

Coordination mechanisms: Towards a conceptual foundation of CSCW systems design

KJELD SCHMIDT* & CARLA SIMONE*

* *Systems Analysis Dept., Risø National Laboratory, Roskilde, Denmark, Email: kjeld.schmidt@risoe.dk*

* *Department of Computer Sciences, University of Torino, Torino, Italy, Email: simone@di.unito.it¹*

Abstract. The paper outlines an approach to CSCW systems design based on the concept of 'coordination mechanisms.' The concept of coordination mechanisms has been developed as a generalization of phenomena described in empirical investigations of the use of artifacts for the purpose of coordinating cooperative activities in different work domains. On the basis of the evidence of this corpus of empirical studies, the paper outlines a theory of the use of artifacts for coordination purposes in cooperative work settings, derives a set of general requirements for computational coordination mechanisms, and sketches the architecture of Ariadne, a CSCW infrastructure for constructing and running such malleable and linkable computational coordination mechanisms.

Key words. Cooperative work; articulation work; coordination; artifact; coordination mechanisms; CSCW environments; Ariadne.

A major research issue in CSCW is to understand how computer systems can be instrumental in reducing the complexity of coordinating cooperative activities, individually conducted and yet interdependent.

In fact, the issue was identified and defined with admirable precision quite early in the history of CSCW, by Anatol Holt: '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).

Since then, this issue has been investigated by a range of eminent CSCW researchers. The initial results were not encouraging, however, in that coordination facilities were experienced as excessively rigid, either because the underlying

¹ The first version of this paper was written while Carla Simone was with the University of Milano.

protocol was not accessible and could not be modified (e.g., The Coordinator, cf. Winograd, 1986; Winograd and Flores, 1986; Flores et al., 1988), or because the facilities for changing the protocol did not support actors in modifying the protocol (e.g., DOMINO, cf. Kreifelts et al., 1991a; Kreifelts et al., 1991b).

In response to these initial experiences, a number of research projects have attempted to make coordination facilities flexible to actors, e.g., Egret (Johnson, 1992), ConversationBuilder (Kaplan et al., 1992; Bogia et al., 1993; Bogia et al., 1996), and OVAL (Malone et al., 1992; Malone et al., 1995).

While closely related to these and other CSCW research efforts, our research has taken a somewhat different approach in that we have aimed at developing a computational notation which, on one hand, is sufficiently general to facilitate the construction of computer-based coordination mechanisms for any cooperative work arrangement and which, on the other hand, supports the cooperating actors themselves in constructing mechanisms which are both malleable and linkable. This notation — named Ariadne² — exists in the form of an abstract formal specification which has been partially implemented as a ‘concept demonstrator’ (Simone and Schmidt, 1994; Simone et al., 1995a).

The development of the general notation for constructing coordination mechanisms involves a range of research activities. Firstly, we need to understand how cooperating actors devise and use coordinative constructs such as coordinative protocols and workflows and how such constructs are supported by artifacts. A series of focused, in-depth field studies of coordinative practices in real-world cooperative settings have therefore been undertaken with the specific objective of investigating how the distributed activities of cooperative work arrangements are articulated and, in particular, how prescribed procedures and artifacts are devised, appropriated, and used for these purposes. On the basis of these and other investigations, we have developed a general conceptual framework which provides a set of categories and models which, in turn, form the basis of Ariadne. The whole process has been highly iterative, of course, in that the development of Ariadne has raised questions to the underlying conceptual framework which, again, has generated issues to be investigated further in the field studies. On the other hand, the findings from the field studies have been used to put the conceptual framework and the notation to test; as a result the framework and the notation have been modified and refined repeatedly.

While the research reported in this paper has been underway for several years and has been reflected in a number of papers and reports, the purpose of the present paper is to present a systematic exposition of the conceptual framework that has

² In Greek mythology, Ariadne was the daughter of Minos, the king of Crete. She fell in love with Theseus and helped him slay the Minotaur, the monster of the Labyrinth. She did that by giving Theseus a thread that would help him to find his way in and out of the Labyrinth.

been developed in the course of this research and which serves as the foundation for the Ariadne notation.

