Pankoke-Babatz, 1989, Part I). Accordingly, the AMIGO Advanced work does not have the foundation in a general consideration of theories of communication, unlike the work of, say, Winograd and Flores (1986) and those that follow them (see Simone et al., 1993). This is not to say that the AAM is unduly limited, rather it is to remark on the historical tradition in which the AMIGO Advanced work was situated.

3.5.3.2. The Basic MOI

Activities expressed in the notation of the AAM provide the basic MOI for the AMIGO Advanced project. An activity is defined as the *intention or goal of a group* (e.g. date planning or newspaper editing to give two of the examples discussed in Pankoke-Babatz, 1989, Part IV). The basic components of an AMIGO activity are:

- **Roles** — A set of role classes is defined for each activity and each group member must be associated with one or more instances of one of these role classes. Defining a class involves specifying the functions that class instances may perform or request to be performed, and the message object types that can be sent or received by class instances.
- **Functions** — Functions are the operations which are performed by the group on objects (e.g. send, receive, forward). Functions (like the EAs in SDL, see above) are associated with just one processor (a role instance or the system) who is responsible for its execution. Specifying a function involves naming it, defining its input/output parameters, and the procedure which is to be executed.
- **Message objects** — Each activity involves the definition of its message object types. Defining a message object type involves specifying the attributes instances of that type may have, together with a range of values if appropriate. Message objects may be associated with *meta-information* continuing a description of the semantics of (parts of) the message, its history and overall context.
- **Rules** — Rules define how roles are to behave in the activity. As in SDL, the rules are of a production rule format, linking contexts with actions. Certain kinds of rules are also available which activate and deactivate other rules, thereby changing the contents of the current rule-set. In this way, activities containing more than one ‘phase’ or ‘sub-task’ can be modelled. Meta-rules also exist which may exert limits on the number of tasks or rules being executed at one time or may automatically cancel rules which have timed out (etc.).

Describing roles, rules, functions and message objects is to define an *activity template*. Executing an activity involves creating an instance from an activity template, together with instances of roles and message objects. The rule-bound performance of the activity is coordinated by a *central coordinating entity* which permits the execution of functions and so forth in an appropriate order. This entity could be an automated component of a system implementing the AAM to give computer support for group communication.

The AAM has many features in common with the SDL developed in COSMOS. Both are activity oriented models for cooperative work with a procedural flavour conferred on them by their use of rules with a production rule format. However, the AAM — as its terminology implies — has also been influenced by object oriented programming and systems development concepts. This means that AAM can model inter-activity relations in, perhaps, a more systematic way through notions of inheritance (etc.) than is possible in SDL (see Pankoke-Babatz, 1989, Part II, p.115-122).

3.5.3.3. The Notation

The AAM has been outlined above. Below is an extract from Pankoke-Babatz (1989, p.211-216) showing the role component for an AAM rendition of a bulletin board:

EDITOR: set of (<instance address>)

AUTHOR: set of (<instance address>)

CONTRIBUTOR: set of (<instance address>)

READER: set of (<instance address>)

and next part of the function component showing the definition of the **store** and **send** functions (**rubout**, **obsolete** and **fetch** are also specified):

Store

IN <notice>, <storage>
PROCEDURE:
CALL store(notice, storage)

Send

IN <message, <recipient>
PROCEDURE:
message.recipient=recipient
CALL send(message)

In the AAM version of a bulletin board in the AMIGO Advanced monograph, the message object component presents two types of message (notice and review) with their attributes listed. As this ‘form-oriented’ structure is not particularly informative of the syntax of the AAM, the interested reader is referred to pages 213-214 of Pankoke-Babatz (1989) for details.

The rule component for the bulletin board, clearly shows the production rule format of AAM’s rules in use. Here are some excerpts:

ON notice *FROM Contributor*
notice.*author*=notice.*sender*
notice.*editor*=" "
store notice *IN storage*
ON notice *FOR* ***rubout*** *FROM Author OR Editor*
rubout (notice *WITH message-id* = notice.*rubout-id AND* notice-*sender* = notice-*sender*) *OF storage*
ON review *FOR* ***fetch*** *FROM Reader*
fetch (*review, notice-set, storage, result*)

3.5.3.4. The Features of the Mechanism

Generality

AAM is proposed as a general model for activities of the sort classically modelled as formal office procedures. Although it does not boast the ambitiousness or universality of COSMOS' SDL, in practice, the two are comparable in their scope. Indeed, some notions seemed to be shared across the two projects (e.g. the use of production rule-like constructions). This should not be surprising as several COSMOS project members also participated in AMIGO. Perhaps, the function component of AAM's approach to activities allows a more systematic and extensible definition of the actions which can be performed by roles than does SDL's treatment of actions (where the syntax allows for a circumscribed set of possibilities together with the catch-all 'other action'). Again, AAM's use of object oriented notions may permit a more general and systematic treatment of inter-activity relations.

Malleability

As AAM, like SDL, is a fairly general purpose language/modelling notation, its users can generate and modify definitions of activity templates at will. In addition, inheritance mechanisms permit the generation of new templates on the basis of similar ancestors. While a user of AAM cannot change the fundamentals of the notation, the facilities available for defining functions allow a degree of extensibility.

Visibility

The AAM notation itself highlights the procedural aspects of an activity and hence brings roles, their function and so forth into focus. AAM does not attempt to model such matters as the 'knowledge' that agents might utilise in the execution of an activity.

In the AMIGO Advanced project, AAM was not implemented as a working system. Accordingly, which of the components of an AAM definition of an activity template may be visible to a role-player, which not, and under what circumstances are open issues. In the case of COSMOS and SDL, some insight into this could be gained from the specification of different user interface contexts in the

final project document, each of which specify what would be visible to users at what times etc. No comparably detailed work was done in AMIGO Advanced.

Level of abstraction

Like COSMOS' SDL, an activity template in AAM is an abstract representation of an activity which abstracts away from the identity of specific users and specific occasions of use. It is intended to capture the general features of the activity in question so that group communication could be supported. The level of abstraction intended in the AMIGO work is again revealed by the examples discussed: a date planning activity (e.g. for arranging a meeting), a bulletin board, coordinating newspaper reporting.

In the discussion of COSMOS above, it was noted that a distinction was made there between the actions represented in SDL and their actual realisation in moment-by-moment, turn-by-turn, message-by-message activity (i.e. a request could be satisfied in several messages). A CS, however, abstracts away from this message-by-message activity. The AAM, however, does not make this distinction. That is, these two 'levels of abstraction' in how actions/functions could be understood are conflated.

Propagation of changes

The AMIGO Advanced monograph says very little about how, say, run-time changes in the execution of an activity would be managed, let alone how they might be propagated through the activity. On page 97, it is noted: "The Rule Component defines what *should* happen. It is relatively easy to define this positive case in the specification. The Component must, however, also define what is to happen if things go wrong, e.g. if time limits are not kept, if unknown Message Objects are received, if specified regulations are disregarded, etc. A set of default regulations for checking the state of an Activity against these situations and possible deadlocks may be supported." [emphasis in the original]

However, little further detail of what these default regulations would look like, how they would be over-turned, how checking against them is coordinated with the normal execution of an activity and so forth are given. All of this of course points to some very hard problems to do with exception handling in production systems and also to important conceptual questions about the relation between formal notations like AAM and actual situated communicative practice. The AMIGO work notes the problem but does not offer a solution. The COSMOS work conjectures a solution (which involves a re-orientation of the relation between what is captured in the formalisms and actual moment-by-moment communication) but one which was not tested during the life-time of the project.

The context maintained

AAM's presentation of an activity template in terms of different components, in principle supports role-based, function-based, rule-based and message object-based contexts for inspecting the organisation/execution of an activity.

Additionally, meta-information attached to message objects allows an object to be defined in relation to a description of its meaning, history and overall context. In principle, the notation of AAM would support building a system which would maintain all these sources of contextual information.

The degree of openness

As AAM contains a general notation for procedure calls within the function component, systems embodying AAM could be connected up to external environments much in the same manner as described for COSMOS and SDL. However, as with many of these assessments of AAM, this remains an in-principle possibility as a system embodying AAM was not built during the life time of the AMIGO Advanced project. In principle, though, a general mechanism for external procedure calls exists within AAM. Again, as the AMIGO Advanced Group's modelling work remains at an abstract level, questions of the data formats and protocols supported (etc.) did not need to receive definitive answers. While COSMOS had a commitment to X.400 standards, the AAM is independent of such commitments, though Part V of the AMIGO Advanced monograph describes an architecture which could be realised using X.400/X.500 standards.