The argumentation underlying this exposition is, unfortunately, rather intricate. Thus, in order to help the reader to navigate the labyrinthine exposition, its flow is punctuated by a number of propositions which highlight and offer a condensed expression of the crucial points of the argumentation.

1. The issue of articulation work

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 typically 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., administrative database systems), the issue of *supporting the articulation of cooperative work by means of such systems* was not 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.³

Consider, for example, the booking system of an airline. It 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. Taken together, the seating arrangements and the database model constitute what we call *the common field of work* of the booking agents. Thus, the operators of the booking agencies cooperate by changing the state of the field of work, *in casu*, by reserving seats. Apart from providing a rudimentary access control facility, the booking system does not in any way support the coordination and integration of the interdependent activities of the operators. In this case, however, the field of work can be treated as a system of discrete and extremely simple (binary) state changes. A seat can only be assigned to one person at a time, there are no interactions between processes, and the state of any given seat can be ascertained unambiguously. Accordingly, even though a booking system does not support articulation work, it is seemingly quite sufficient for the job.

³ 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).

However, some if not most cooperative work arrangements in modern industrial society are faced with a far more complex field of work. The field of work may have a multitude of possible states, the state of the field of work may be ambiguous, and state changes may be interdependent in numerous ways, may occur intermittently and concurrently, and may be dynamic and unpredictable. Since members of such ensembles therefore are faced with complex interdependencies between individual activities, they cannot rely on accomplishing their individual and yet interdependent tasks merely by changing the state of the field of work. They must articulate their distributed activities in other ways. With CSCW, these issues become crucial. In fact, CSCW can be conceived of as a field devoted to exploring how computer-based systems can enhance the ability of cooperating actors in articulating their activities (cf., e.g., Schmidt and Bannon, 1992; Fitzpatrick et al., 1995).

In order to be able to conceptualize and specify the support requirements of cooperative work, we make an analytical distinction between ‘cooperative work’ and ‘articulation work.’ This distinction is fundamental to the approach presented here — and, in our opinion, to CSCW in general (Schmidt, 1994). *Cooperative work* is constituted by the interdependence of multiple actors who, in their individual activities, in changing the state of their individual field of work, also change the state of the field of work of others and who thus interact through changing the state of a common field of work. Since it involves multiple actors, cooperative work is inherently distributed, not only in the usual sense that activities are distributed in time and space, but also — and more importantly — in the sense that actors are semi-autonomous in terms of the different circumstances they are faced with in their work as well as in terms of their strategies, heuristics, perspectives, goals, motives, etc. (Schmidt, 1991a; Schmidt, 1991b). To deal with this source of confusion and disorder, individual and yet interdependent activities must be coordinated, scheduled, aligned, meshed, integrated, etc. — in short: articulated.⁴ That is, the orderly accomplishment of cooperative work requires what has been termed *articulation work* (Strauss et al., 1985; Gerson and Star, 1986; Strauss, 1988; Strauss, 1994).

Proposition 1. Cooperative work is constituted by the interdependence of multiple actors who interact through changing the state of a common field of work, whereas *articulation work* is constituted by the need to restrain the distributed nature of complexly interdependent activities.

The distinction between cooperative work and articulation work is *recursive*; that is, an established arrangement of articulating a cooperative effort may itself be subjected to a cooperative effort of re-arrangement which in turn also may need to

⁴ The word ‘articulate’ is used in the sense of ‘to put together by joints’.

be articulated, and so forth. To take a simple, and perhaps simplistic, example: At some point during a design meeting one of the participants interrupts the design discourse to change the agenda for the meeting. Following that, the participants discuss the proposal for some time, adopt it in an amended form, and resume the design discourse where they broke off. In this case, the established arrangement (the agenda) is treated as the field of work of another cooperative effort, namely that of rearranging the agenda. This recursion is, in principle, infinite. For instance, during the discussion about the proposed change to the agenda, someone may raise the issue of floor control by, say, proposing that nobody should be allowed to speak about the proposal more than once, which may in turn ignite yet another round of exchanges at another level of recursion. While this could go on forever, the infinite recursion is made finite and closed, to get the job done. The severe constraints under which work takes place in the real world dictate that such recursions are normally terminated well before they become frivolous.