3.5.4. MACALL II

3.5.4.1. A Short Description

MacAll II was a project undertaken by members of the Communications Research Group of the Computer Science Department of Nottingham University. MacAll II — like its predecessor MacAll I — was funded by the Digital Equipment Corporation and was intended to demonstrate the possibility of adding CSCW-related functionality to their All-In-One office system. Smith, Hennessy and Lunt (1991) is the fullest available account of MacAll II.

Like the AMIGO Advanced work, MacAll II is specifically devoted to the support of office procedures and, indeed, it implements the AAM as its means for notating office activities. However, MacAll II adds to the AAM by introducing a number of other key concepts which had been experimented with in MacAll I and which arguably provide a more realistic context for activity modelling to operate within. These concepts include a notion of 'workspaces' (where inter-related information can be clustered together) and 'organisational manual' (which brings together specification of activities with organisational information regarding people's skills and responsibilities etc.). This composite environment is known as the AME (for Activity Model Environment).

In addition to adding workspaces and organisational information to the AAM, MacALL II is more thorough going in its object-orientation. All of the basic components of the AME are understood as object classes drawn from or based upon a standard class library of C++. The prototype of AME that Smith et al. (1991) report on was not developed in a truly distributed environment, nor was anything

but a crude command line interface developed as its user interface. However, this prototype does demonstrate many of the central concepts of the AME.

3.5.4.2. The Basic MOI

As with the work of the AMIGO Advanced project, activities offer the basic MOI in MacAll II. However, activities are given a richer elucidation in the AME. The basic components are:

- **People —** These are 'placeholders' that represent the actual individuals working within the organisation, and their capabilities.
- **Roles —** These represent what people can do in the execution of a specific office activity. One person can play one or more roles within the same or different activities.
- **Messages —** These are objects such as memos and forms which are used to transfer information between role instances. In AME, messages are typically considered to be *active* in that they contain rules which determine how they should be processed and what might happen to them after they have been processed.
- **Iunits** or **Information Units —** These are the constituents of messages. They may be fields which a role instance should supply information for or they may be attached documents or some combination etc.
- **Workspaces —** These are virtual work areas where information related to the execution of a role is organised. Workspaces are structured around in and out trays, an 'in progress' space and storage facilities.
- **Rules —** These define the actions that can be performed by a role and under what circumstances. Other rules add 'intelligence' to workspaces and messages. Rulesets are also identified. These are sets of rules which are always activated together (cf. notions of activity 'phase' and 'subtasks' in the AAM).
- **Functions —** These are operations carried out by roles and messages as part of an activity.
- **Activities —** These point to objects that are all involved in the same activity. Subactivities are also identified. These are groupings of related tasks within activities.
- **The Organisational Manual —** This contains definitions of all the above components together with information about the general organisational context in which activities can take place.

Activity instances are instantiated by assigning people to roles. Activity instances are executed by following the rules associated with the roles, information is transferred by sending messages from one workplace to another. A message's arrival at a workspace leads to the notification of the relevant role instance and the message is then processed in accordance with its rules.

As such, the AME provides automated support for office procedures. However, the rules associated with an activity are weighted with a 'division of responsibility' (DOR) which determines whether the rules should automatically execute or whether the task they express should be executed entirely under human control. In this way, the automated support of the AME is switchable. Smith et al. (1991) describe a simple implementation of this idea where the DOR is given an integer value and a threshold is set at the commencement of an activity. This is a crude rendition of an interesting idea worthy of further exploration.

3.5.4.3. The Notation

As noted already, MacAll II is in many respects an implementation of the Amigo Activity Model (AAM). Hence, many features of its underlying notation are exactly those of the AAM. In addition, MacAll II specifies the syntax for a number of other command, operations and functions associated with the management of workspaces, iunits and so forth. In Smith et al. (1991), just one example is given as an appendix to their paper — the syntax of one of the browsing commands 'display' — thus:

Syntax	display *type id*
Means	Display the Object of *type* with *id*. Shallow descriptions of objects contained within the displayed object will be used — e.g., the description of a Ruleset within an Iunit will not be displayed. The *id* is an integer (the creation id). However, for the *types* Workspace and Person, it is permissable to use the "name".
Example	display message 11; display workspace "UIA Secretary"

Although this is a small example, it does indicate that the additions to the AAM in MacAll II continue the syntactic and textual style of AAM.

3.5.4.4. The Features of the Mechanisms

Generality

The generality of the MacAll II MOI is much as that for the AAM. However, MacAll II includes components which allow the specification of the structure of objects in terms of Iunits and which allow the modelling of workspaces. Additionally, some recognition of organisation context can be found in MacAll II. However, just what this consists in is a little unclear in detail. The worked example in Smith et al. (1991) suggests that relations like 'is secretary to' and other organisation structural matters might be what the authors have in mind for the support of organisational context. These are notions not given explicit treatment in the AAM.

Malleability

Again as AAM. In addition, the modelling of workspaces, object structure and organisational structure are also supported. As before, although a user of the MacAll II MOI does not seem to be able to alter the fundamentals of its corre-

sponding notation, the function-definition facilities of the AAM allow some extensibility of the notation.

Visibility

MacAll II allows role-players to browse the state of the MOI and to inspect the current status of their workspace. An MOI for an activity can be browsed in terms of any of the model components (roles and the rest). This is supported by means of an ASCII command line interface in the prototype system reported in Smith et al. (1993, p.155). The actions which a user can perform given the current state of the activity are also made visible to the user. Similarly, this is implemented in the command line interface in terms of a series of commands to update the state of the underlying activity model. The AME itself will also make visible to the user any necessary changes in the state of the model, e.g. if an Iunit needs completing. Admittedly, the command line interface prototyped by Smith et al. is crude but it does indicate what commands are made available to users and what system events become visible ('browsing', 'updating' and 'AME generated'; for details, see Smith et al., 1991, p.155).

Level of abstraction

See the AAM above. MacAll II follows the same level of abstraction.

Propagation of changes

Like the AMIGO Advanced work, the MacAll II project did not really grapple with issues surrounding how run-time and other changes might be propagated through the system. Indeed, Smith et al. (1991, p.156) admit that many issues of this sort had not been dealt with in their prototype AME.

The context maintained

The contexts maintained in the AAM are also maintained in the AME of MacAll II. Additionally, some organisational, person and workspace-related context can be given in terms of the AME.

The degree of openness

The prototype AME was developed using a standard C++ class library. In principle, the AME would be open to the inclusion of other relevant object classes from this library or developments of them. At the end of their account, Smith et al. (1991) anticipate the further possible development of the AME by re-implementing it in a truly distributed environment using standard commercially available object oriented database technology. While this can be taken as indicating a desire that the AME should be developed on familiar platforms and be open to interchange with other applications, perhaps this is reading too much into a promissory note at the end of their paper.

3.5.5. Conclusions

In this paper, three early European contributions to CSCW have been reviewed. Each of these projects involves some commitment to activity modelling or to modelling communication structures (a very similar notion). This treatment has attempted to analyse the basic mechanisms of interaction involved in the approaches of these projects following the features discussed by Simone et al. (1993). However, it is important to note that the mechanisms of interaction to be found in the three projects discussed here raise some further issues. Some of these are noted by Hennessy et al. (1989). By way of conclusion, some of Hennessy et al.'s points are noted and elaborated on when they seem to be of especial interest to future work in COMIC.

3.5.5.1. Procedural versus Information Sharing

Hennessy et al. (1989) distinguish between approaches to activity modelling which are procedural in their orientation from those which have as their central emphasis information sharing. All the models we have examined (COSMOS, AMIGO's AAM and MacAll II's AME) have a procedural flavour to them. All of them use some production rule formalism to notate their respective MOIs. Activity coordination takes place through message passing and messaging is modelled on a traditional send-receiver basis. There is less of an emphasis on activity coordination taking place through information sharing. In this respect, Hennessy et al. (1989) contrast COSMOS, AAM and AME with the approach taken by the Amigo MHS+ project (see Benford, 1988) which attempts to model the different ways information can be shared. In COSMOS (Bowers, personal communication), some attempts were made to reformulate the notion of an exchange so as to understand exchanges *not* in terms of one party sending information to another *but* in terms of one party *granting access rights to another over information*. In this way, perhaps, a procedural, message-passing approach might be opened out to accommodate information sharing. However, this was not the approach of the COSMOS prototype implementation, nor was this consistently worked through in the final report of the project. As such, at the very least, all the models reviewed here can be criticised as being one sided in their treatment of activities.

3.5.5.2. Treatment of Objects

The three projects discussed differ in their treatment of objects. In COSMOS, a two-way distinction is made between objects which are produced in EAs and more ephemeral objects (confusingly, perhaps, these are called 'messages') which do not require an EA to be notated to represent their creation. This is an approximate attempt to distinguish between entities of varying 'life-times'. Little effort is made to explicitly model the structure of such objects. One object can be notated as a 'component' of another but exactly how components are organised into an

overall object-structure is not dealt with. Finally, COSMOS objects are not conceived of as 'active' but rather as the 'passive' contents of exchanges.

In contrast, the structure of AMIGO Advanced objects is defined in a separate model component which takes the form of a list of attributes, default/initial values and functions which can affect the values for each component of the object. A more detailed set of options are provided for to indicate the relationships between object-parts too. This more detailed approach to modelling objects is inherited by MacAll II, where objects are understood in relation to the component Iunits. In the examples available, the MacAll II project team show how Iunits can be used in modelling the filling in of intelligent fields. Iunits and messages they have compose are 'active' in the sense that they contain code that is executable in the AME for either routing or constraining the actions which can be performed on them.

3.5.5.3. Activity Modelling Reconsidered

Role-Oriented versus Mediator-Oriented

The activity modelling approach of the three projects reviewed here raises many questions. Some of these questions are shared between the projects and relate to the fundamental viability of activity modelling as an approach to CSCW. These will be addressed shortly. Hennessy et al. (1989) argue, however, that a distinction can be made between different approaches to activity modelling, reflecting profound differences in how activities are conceived. In the terms of Pankoke-Babatz (1989), activity modelling can be done from a role-oriented or a mediator-oriented perspective. A role-oriented perspective treats the roles of the different participants in an activity as primary and how these different roles get coordinated as a secondary phenomenon. A mediator-oriented perspective takes the activity as primary and regards the division of the activity into different roles (whose coordination is mediated somehow) as secondary. One way of addressing this distinction is to ask at what level the rules which model the activity reside. In the COSMOS and AMIGO Advanced projects, rules are attached to CSs and activities whereas rules are implemented in MacAll II on a per role basis. Accordingly, Hennessy et al. (1989) argue that the COSMOS and AMIGO Advanced projects pursued a mediator-oriented approach whereas MacAll II was role-oriented.

The Concept of Role

The notion of role employed in these projects has aroused some criticism. For example, Robinson and Bannon (1991) argue that 'role' confuses a descriptive with a prescriptive sense. 'Playing out one's role' can be taken to be a prescriptive matter, something one *should* do. We have noted above in the emphasis that the AMIGO project gives to what *should* happen in an activity a tendency to a prescriptive notion. On this criticism, prescribing a role is quite a different matter from merely describing *a* way of doing some activity. Through an unreflective

concept of role, some approaches to activity modelling, according to Robinson and Bannon (1991), conflate an important distinction.

Exception Handling

Whether Robinson and Bannon's (1991) criticism really bites depends upon how exceptions are handled in systems based on activity models. A system will seem much less prescriptive of *the way* of doing an activity, if some exception handling mechanism is smoothly and appropriately integrated into how it supports the execution of activities. The projects described above have not really grappled with this issue. Mac All II scarcely addresses it at all. AMIGO Advanced notes the problem and points to solutions but not in any depth. In the final report to the COSMOS project, there are a number of suggestions as to how exception handling may take place (see the description above concerning 'SPACE') but none of these suggestions were tested still less proven in any implementation.

As recognised in the COSMOS report, there is a sense in which no completely satisfying solution to the problem of exception handling can be found. If we can never be confident that an activity model covers all possible cases, how can we be any more confident that an exception handling mechanism will cover all possible exceptions. However, noting the *in principle* impossibility of a perfect exception handling mechanism does not entitle us to argue that none can be found which will be adequate *for all practical purposes*. In other words, the problematic nature of the concept of role and the difficulties of implementing adequate exception handling mechanisms cannot be taken as knock down arguments against the viability of CSCW systems based on activity modelling.

3.5.6. Some Possible Future Directions

Clearly, though, the activity modelling approach to CSCW has some difficulties. If the approach is to yield workable systems, a number of issues need to be addressed. As argued by Hennessy et al. (1989), activity modelling needs to be integrated with information sharing. It is not surprising, therefore, that much recent work in CSCW has concerned addressing questions of information sharing which are somewhat conspicuously passed over in the activity modelling work. Trevor et al. (1993), for example, see support for information sharing as a more appropriate research issue than 'heavyweight' activity modelling with inadequate exception handling mechanisms and unrealistic models of the real world. Instead, they advocate a 'lightweight' approach to activity modelling within a framework where modelling information sharing takes priority.

In the face of work of this sort, the continued viability of activity modelling hinges on whether cooperative action and activity can be re-formulated so that the difficulties with existing systems can be circumvented. This dissolves into a number of research questions. Four (of doubtless many) are identified below.

3.5.6.1. The Granularity of Action

Crudely: how 'big' (in terms of its impacts on a cooperative system) is an action? To see the sense and pertinence of this question, contrast again the approach of the COSMOS and AMIGO Advanced projects with respect to, say, the relations between passing messages and performing actions. In AMIGO Advanced, passing a message would be to perform one of the actions notated in the formalism of AAM. That is, there is a one-to-one correspondence between the actions that are legible in the formalism and those actions as they would be realised in the execution of the activity. As noted above, in spite of their superficial similarity, COSMOS' SDL works at a different level of abstraction. Notating one exchange action in SDL does not imply that this would be realised as just one message passing action. The relation could be many-one, one-many, many-many or one-one. Similarly, Bowers and Churcher (1988) criticise Winograd and Flores' (1986) work on Conversations for Action as assuming a one-one relation between illocutionary acts and the exchange of messages. In ordinary conversation, a request can be worked up over several turns, anticipated by its recipient or, indeed, never issued explicitly at all. In the COSMOS final report, a suggestion is made that an action can be regarded as (relatively) complete if an action which is anticipated as following on from it can be taken up by its initiator. This approach to action makes the completion of an action, a, an implicitly negotiated product between the initiator of a and any action which might depend on a. This flexibility may go some way to anticipating certain classes of exception an approach like that of the AAM would have to deal with. The suggestion, though, remains unimplemented and untested.

3.5.6.2. Footings

Message passing or some form of information exchange is an important component for many of these models. Indeed, it is the typical mechanism for activity coordination to take place. However, most message passing is modelled in terms of senders and recipients. This is a very crude treatment of the many and varied orientations that someone can have to communicative acts. In ordinary communication, one can overhear, pass on the words of another, be addressed in the presence of others, and so forth. Goffman (1981) addresses these issues in terms of a concept of footing (see also Jirotka et al., 1991). There are many different orientations that a participant can have to an exchange than just sender or recipient. None of the activity models reviewed here adequately appreciate this range of possibilities.

3.5.6.3. The Relation Between Modelling-Time and Execution-Time

The three projects reviewed all have a tendency to regard 'modelling time' (i.e. the time when an activity model is formulated, notated and implemented) as a discrete time prior to 'execution time' (i.e. the time when users play roles in the activity). That is, there is a tendency in the literature reviewed to think that an activity model has to be completely formulated and implemented *before* any user

can commence playing a role. This is not necessarily the case. It would be quite possible to envisage a system along the lines of TRACE (Bowers, 1990) in which activity models could be developed 'as you go' and changed 'in line' as and when problems arise by those who are playing the roles in the activity. To use and extend a metaphor, this would be like building and repairing your ship while it was taking you where you wanted to go. Of course, there are many practical implementational problems with such a suggestions but — again — exploring this possibility might be another way to anticipate what would have to be handled as 'an exception' when activity models are put in place in their entirety *ab initio*. Some of the mechanisms for 'navigating through SPACE' in COSMOS indicate such a possibility but once again the suggestion remains unproven.