Proposition 2. Articulation work is a *recursive* phenomenon in that the management of an established arrangement of articulating a cooperative effort may itself be conducted as a cooperative effort which, in turn, may also need to be articulated.

2. The complexity of articulation work

Cooperative work is, as noted above, inherently *distributed*. This distributed character of cooperative work varies, however, according to the complexity of the interdependence, that is, depending on factors such the distribution of activities in time and space, the number of participants in the cooperative ensemble, the structural complexity posed by the field of work (interactions, heterogeneity), the degree and scope of specialization among participants, the apperceptive uncertainties posed by the field of work and hence the variety of heuristics involved, and so on. The more distributed the activities of a given cooperative work arrangement, the more complex the articulation of the activities of that arrangement is likely to be.

With low degrees of complexity, the articulation of cooperative work can be achieved by means of the modes of interaction 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 our repertoire of intuitive interactional modalities that the *distributed* nature of cooperative work is not manifest. As demonstrated by the body of rich empirical studies of cooperative work within CSCW, actors tacitly monitor each other; they perform their activities in ways that support coworkers' awareness and understanding of their work; 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, and they mesh these interactional modalities dynamically and seamlessly (Harper et al., 1989; Heath and Luff, 1992; Harper and Hughes, 1993; Heath et al., 1995).

However, in the complex work settings that characterize modern industrial, service, and administrative organizations (hundreds or thousands of actors engaged in myriads of complexly interdependent activities), the task of articulating the interdependent and yet distributed activities is of an order of complexity where our everyday social and communication skills are far from sufficient.

Faced with a high degree of complexity of articulation work, cooperating actors typically use a special category of artifacts which, in the context of a set of conventions and procedures, stipulate and mediate articulation work and thereby are instrumental in reducing its complexity and in alleviating the need for ad hoc communication. Consider, for example, the case of the S4000 project.

The S4000 project. Foss Electric is a Danish manufacturing company that produces advanced equipment for analytical measurement of quality parameters of agricultural products, e.g., the compositional quality of milk in terms of fat content and the count of protein, lactose, somatic cells, bacteria, etc. At the time of the field study,⁵ the company was engaged in a large design project called S4000 which aimed at building a new instrument for analytical testing of raw milk. The S4000 project was the first project aiming at building an integrated instrument that would offer a range of functionalities that previously had been offered by a number of specialized instruments. In addition, as an innovation compared to previous models, the S4000 system would introduce measurements of new parameters in milk (e.g., urea and citric acid), and the performance was to be radically increased. The instrument would consist of approximately 8,000 components grouped into a number of functional units, such as cabinet, pipette unit, conveyer, flow-system, and measurement unit. Finally, the S4000 was the first Foss instrument to incorporate a personal computer (an Intel-based 486 PC) by means of which the configuration and operation of the instrument were to be controlled (through a Windows interface). Eventually, the first version of the software consisted of approximately 200,000 lines of source code. Altogether more than 50 people were involved in the S4000 project, which lasted approximately 30 months (for the first version).

The design team was faced with quite a challenge:

(1) The different subsystems, e.g., the software control system and the mechanical and chemical processes in the flow and measurement system, were intricately interdependent and might interact in unforeseen ways.

(2) The S4000 project introduced measurement of new parameters in raw milk for which new technologies had to be developed and mastered.

⁵ The field research of the S4000 project was done by Henrik Borstrøm, Peter Carstensen, and Carsten Sørensen.

(3) The different subsystems were developed concurrently and the requirements to be satisfied by each subsystem would therefore change as other subsystems were developed.