3.5.6.4. Activity Models as Executable Code Versus as a Resource

Finally, one can envisage a radical change of focus for activity models in which they cease being executable code determining in a CSCW system what can be done and at what time or whatever and, instead, are seen as models in another sense — as resources for action rather than specifications of it. The determining role of the activity model on this view would be much weakened. Activity models would not determine action in any direct way. This suggestion echoes Suchman's (1987) familiar remarks on plans and their relation to action. For Suchman, a plan does not determine action, it is a resource for it (along with the actor's embodied skills and the specificities of the situation in which the action gets performed). Exactly what such a reorientation would mean in detail is an open issue to be explored but it would point to different criteria for activity models. One would be less concerned with the descriptive adequacy of an activity model than with (perhaps) its manipulability, shareability, combinability, durability, mobility, visibility, provocativeness and so forth (see Latour, 1986; Bowers, 1992).

3.5.7. References

Benford, S. D. (1988). A data model for the AMIGO communication environment. In R. Speth (ed.) *EUTECO '88 — Research into networks and distributed applications*. North Holland: Elsevier.

Bowers, J. M. & Churcher, J. (1988). Local and global structuring of computer mediated communication: Developing linguistic perspectives on group working. In *CSCW '88: Proceedings of the second conference on Computer Supported Cooperative Work*. New York: Association of Computing Machinery.

Bowers, J. M. (1990) *Narratives of technology*. Department of Psychology, University of Nottingham, U.K.

Bowers, J. M. (1992). The politics of formalism. In M. Lea (ed.) *Contexts of computer mediated communication*. London: Harvester.

COSMOS (1989). Specification of a configurable structured messaging system, edited by R. E. Young. Cosmos Coordinator's Office, Queen Mary College, London.

Dollimore, J. and Wilbur, S. (1991). Experiences in building a configurable CSCW system. In J. M. Bowers and S. D. Benford (eds.) *Studies in computer supported cooperative work: Theory, practice and design*. North Holland: Elsevier.

Goffman, E. (1981). *Forms of talk*. Cambridge: CUP.

Hennessy, P., Benford, S. & Bowers, J. M. (1989). Modelling group communication structures: Analysing four European projects. In *Singapore International Conference on Networks '89*. New York: IEEE.

Jirotka, M., Luff, P. and Gilbert, G. N. (1991). Participation frameworks for computer mediated communication. In L. Bannon, M. Robinson and K. Schmidt (eds.) *ECSCW'91: The Second European Conference on Computer Supported Cooperative Work*. Dordrecht: Kluwer.

Latour, B. (1986). Visualization and cognition: Thinking with eyes and hands. *Knowledge and society: Studies in the sociology of culture past and present,* 6, 1-40.

Robinson, M. and Bannon, L. (1991). Questioning representations. In L. Bannon, M. Robinson and K. Schmidt (eds.) *ECSCW'91: The Second European Conference on Computer Supported Cooperative Work*. Dordrecht: Kluwer.

Searle, J. R. (1969). *Speech acts*. Cambridge: CUP.

Simone, C., Grasso, A. and Pozzoli, A. (1993). *Mechanisms of interaction based on conversations*. COMIC document MILAN-3-1.

Smith, H., Hennessy, P. and Lunt, G. (1991). An object oriented framework for modelling organisational communication. In J. M. Bowers and S. D. Benford (eds.) *Studies in computer supported cooperative work: Theory, practice and design*. North Holland: Elsevier.

Suchman, L. (1987). *Plans and situated action: The problem of human-machine communication*. Cambridge: CUP.

Trevor, J., Rodden, T. and Blair, G. (1993). COLA: A lightweight platform for CSCW. In *ECSCW'93: Proceedings of the Third European Conference on Computer Supported Cooperative Work*.

Winograd, T. and Flores, F. (1986). *Understanding computers and cognition: A new foundation for design*. Norwood, NJ: Ablex.

3.6. USENet News, GRACE, ATOMICMAIL, Active mail, and GROVE

Jordi Íñigo Griera, Nemer Elkassem

Universitat Politècnica de Catalunya

3.6.1. USENet News

3.6.1.1. Short genesis and description

This is a worlwide bulletin board system in which thousands of computers, pass articles back and forth (several megabytes per day). The system was developed shortly after the release of V7 Unix with UUCP in 1979, and the subsequent apparition of the USENet. The next year appeared the first version of the news software implemented in UNIX. Nowadays, there are implementations in most of the relevant operating systems.

Basically, the system is a distributed database with the information replicated on all the server nodes. Users can access information either locally or remotely through well-known protocols.

The system can easily maintain either world-wide groups or local groups. These groups are open to everybody that has access to USENet. The users exchange plain text (but they exchange often, software and even pictures and sounds coded in wide-known formats), inside internet-like messages.

MoIs to support Sharing Knowledge

The only formally defined interaction in News is the way in which people know and share the structure of the information –for this reason the main News MoI is *Sharing knowledge*. There is an agreed way by which all the articles are organised and a methodology to follow on on-going conversations. They are classified according to two attributes:

- within a topic (*newsgroup*)
- conversation title (*subject*)

The topic pre-exists the conversation and is quite static. This allows users to subscribe the topics of their interest, filtering part of the huge amount of information. The policy associated to this MoI is to reduce the effort in reaching conversations of our interest. This MoI enables us to contact people interested on the same topics as ours.

The second aspect that marks a conversation is the *title* (subject) of the initiating article. This allows the people already on this topic to join or discard that conversation (thread) only by seeing the title of the first article. The policy associated

to this MoI is also to reduce the effort needed in reaching the information of our interest.

MoIs to support Human to Human Communication

The basic mode of interaction is asynchronous conversation. Users interact in non-structured conversations, in the sense that there is no kind of constrain in either the order of participations or the content of the interaction. Conversations appear dinamically, without a previous definition. This mechanism can be considered of the Human to Human Interaction type.

Antoher MoI that supports communication (constraining it a little more) is present in the system. Some of the groups have a *moderator* who checks the worthiness of the contribution before it reaches the rest of the people. The goal is to increase the quality of conversations above a minimum average. However, normally in this class of groups there are no conversations since all contributions are supposed to be of "unquestionable" value.

USENet News has another mechanism that was not implemented from the beginning, that guides newcomers when they access a group for the first time. This MoI (?) is represented by an article that is present in most of the groups called *FAQ* (*Frequently asked questions*). It exposes background knowledge that can be useful to the newcomer. Normally there are experts in each group that have the role of writing regularly these special articles.

3.6.1.2. The Notation

This mechanism is specified in a very simple notation that specifies the (news)groups, and the topics they refer to. This could be a part of the hierarchy of groups (that contents a mixture of local and global groups):

comp.binaries.ibm.pc Binary-only postings for IBM ...
comp.binaries.ibm.pc.d Discussions about IBM/PC bin ...
comp.sys.ibm.pc.hardware XT/AT/EISA hardware, any ven ...
comp.os.linux The free UNIX-clone for the ...
sci.astro Astronomy discussions and in ...
comic.strand3
comic.strand4
comic.misc

Also conversations, can be sorted by title. All the participations in the same conversation share the same subject.

3.6.1.3. Features

Generality

The basic News MoI (of sharing knowledge type) is not general. It does not allow us to specify roles to users (with the exception of moderators), so all the users can access any of the groups without restrictions. Furthermore, any user can change the information hierarchy of all USENet.

Malleability

The information hierarchy can by modified by users. But the way the MoI behaves cannot be modified.

Visibility

At any time, it is possible to change the membership from a group to another, knowing at any moment the *history* of the new group. Also the names and topics of all the groups are available to all users.

Propagation of changes

The changes of the system are propagated to the neighbours (flooding machanism). So its possible that a change (a new article in a conversation or a new group) made in Europe needs one or two days to reach America. The only case in which this delay could pose a problem would be when a group is deleted (some people could send a contribution to a non existent group). But this fact seldom happens because normally the only groups that are deleted are those without postings.

The Context Maintained

Any change on the hierarchy does not disturb the rest of the groups. However the new group cannot inherit the contents of the old one it came from.