(4) Production facilities at the manufacturing plant were constantly changing as the use of existing machines was optimized and new machines and processes were introduced. Hence, the repertoire of manufacturing processes that the production function could offer to designers and that designers thus had to take into account in their decisions was continually changing.

(5) Because of its technological heterogeneity, the S4000 project involved a number of specialisms. The core design team consisted of designers from mechanical engineering, electronics, software, and chemistry. In addition, a handful of draught-persons and several persons from organizational entities such as production, model shop, marketing, quality assurance, quality control, service, and top management were involved to varying degrees at different stages in the course of the project.

All in all, the project was significantly more complex than previous projects at Foss.

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

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

However, the amount of detailed information that had to be communicated, aligned, negotiated, etc., required more robust measures. A number of procedures and artifacts were introduced to keep track of the state of affairs and to manage relations and dependencies among actors, tasks, and resources: an 'augmented' bill of materials that identified actors responsible for parts in order to support the coordination of mechanical design, process planning, and production in the construction of prototypes (Sørensen, 1994a); a CEDAC board (Cause and Effect Diagram with the Addition of Cards) for coordinating the diagnosis of faults between mechanical design and process planning (Sørensen, 1994b); and a product classification scheme supporting the distributed classification and retrieval of CAD models (Sørensen, 1994c). Some of these procedures and artifacts were invented for this project, some were redesigns of existing artifacts, and others were merely adopted.

The most dramatic measures were taken with respect to the software design process. In the early phases of the software strand of the S4000 project, the software designers felt that their overview of the state of the project was quite

defective and that they needed much greater coordination. As one of the software designers put it:

‘It has really been problematic that we did not have any guidelines and descriptions for how to produce and integrate our things. The individual designers are used to work on their own and have all the required information in their heads, and to organize the work as they wish to [...] When we started, we were only a few software designers. And suddenly — problems! And, oops, we were quite a few software designers and external consultants involved.’

At the height of the crisis the software design goals were almost abandoned. To overcome the crisis, the software designers developed a repertoire of procedures and artifacts to ensure the monitoring and control of the integration of software components and modules.

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

Moreover, the software design team devised and introduced other procedures and artifacts. Most importantly, a ‘bug report form’ with corresponding procedures for reporting, classifying, and correcting faults were introduced to ensure that bugs were properly registered, that corrected bugs were duly reported, and to make the allocation of responsibilities clear and visible to all members. As a complementary measure, copies of bug forms were collected in a publicly available repository in the form of a simple binder. (For further details, cf. Carstensen et al., 1995a).

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

The case of the S4000 project is particularly valuable because we here witness the introduction of specialized artifacts for coordination purposes in response to overwhelming problems encountered in coping with the complexities of articulating the distributed and interdependent activities of cooperative design under conditions that are typical for contemporary industry. However, while daunting to the participants, the complexity of the S4000 project is not exceptional. Such complexities are an everyday occurrence in modern industrial, service, and administrative settings.

Proposition 3. In cooperative work settings characterized by complex task interdependencies, the articulation of the distributed activities requires specialized artifacts which, in the context of a set of conventions and procedures, are instrumental in *reducing the complexity of articulation work* and in alleviating the need for *ad hoc* deliberation and negotiation.

Artifacts have been in use for coordination purposes in cooperative settings for centuries, of course — in the form of time tables, checklists, routing schemes, catalogues, classification schemes for large repositories, and so on. Now, given the infinite versatility of computer systems, it is our contention that such artifacts in the form of computational coordination mechanisms can provide a degree of perspicuity and flexibility to artifactually supported articulation work that was unthinkable with previous technologies, typically based on inscriptions on paper or cardboard. This opens up new prospects for moving the boundary of allocation of functionality between human and artifact with respect to articulation work 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 generalization, we call these artifacts and the concomitant procedures and conventions ‘coordination mechanisms.’⁶ In the following sections of this paper, we will expound this concept at length.