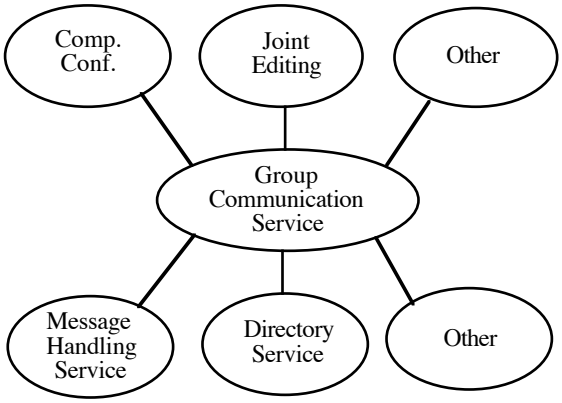
The Degree of Openness

USENet News has easy access to mail through gateways that convert articles to mail (Internet mail). Also there are gateways to more modern services as Gopher.

3.6.2. GRACE

3.6.2.1. Short Genesis and Description

The GRACE (Group Activity Environment) is a system that was designed to support group communication activities within an ISO networking environment. The design adopted for GRACE, was based on the notion of a generic Group Communication Service (GCS). The new facilities provided by this new service include a Shared Information Base, a global name space for group communication objects and co-ordination of underlying OSI services such as MHS (X.400) and the Directory Service (X.500). The relationship between the GCS, applications and OSI services is explained in the following figure:



MoI to support shared knowledge

This aspect of GRACE is defined by its capacity of representing organisational structures, activity groups, people associated to them, and the roles associated to these people. The model implements an object-based view of an information sharing universe in which information objects inhabit a set of domains. Domains may represent applications, organisations or role environments and principally exist to facilitate the management of objects.

MoI to Support Human to Human Communication

The MoI for communication used in this system is the asynchronous conversation. Users can establish conversations by joining or creating activity groups, that are represented by clusters (a special kind of object) to which the people submit their messages in an asynchronous manner.

3.6.2.2. The Notation

The Notation for the MoI

The GRACE conceptual model makes it easy to describe group communication applications in terms of sets of primitive services. This is due to the fact that this model consists of two components; *Information Model* which represents information structures required for different activities and *Abstract Operations* to define how to manipulate those information structures. So the communication applications dispose of a base set of object classes from which they can create or derive their own specific objects. Moreover, to implement their facilities these applications dispose of a set of elemental operations for these objects.

Information Model has these generic objects:

- *Item*: the basic unit of information. Although an item may have a complex internal structure as represented by its attributes, it is treated as a single unit.
- *Cluster*: group of items and/or other clusters (it is not necessary to use a hierarchical structure).
- *Entity*: it is the representative of a group member (normally a person, but sometimes it can represent an automatic agent).

- *Domain*: provide mechanisms to link group entities with the relevant information structure. It partitions the global information space into logical areas. The model defines three types of domains: organisational, application, and role domains.

  Organisational domains represent the entities and resources of the organisation while the *application domains* provide mechanisms to associate other objects with the application. *Role domains* give support to some specific roles of communication within the application.

The above definitions restrict the behaviour of the applications built on top of GRACE. Hence, the MoIs derived from those applications will work in a similar manner.

The general form in which communication takes place is by creating clusters of objects to which different entities can access, and insert or retract objects. Each application can create or derive its own clusters from the basic ones, in an object oriented way.

The notation used within GRACE to name objects consists of an ordered sequence of attribute type/value pair, each one divided into two components: the <domain part> (representing the naming authority) and the <local part>.

Abstract Operations define the GRACE Access Protocol (GAP). In broad terms, the operations support the retrieval, browsing and modification of the objects specified above. Each operation is informally described by the function signature.

The Notation for the User Interface

GRACE disposes of a graphical user interface (GUI) with X Windows. This interface was designed to be flexible enough in style to support any GRACE asynchronous activity. This tool is offered to the applications implemented on top of GRACE.

The operations made available to user via the GUI encompass those operators identified by the abstract service definition. They are structured so as to reflect the objects in the mapping to the GRACE generic model.

Each GRACE object is presented in the GUI as a horizontal bar which has two key components; “Information field” which contains text which identifies the object and “Operation Button” which gives access to the operations which can be invoked on a given object.

3.6.2.3. The features of the mechanisms

(GRACE is not an application itself, so the following features might not be maintained by the derived applications.)

Generality

The information model represents both information and the members of groups as objects. GRACE gives us an infrastructure to build on top of it a shared data base

of such objects, allowing us to construct a tailored application based on the information defined by these objects.

It is general since we can define any structure information with any kind of attributes in a GRACE based application, .

Visibility

As a shell GRACE provides application designers with operations that allow him/her to consult the mechanisms of interaction. It is possible to realize the actual roles of the people, the workgroups, domains, etc.

Malleability

GRACE provides operations to manipulate the structure of the information database at run-time. However, it is up to the GRACE based application designer to allow users to modify the structure of the mechanism.

Level of Abstraction

The notation is at a level of abstraction appropriate to a CSCW shell, since it allows us to build specific definitions to the application implementation.

Propagation of Changes

Since GRACE has a distributed architecture, it has to deal with the replication issue. GRACE solves that problem allowing only a master-copy and several slave-copies, forcing changes to be made to the master-copy.

The context maintained

In GRACE applications the context can easily be maintained since it offers operations to store and arrange old objects (e.g. clusters). This, of course, should be a chief feature in a sharing knowledge system –and it is so in GRACE, since it aims to make information available to all users.

Obviously, it is up to the application designer to develop or not this feature.

The degree of openness

GRACE is an open system in the sense that it was built as a platform on top of several standardised services (X.500, X400). The information structure is partially maintained on X.500 (some objects are stored on a commercial database). This fact allows non-GRACE based applications to use the information stored in X.500 by GRACE based applications. However, until now, the use of X.400 is not yet implemented.

3.6.3. HelpDesk

3.6.3.1. Short Genesis and Description

The Help Desk is a demonstrable prototype of the GRACE system. It was implemented to test the conceptual model of the system and to discover any limitation in its early stages.

The goal of this application is to provide users with an organisational inquiry service set in a Computer Service Department (CSD). This inquiry service called frequently Help Desk Application Service enables individual users to obtain the assistance of the CSD to answer questions about computing facilities.

MoI to support sharing knowledge

Before the initialisation of the Help Desk Service, some domains (which represent the environment) like the organisational domain, role domain and other components must be present. After the initialisation of this scenario a HelpDesk application domain is created along with role domains for Inquirers and Advisors. Two clusters are also created; a special cluster to maintain inquiry forms and another one to register all on-going inquiries.

A user who plays the Inquirer role can see two available clusters of information: Inquiries cluster which contains all inquiry forms created but not submitted; and On-going Dialogue which contains all dialogues in which the inquirer is currently involved. An entity (individual) who plays an advisor role can see two available clusters: Outstanding Inquiries that contains Inquiries submitted but still not assigned to any Advisor; and Ongoing Dialogues which contains the dialogues in which the Adviser is currently involved.

MoI to Support Human to Human Communication

After an inquiry is assigned to an advisor, an asynchronous conversation is stablished between advisor and inquirer. This conversation cannot be participated by any other user. All steps in the conversation are maintained by the system, and are accessible to both participants at any moment.

MoI to support action

The only MoI to support actions we can see in this implementation is that presented by the *in-tray* method to schedule inquiries on advisors. Here a rule is to be followed to take responsibility of some inquiry. An advisor browse the list of *Outstanding Inquiries*. He reads, then chooses to ‘Accept’ our Inquirer’s forms. In so doing he assumes responsibility to answer the inquiry. The artefact in this context is the Outstanding Inquiries pool (cluster).

3.6.3.2. The Notation

As an application built on top of GRACE, the Help Desk derives its own specific objects from the set of object classes defined by GRACE. Thus, it imports the

people who may be Advisers or Inquirers from the Directory and grants them the ability to play these roles depending upon whether they are marked as specialists or computer users respectively in their own companies.

On the other hand, to implement its facilities of interaction, the Help Desk defines new GRACE object classes as shown by the following object hierarchy; As instances of the generic basic unity of information(*item*) defined by the GRACE model the Help Desk creates what it calls inquiry description, inquiry form and responses. As an instance of the group of items(*cluster*) of the generic model, the Help Desk defines its own inquiry forms, set of inquiries and inquiry dialogue. And finally it derives its own Application, Organisation, Work Space and Role domains from the generic ones.