3. Coordination mechanisms: evidence and concept

The concept of coordination mechanisms has been developed as a generalization of phenomena described in different ways in numerous empirical investigations of the use of artifacts for coordination purposes in different work domains:

- standard operating procedures in administrative work (Zimmerman, 1966; Zimmerman, 1969b; Zimmerman, 1969a; Wynn, 1979; Suchman, 1983; Suchman and Wynn, 1984; Wynn, 1991);
- classification schemes for large repositories (Bowker and Star, 1991; Andersen, 1994; Sørensen, 1994c);
- time tables in urban transport (Heath and Luff, 1992);
- flight progress strips in air traffic control (Harper et al., 1989; Harper and Hughes, 1993);
- production control systems in manufacturing (Schmidt, 1994);

⁶ In earlier papers we used the term ‘mechanism of interaction’. The change of terminology does not imply any conceptual changes and is merely motivated by our experience that the term ‘mechanism of interaction’ can have unintended connotations.

- schedules in hospital work (Zerubavel, 1979; Egger and Wagner, 1993);
- planning tools for manufacturing design (Bucciarelli, 1988; Sørensen, 1994a; Carstensen et al., 1995a);
- fault correction procedures in engineering and software design (Carstensen, 1994; Pycock, 1994; Pycock and Sharrock, 1994; Sørensen, 1994b).

Consider, for example, the bug form as described in the study of the Foss Electric S4000 project:

The bug report form. The bug report form (see Figure 1) was a two page form (both sides of one sheet of paper). A new bug report was initiated and filled-in by anyone involved in testing the software, which could be software designers, other designers, quality assurance staff, or marketing people. The originator of the bug report also provided a preliminary description and diagnosis of the problem. A so-called ‘spec-team’, that is, a group of three software designers responsible for diagnosing and classifying all reported bugs, then determined the module which presumably was responsible for the fault and, by implication, identified the designer responsible for that module and therefore for correcting the bug. The ‘spec-team’ also specified the platform integration period by which the bug should be corrected, and classified the bug according to its perceived severity (as seen from a software reliability perspective). Finally, each designer was responsible for fixing the problems (handling ‘his’ or ‘her’ bugs) and reporting back to the Platform Master, i.e., the designer responsible for the next software module integration and verification period.

Initials:	Instrument:	Report no:	
Date:			<i>The tester</i>
Description:			<i>The spec-team</i>
Classification:			<i>The tester</i>
1) Catastrophic 2) Essential 3) Cosmetic			<i>The spec-team</i>
Involved modules:			
Responsible designer: Estimated time:			<i>The spec-team</i>
Date of change: Time spent: Tested date:			<i>The designer</i>
<input type="checkbox"/> Periodic error - presumed corrected			<i>correcting the bug</i>
Accepted by: Date:			
To be:			<i>The spec-team</i>
1) Rejected 2) Postponed 3) Accepted			
Software classification (1-5): ____			
Platform:			
Description of corrections:			<i>The designer</i>
Modified applications:			<i>correcting the bug</i>
Modified files:			

Figure 1. The bug report form (translated from Danish), with indications of which actor (role) is supposed to fill in which fields (Carstensen et al., 1995b).

As noted above, all bug forms were filed in a public repository ('the binder') which was organized according to the following categories: (1) non-corrected 'catastrophes', (2) non-corrected 'semi-serious' bugs, (3) non-corrected 'cosmetic' bugs, (4) postponed, (5) rejected, (6) corrected but not yet tested, and (7) corrected bugs. The forms collected in the binder were successively re-classified and re-filed according to decisions made by the spec-team, messages concerning specific bugs from the designers, results from the verification of reported corrections, etc.

Finally, a list of not-yet-fixed bugs (category 1, 2, and 3) was produced continually and accessed by the software designers. It gave them an indication of the state of the software system as a whole and supported them in being aware of design activities concerning modules for which they were not responsible but with which their own modules might interact.

Basically, five different roles were involved in coordinating the debugging activities in the S4000 project: the testers testing the software, the 'spec-team' diagnosing the reported bugs, the software designers correcting the diagnosed bugs, the 'platform master' verifying the corrected bugs, and the designer maintaining the bug form repository to keep track of the distributed debugging activities (see Figure 2).

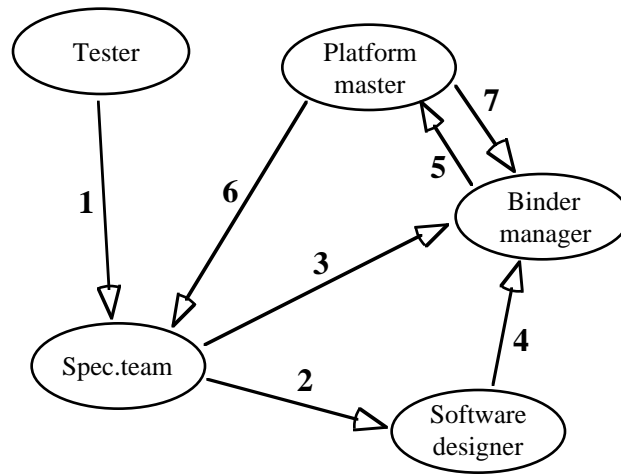


Figure 2. A schematic illustration of the roles and information flows in software testing in the S4000 project. The flows in the diagram indicate the intended flow according to the bug handling protocol. The protocol follows seven major steps: (1) a tester sends a form, describing a bug, to the 'spec-team;' (2) the 'spec-team' diagnoses the bug and sends the annotated form to a software designer; (3) a copy is send to the manager of the 'binder', i.e., the repository of bug forms; if a reported bug is rejected, the original is sent to the 'binder' manager; (4) the software designer reports on corrected bugs and sends the form to the 'binder' manager; (5) the 'binder' manager sends a stack of forms of bugs to be verified to the 'platform master'; (6) forms of bugs that cannot be verified are sent to the 'spec-team'; (7) verified forms are sent to the 'binder' manager. (Carstensen et al., 1995b).

As illustrated by the case of the bug report form, a coordination mechanism can be conceived of as constituted by two devices: On one hand we have a *coordinative protocol* in the form of a set of agreed-to procedures and conventions which, to competent members of the ensemble, stipulates the responsibilities of the different roles, the possible classifications of bugs, the intricate flow of forms, acknowledgments, reports of bugs corrected, etc. On the other hand we have the bug report form as *an artifact*, i.e., as a distinct and persistent symbolic construct, in which the protocol is imprinted and objectified.

Proposition 4. A coordination mechanism is a construct consisting of a *coordinative protocol* (an integrated set of procedures and conventions stipulating the articulation of interdependent distributed activities) on the one hand and on the other hand *an artifact* (a permanent symbolic construct) in which the protocol is objectified.

Coordination mechanisms are characterized by a specific and crucial relationship between protocol and artifact. In the next sections we will analyze this relationship by investigating the constituent parts in turn.

3.1. Coordination mechanisms: the protocol

While the notion of protocols that stipulate the articulation of cooperative work is crucial to the concept of coordination mechanisms, it is also contested. In a large body of sociological literature, the notion of pre-defined organizational constructs in general (formal structures, procedures, methods, plans) as determinants of action has been subjected to critical examination. Study after study have demonstrated, unambiguously and beyond any doubt, that the status of these formal organizational constructs in the actual course of work is problematic in that these constructs are impoverished idealizations when taken as representations of actually unfolding activities. In the words of Philip Selznick's classic summary of this line of sociological investigation:

'The formal administrative design can never adequately or fully reflect the concrete organization to which it refers, for the obvious reasons that no abstract plan or pattern can — or may, if it is to be useful — exhaustively describe an empirical totality. At the same time, that which is not included in the abstract design (as reflected, for example, in a staff-and-line organization chart) is vitally relevant to the maintenance and development of the formal system itself.' (Selznick, 1948, p. 25)