If we analyse the notation used in the mechanisms of interaction of communication, we find that the conversations between the Adviser and the Inquirer takes the form of question/response pairs. To support this conversation, the Help Desk derives its own operations like: *ListInquiryForm, PostInquiry, CancelInquiry, PostResponse, ListDialogue, CreateInquiryForm, ListInquiryForm* and other operations from the Generic Abstract Operations (GAP) offered by GRACE conceptual model for manipulating objects. Some of these operaations are *Create, Delete, Read, Modify, iInclude, Remove, Search,...*

Notation for the user interface

The Help Desk operations made available to the user via GUI are derived from thosse identified for the abstract service definition described above. They are structured so as to reflect the objects in the mapping to the GRACE generic model. The user dialogue is hierarchical in nature, with *parenthood'*loosely corresponding to *inclusion of* in the mapping. Each node in the hierarchy represents a GRACE object which can represent terminal or non-terminal nodes. Non-terminal nodes typically represent GRACE objects which can include others, i.e. those subclasses from the GRACE generic *Domain* and *Cluster*. Terminal nodes, on the other hand, represent subclasses of the GRACE generic class *item*.

As all GRACE objects, an objects within the Help Desk Application is presented in the GUI as a horizontal bar with two key components:

Information field: contains text which identifies the object.

Operation Button: Gives access to the operations which can be invoked on a given object.

3.6.3.3. The features of the mechanisms

Generality

The mechanism is not general. First of all, it just allows interaction between two users in its second stage. When the advisor takes the responsibility of an inquire, this is withdrawn of the general *in-tray*. From that moment interaction is constrained to two users.

At the first stage, there is a group of members formed by all inquirers and advisers, but it is a particular interaction since it is finished (with the stage itself) at the very moment an adviser accepts the responsibility of an inquire.

Malleability

This application does not inherits the malleability features of GRACE. It is not possible to change the behaviour of the mechanism (not without re-compiling the whole application). It is only possible to change members and role assignation at run-time.

Visibility

It is possible at any moment to see again old conversations (inquires, answers, explanations, etc.), to query about personal information of participants, etc. Furthermore, it is possible to know the stage of any of the current conversations on which we participate.

Level of Abstraction

The level of abstraction is appropriate to the purpose of this system. Inquirers can make questions without knowing a priori who is more suited to answer. The central artefact (the Outstanding Inquiries) serves this purpose. Private trays maintain the on-going conversations between inquirer and advisor.

Propagation of Changes

The mechanism cannot be modified at run-time, and the trays are maintained by just one server. Then, there are no problems to maintain coherence, because it is not possible to loose coherence on the system.

The context maintained

Domains that represent the entities and the resources of the organisation are initially present before the Help Desk being invoked. During a Help Desk session the contributions are maintained. A Work Space (a role domain) is created for the Inquirer role and another one for the Adviser role, and contain the Inquiry Dialogue associated to each of them.

Once the Inquirer is satisfied that his/her Inquiry has been answered, then s/he may request that it should be cancelled (thus eliminating the context). The Adviser then removes the Inquiry and its associated dialogue from her/his Work Space and it is up to the Inquirer to do so in her/his own Work Space.

The degree of openness

As an application implemented on top of GRACE, the Help Desk makes profit of the interface that GRACE has to the Directory. For example it imports people who may be Advisers or Inquirer from the Directory and grants them the ability to play these roles. So it is possible that other non-GRACE applications can make use of the objects created by this application.

3.6.4. Mail based systems

E-mail systems have been used as a platform to build some systems on top of it. The first one, and most used are distribution lists. Lately, it has appeared more complex systems like the Active Mail and the ATOMICMAIL that are reviewed here.

As in the case of News, here the MoIs involved (and the artefacts used) are almost always asynchronous. For this reason, these systems imply *non-intrusive* methods. Therefore, the reaction of the recipients does not need to be immediate (e.g. they can delay the reaction until they have time to do it). Nevertheless, some kind of interactions could be really hard to carry out, due to the delay between actions. These delays should not be always attributed to the system's response time (local mail could be instantaneous) but to the nature of the systems themselves and MoIs associated to them.

3.6.5. ATOMICMAIL

3.6.5.1. Introduction

The basic idea of *computational e-mail* is very simple: Instead of sending plain text in the mail message, what is sent is a computer program in a common well-defined language. At the recipient's end, the mail-reading software recognises that the mail message is intended to be executed rather than read.

With this basic idea in mind, and putting the stress on security issues, ATOMICMAIL was developed. Basically, programs are a sequence of input-output primitives, like get or print a string, select in a multiple-choice question, or fill in the form. This allows us to make different applications on top of it.

Examples of applications that have been built on top of this system are: a survey generator, a meeting scheduler, a document distributor, an information finder, an activist alert mechanism, a distributed mail-based game, and dynamic mailing lists.

3.6.5.2. The basic MoI

As a shell, it has no implicit MoIs, but from the primitives it has, it is possible to know how an ATOMICMAIL based application would behave. (*see notation part*).

When the recipient reads the message, the message may first show him some introductory text, and then engage him/her in a question-answer dialogue. For example, it may ask him to select which of ten proposed dates and times he can make it to a meeting. The program may then mail the answers back to a network server that was collecting such answers from people, in order to find a meeting time suitable to all of them.

The MoIs associated to this system depend on each message but normally they are a mixture of the fill-in-the-form-and-send-me-the-answer type and unconstrained conversation type.

Other MoIs that are slightly different from its e-mail counterparts are the *dynamic mailing lists*, that allow a more flexible way to engage-disengage lists/groups, near to the mechanism seen on the News. ATOMICMAIL allows the initiator to predefine a list of members (in opposition to News, here the groups are closed). Each of the participants can leave the group whenever they want.

3.6.5.3. Notation

The notation is built around a short number of primitives. These primitives mainly relate to input and output handling. They are designed to be hardware independent (i.e. they work both in a X-Terminal and in a text terminal), therefore the local agent is the responsible to adapt each primitive to the particular environment of the user.

The main primitives are related to:

- Get text from the user.
- Get a number from the user.
- Ask the user a multiple choice question.
- Fill in the blanks in a form.
- Display a chunk of text to the user.

It uses an implementation of LISP, without all the standard input-output facilities, that where replaced by new input-output mechanisms to not compromise security. The fact that LISP is an interpreted language makes it easy to control the execution of messages, and it is machine independent. Here is a piece of an ATOMICMAIL programme:

```
(handle-survey
    "nsb" NIL "Re: test"
    '((survey-complex-form "Please answer the following questions to help us with our research:"
  (("How many email messages do you receive each day?" "integer")
  ("How many messages do you send each day?" "integer")
  ("Does email enhance your productivity, overall?" "boolean")
  ("Do you use email for non-business purposes?" "boolean")))))
```

3.6.5.4. The features of the mechanism

Generality

With ATOMICMAIL structured messages with any content can be defined. The *artifact* (represented by the message) can process somehow the reply, and it could send back the result if this was foreseen by the initiator of the interaction.

However, this mechanism does not permit the interaction among more than two participants[1].

Visibility

The visibility of messages is restricted to the sender and the recipient. Other users cannot join nor see the state of the interaction.

Nevertheless, in some specific applications (as in dynamic mailing lists) it is possible to implement mechanisms to allow new users to join an access already started conversations.

Malleability

The mechanism cannot be changed after the message has been defined. The recipient only can decide whether s/he answers or not, but if s/he does so, s/he is constrained to fill the blanks of the message.

Level of Abstraction

The system was designed to facilitate the processing of answers to message queries. The implemented language is able to do it. However, the system does not have predefined procedures to process the answers received.

Propagation of Changes

After the query is defined and sent (the interaction begins), there is no possibility to be changed. Therefore, no changes should be propagated.

The context maintained

The context is embedded in the message itself. It is fully maintained in the interaction.

The degree of openness

ATOMICMAIL is independent of the type of mail used. It is also prepared to work in different kinds of terminals (from ASCII to graphic terminals).

3.6.6. Active Mail

3.6.6.1. Description

Active Mail is another enhancement to the plain mail system. It maintains the original functionality e-mail had like ATOMICMAIL does. Nevertheless, instead of sending a kind of batch-programs to the recipient, Active Mail supports ongoing interaction between sender, recipient and future participants through *direct* (non-e-mail) *communication channels*.

---

1 In fact, the initiator can send the same message to several people, but each of these recipients should reply to the initiator. Therefore, this ‘broadcast’ of messages are merely interactions between two users.

With the tools that Active Mail has, several applications have been designed: Conversation Agent (Asynchronous), Shared-Document Agent (floor passing), Meeting Scheduler (negotiation method), etc.

3.6.6.2. The Basic MoI

The knowledge of the existence of those channels can be shared with other users through e-mail. All people that has received a channel is enabled to join the group. That group is defined upon the control of a common application, which is in charge to manage the group. It is this application who defines the mechanism of interaction that will be followed by members. Even though applications can use direct connections, the applications referenced above are mainly asynchronous.

The use of electronic mail avoids intrusive actions like blocking the console until getting a response from the called (Active Mail notifications are simply a change in the aspect of its icon), delaying the response until the called reads the incoming mail and decides to answer.

As in electronic mail two kinds of messages exist: the user mail-spool, and old messages in local folders. However, in opposition to passive messages, the active peers can still communicate with users and applications long after they were received (when they are already in a local folder).

3.6.6.3. Notation

Active Mail system has *agents*, and dual-ported *channels*. Agents are both applications and users, and can be connected through those channels. An user is participant in an application if there exists a port connecting her to the application. If that user wants to involve some other in the application (the group), he should send a copy of the application port to the new user.

Active Mail gives procedures based on these ideas, and the application designer should program the application using these tools and libraries, but there is not an explicit notation to build artifacts or mechanisms.

3.6.6.4. Features

Generality

This approach is general in the sense that the application programmer can do anything with those tools, but the system does not give us any support to create new mechanisms without recompiling the whole application.

Malleability

Active Mail imposes a very strict behaviour on the process to join applications. To create new groups the *initiator* should send an active message to the desired participants, and create an *application agent* that will be the representative of the group. After that, the malleability of the system is dependent of each application.

Visibility

The system maintains the old messages. In this case this is more important than in plain e-mail since old messages maintain on-going conversations. The messages give the *active connections* to the applications agents.

When some change on the application agent occurs the message that is connected to it notices the change, and signals it to the user.

However, the description of the mechanism of interaction is not visible to the user, since there is not an explicit description (a real notation) but a compiled program.

The Context Maintained

The message exchange is maintained in each user's folder. The state of the current activity is updated by sending the copy of the message to all users. It is a matter of each application agent to handle appropriately the changes that affect it.

Propagation of changes

With Active Mail it is possible to change membership easily by sending an active message to them. How the status of the application agents is maintained is a matter of the application itself.

The Degree of Openness

The applications built with Active Mail use standard e-mail to create and maintain groups. However, non-e-mail channels can be used by the applications that are specifically designed for Active Mail.

3.6.7. GROVE

3.6.7.1. Description

GROVE allows a (small) group of users to simultaneously edit a text (an *outline*). It provides users with windows through with they can see a part of the document, depending on some specified privileges.

One of the most relevant features of this system is its build-in reordering mechanism, that allows users to modify the same area without any loss of coherence among the replicas. This is mainly used, not to work like this, but not to oblige users to make explicit demands to write on the desired area. This factor, allows users to work in a more relaxed way, and to have a shorter training period. Nevertheless, they do not collide so often, mainly due to the fact that they can actually see where and in what others are working.

3.6.7.2. The basic MoI

A GROVE interaction (*session*) is formed by a group of users, each of them with a view of the work document through one or more *views*. The document is divided in areas depending on the access rights of each member. These areas in the

document can be divided on three categories: visible to everybody, visible to a specified group of users, or private.

But despite these initial access privileges, users can access and modify any area without have into account what the others are doing. GROVE give participants synchronous information of who is working at any moment, where, and what s/he is doing.

3.6.7.3. Notation

There is not an explicit notation describing the interaction. Groups work around a shared document in a session. Each of them can choose to work on one or more views of the document depending on the access capabilities in each area. Each view can be *private* (if it contents items only visible to an user), *shared* (if it contents items visible to some users) and *public* (it contents items readable by everybody).

3.6.7.4. Features

Generality

GROVE is an application dedicated to cooperative editing. It is general in the sense that it allows both work in a synchronous and asynchronous manner. Also, it can divide the document contents in sub documents and give each user different rights, allowing to the cooperative ensemble manager to direct the work and divide effort to each participant. It also allows users to work in an open way without any preliminary access restriction.

Malleability

The system is malleable mainly due to the lack of constrains that support members. The purpose of the work and the method of work can be changed without changing the mechanism, because the mechanism does not impose constrains.

Visibility

The state of the document is visible at any time. Also, each user is aware at any moment of what others are doing. The questions: where, what or who is doing something?, are easily answered by the system. However, the system does not facilitate answers like why is he doing that?.

The Context Maintained

Each user has an actualised view of the state of work (document contents) and the state of the interaction (position of users, their actions, their rights, etc.).

GROVE does not maintain the history of the interaction. Only locally, is possible to maintain copies of old versions of the work.

Propagation of changes

The changes are propagated automatically (and synchronously), keeping synchronised the global work. It is a principal characteristic of the system.

Level of Abstraction

The system is at a right level of abstraction. The actions needed to successfully work with a shared document have its counterparts in the system. The setting of the ensemble is easily done by members.

The Degree of Openness

GROVE has little capacity to connect with other systems. It is designed "as is".

3.6.8. References

[1] Nathaniel S. Borenstein. *Computational Mail as Network Infrastructure for Computer-Supported Cooperative Work*. Proceedings of the Conference on Computer-Supported Cooperative Work. 1992.

[2] Yaron Goldberg, Marilyn Safran, Ehud Shapiro. *Active Mail —A Framework for Implementing Groupware*. Proceedings of the Conference on Computer-Supported Cooperative Work. 1992.

[3] Clarence. A. Ellis, Simon. J. Gibbs, Gail. L. Rein. *Groupware: Some issues and experiences*. Communications of the ACM, vol. 34, #1. 1991.

[4] USENET News. See RFC Documents.

[5] Steve Benford, Adrian Bullock, Paul Harvey, Howidy Howidy, Alan Shepherd, Hugh Smith. *The GRACE Project. Group Communication in an Open Systems Environment.* Communications Research Group, Nottingham University. 1991.

[6] Mark Stefik, Daniel G. Bobrow, Gregg Foster, Stan Lanning, Deborah Tatar.*WYSIWIS Revised: Early Experiences with Multiuser Interfaces*. ACM Transactions on Office Information Systems 5, #2. 1987.

[7] Deborah Tatar, Gregg Foster, Daniel G. Bobrow. *Design for Conversation: Lessons from Cognoter*. International Journal of Man-Machine Studies 34, #2. 1991.

[8] *The ISIS Distributed Toolkit. Version 3.0. User Reference Manual*. Isis Distributed Systems, Inc. 1992.

[9] Clarence. A. Ellis, Simon. J. Gibbs. *Concurrency Control in Groupware Systems*. ACM SIGMOD Record 18, #2. 1989.

3.7. CoDesk and DIVE

Rune Gustavsson

SICS/KTH

3.7.1. Background

The strand 3 meeting at Risø, February 9-10th 1993, discussed several aspects of Mechanisms of Interaction (MoIs), their characteristics and use [Risø-3-4-minutes].

As a preparation of upcoming deliverables a work plan was set up. The first phase of the workplan consist of assessments of existing tools, shells and notations used in CSCW settings.

The following framework for assessments was agreed.

- Generality
- all aspects, constraints when using it
- Malleability
- or modifiability
- Visibility
- how much can be accessed and controlled — and how?
- Level of abstraction
- levels in the domain vs. implementation levels
- Propagation of changes
- context accessible from the MoI
- The context maintained
- how is the context modified?
- The degree of openness
- occasion of integration — conceptual and technological

In addition we must try to characterize the following dimensions:

- Constraints and affordancies
- Function and purpose

It is also important to characterize the level of usage in terms of

- Organization
- Task
- Activity

At the Strand 3 meeting 17-18th of June in Milan, the framework was further discussed and elaborated. At that meeting it was agreed that SICS would extend

the earlier SICS-3-2 report on CoDesk and DIVE and incorporate some new aspects.

3.7.2. CoDesk

3.7.2.1. Introduction

CoDesk [Pehrson ed. 1992] is a tool oriented shell for CSCW. CoDesk is a pilot project within the Swedish research program MultiG. CoDesk, together with KnowledgeNet , is an environment supporting distributed collaborative design. CoDesk is at present evaluated in different experimental settings such as KnowledgeSharing93.

The Collaborative Desktop, CoDesk, consists of a set of generic tools for CSCW. CoDesk is integrated with the private desktop. The Collaborative Desktop is an attempt to make collaboration a natural part of the daily use of a computer. It is an environment that lets users work together. CoDesk is an interface to TheKnowledgeNet.

TheKnowledgeNet is a vision of a system for collaboration in teams where the members have access to a common base of information. The team members contribute to the KnowledgeNet with their knowledge both in the form of documents and as "mental knowledge", made attainable for other team members through shared information about "who knows what".

3.7.2.2. Basic assumptions of the CoDesk approach

The basic assumptions behind the design of CoDesk and KnowledgeNet are the following.

- The organization of design tasks is dynamic. That means, that from a CSCW point of view, we do not have an organizational structure, based on task decomposition, to support .
- The work is organized from a net of knowledge, the KnowledgeNet. The agents of the KnowledgeNet knows about other agents capabilities and work is organized according to the competencies needed for a specific task (design).
- The design of CoDesk, Collaborative Desktop, is tool oriented. The CoDesk is an interface to KnowledgeNet giving the users an environment facilitating informal and formal collaboration, shared objects and communication tools.

In summary, CoDesk and KnowledgeNet can be seen as tools supporting "double level language", i.e. the both the cultural and formal levels of communication [Robinson 1993].

3.7.2.3. General requirements on CoDesk

CoDesk facilitates communication via integration of different media such as

- text,

- graphics,
- shared windows,
- sound and
- video.

The collaborating persons are primary objects in a direct manipulative interface, by which they can form groups and start collaboration based on a room metaphor. Spontaneous meetings are supported as invitations to communicate when team members happen to use the same tool, e.g. a bulletin board, and as an electronic hallway, where one can observe allowed "disturbance level" of group members, e.g. as opened or closed doors.

The CoDesk also offers a "cut-and-paste" service for transferring data between collaborative tools and between private applications and collaborative tools.

3.7.2.4. Objects and tools in CoDesk

The objects in CoDesk are as follows.

- Persons, represented both as icons and forms ("cards"). The attributes includes private, named, mailing/communication lists of other persons. The KnowledgeNet who-knows-what information is also part of the Person's knowledge.
- Groups, are collections of Persons and/or other groups. We thus can have a hierarchical group structure.
- Tools and material.
- Rooms, furnished with persons/groups, tools and material.

A user starts a new collaboration by creating an environment for the collaboration. This is done in the following three steps.

1. Creating a room box and furnishing it with persons , tools, material and (optionally) a meeting time.
2. Make the system ask the tentative participants if they want to participate.
3. When all have responded, the initiating person can push "start communication" button. The participants then are connected by the communication media chosen.

The CoDesk tools are of the following three main types.

- Tools for creating persons, groups and rooms.
- Communication environment tools.
- Co-editing tools that operate on material (e.g. text, graphics, sound and/or video)

The CoDesk prototype is realized on Sun Sparc work stations with video cards and audio equipment and implemented in a UNIX/X11/C++ environment.

3.7.2.5. Assessments of CoDesk

Generality

The team members are primary objects in a direct manipulative interface, by which they can form groups based on a room metaphor. The object oriented and direct manipulative approach is also used in the interface to the generic collaboration tools.

When a team member wants to cooperate with some persons;

- she/he creates a room and put persons and tools into the room by dragging icons from the user folder and the tool folder (see below).
- she/he pushes a "start button" and the system ask the other persons if they want to participate.

Malleability

It will be possible to add new CSCW tools. New persons can be enrolled to CoDesk.

Visibility

Every person can create new room. These rooms are private room, e.g. only the person who created the room sees the room and the members and tools in the room.

There are four folders that are visible to all users.

User — a folder with all the persons that are members in CoDesk.

Tool — a folder with all the tools available in CoDesk.

Document — a folder with documents.

Expert — a folder with expert areas and for every expert area there is a folder with all the persons that are interested in this area.

Level of abstraction

No assessment.

Propagation of changes

Changes in the folders will be shown to each user's CoDesk by reading from the database. In CSCW tools (distributed applications) changes are distributed. For example the multi drawing editor (in CoDesk): all objects that one person is drawing are distributed to all the other person's (persons in a group) editor. Every person sees the same picture with the same objects.

The context maintained

Create working groups (rooms).

Adding new tools.

Adding new person.

The degree of openness

No assessment.

Constraints and affordancies

No assessment.

Function and purpose

No assessment.

3.7.3. The MultiG Distributed Interactive Virtual Environment — DIVE

3.7.3.1. Introduction

DIVE, Distributed Interactive Virtual Environment, is a framework for building and experimenting with applications that use a variety of modern Virtual Reality techniques, that is, 3D visualization and interaction, shareability, distribution and multi-user capability. Using its model of distribution , a large number of users and applications can participate and interact with each other in the same virtual environment [Fahlén Jää-Aro eds. 1992].

Of special interest for the CSCW domain is the space based interaction models being developed within COMIC Strand 4 by Benford, Fahlén et al.

3.7.3.2. Basic assumptions of the DIVE approach

A virtual world is an interactive simulation of a real or imaginary world. By its distributed architecture, the DIVE is especially tuned to multi user applications, where several participants may interact even though they may be geographically dispersed.

There exists a multitude of DIVE applications. Some applications address how to map traditional 2D user interfaces into the 3D realm, others concentrates on multi user aspects, such as CSCW, in a virtual environment.

Important features of the DIVE are:

- A peer-to-peer distribution model that enables the construction of multi user, multi application environments, different from many client/server approaches.
- User related higher level concepts which enables application designers may easily construct interactive VR applications.
- Support for direct, real world metaphors in applications.

3.7.3.3. Overview of the DIVE system

A participant in a DIVE world is either a human user or an application process. Users navigate in 3D space and may see, meet and collaborate with other users and applications in the environment.

A user sees and interacts with a world through a rendering application called a visualizer.

Several processes serving one human user may be clustered into a person group, where each member process may control one aspect of the user's interface.

Higher level concepts such as *vehicles*, *visors*, *behaviors* and *person groups* are supported by DIVE.

Concepts such as *aura*, *focus* and *awareness* have also been partly implemented in DIVE.

Currently, DIVE runs on SUN4s with GS or GT graphics hardware, IBM RS6000s with graphics hardware that supports GL and on Silicon Graphics workstations.

3.7.3.4. Assessments of DIVE

Generality

DIVE provides a general purpose framework for any kind of collaborative work and visualization situation, especially when complex spatial metrics and distribution capability are issues.

On the system level, a DIVE application can run in a heterogeneous networked environment (several different platform types are supported simultaneously).

Malleability

DIVE can be dynamically modified on nearly any level. Users, applications and data can be added and removed at any time without disruption to the systems operation.

Visibility

Everything in a SworldS is visible and accessible to users present in that world. It is possible for users to move from one world to another at will.

Level of abstraction

Graphical 3D-objects representing environments, applications, tools, things and users. A concept of logically separate SworldsS can be used to partition domains, spaces and applications.

Propagation of changes

Changes made at one node by an application or user are (usually) immediately distributed and being made accessible to all other nodes(users) being present in that world (process group).

The context maintained

By moving and interacting in virtual space, a user or application can access information, tools and affordancies.

The degree of openness

The DIVE environment and system is completely open, from the way synthetic environments are accessed down to the availability of the source code for the system.

Constraints and affordancies

Few constraints and lots of affordancies.

Function and purpose

See above.

3.7.4. References

[Fahlén Jää-Aro eds. 1993] Fahlén, L., Jää-Aro, K-M. eds.: *Proceeding of the 5th MultiG Workshop,* Royal Institute of Technology, Stockholm, 1992.

[[Pehrson ed. 1992] Pehrson, B. editor: *Proceedings of the 4th MultiG Workshop*, SICS, Kista, 1992.

[Risø-3-4-minutes] P. Carstensen, C. Sørensen: *Work package 3: Notations for Mechanisms of Interaction. Minutes from Risø workshop*. Risø-3-4-minutes, 1993.

[Robinson 1993] Robinson, M.: *"As real as it gets ..." Taming models and reconstructing procedures*. COMIC-SF-4-3, 1993.