This conception of the status of formal constructs has been highly influential in that it, as observed by Egon Bittner in a now classic paper, has 'furnished the necessary theoretical argument for an entire field of sociological investigations by directing attention to a sphere of adaptive and cooperative manipulations, and to the tensions typically found in it.' (Bittner, 1965, p. 240)

The issue — for Bittner and for us — is: what is the status of these formal organizational constructs? The problem with the received tradition of critical studies of formal organizational constructs, however, is the almost ceremonial status it implicitly ascribes to these formal constructs and the ensuing dichotomy of the 'formal' and the 'informal,' the notional and the corporeal. The argument implies that members of the given organizational settings are somehow supposed to take formal constructs literally — as if constructs such as procedural formulations are *supposed* to be exhaustive specifications of how the work gets done.

In addressing this problem, Bittner makes some very cogent observations:

'While Selznick quite clearly assigns the formal schemes to the domain of sociological data, he does not explore the full range of consequences out of this decision. By retaining Weber's conception of them as normative idealizations, *Selznick avoids having to consider what the constructions of rational conduct mean to, and how they are used by, persons who have to live with them from day to day. It could be, however, that the rational schemes appear as unrealistic normative idealizations only when one considers them literally, i.e., without considering some tacit background assumptions that bureaucrats take for granted.*' (Bittner, 1965, p. 242 — emphasis added)

Bittner's methodological recommendation is quite pertinent to the issue of analyzing and designing coordination mechanisms. To be able to contribute constructively to the design of computational coordination mechanisms, we need to

understand not only ‘the tacit background assumptions’ that members take for granted and without which any formal construct would be merely a rhetorical statement but also ‘what the constructions of rational conduct mean to, and how they are used by, persons who have to live with them from day to day.’

In the course of the following exposition and discussion, it is important to keep in mind that we are not trying to address or solve the general problems of general sociological theory. We are not investigating human action in general, merely the means of articulating distributed and interdependent activities *in work settings*, that is, under conditions of severe constraints. Nor are we investigating the nature of tacit and implicit plans, rules, routines, habits, and so on in human social conduct in general but, more specifically and modestly, the role of *artificially imprinted protocols* in the articulation of cooperative work.

However, since it is not merely the status of coordinative protocols that is contested but the status of protocols in general, and since much of the evidence produced for this discussion accordingly does not specifically refer to protocols designed and used to support the articulation of cooperative work but to formal organizational constructs in general, we cannot, in our investigation, avoid to refer to and draw upon evidence of artificially imprinted protocols that are not used in the *articulation* of cooperative work. In doing that, our objective remains, however, to understand the more specific problem of the role of artificially imprinted protocols in the articulation of cooperative work. To make this explicit we will use the term ‘artificially imprinted protocol’ to denote any protocol, coordinative or otherwise, which is objectified in an artifact.

During the last 15 years or so, our understanding of how procedures and artificially imprinted protocols are used by actors in everyday work activities has been greatly enriched by a number of outstanding studies (e.g., Wynn, 1979; Suchman, 1983; Suchman and Wynn, 1984; Bucciarelli, 1988; Wynn, 1991). The general conclusion of these studies is that such procedures and artifacts serve as ‘maps’ (Suchman, 1987, p. 188 f.; Bucciarelli, 1988, p. 114). Consider, for instance, Suchman’s study of the accounting office (Suchman, 1983):

The accounting office. This office was responsible for the orderly payment of money due to outside organizations supplying goods and services to the organizational units in its charge. Orderly payment was documented through record-keeping, and accuracy was monitored by the auditing of invoices against records of requisition and receipt. According to the standard procedure, items on a given purchase order could be received and billed in separate installments over an extended period. Again, if all went smoothly, the items marked off on the receiving report from Shipping or Receiving would correspond to those on the invoice from the vendor. The purchase order, receiver, and invoice would be matched and audited. The payment for the items received would be recorded by margin notes on the purchase order, which would then be returned to the temporary file to wait for

the next shipment and billing. Only after all bills had been received and paid was the completed purchase order filed permanently in the paid file.

In the case presented and analyzed by Suchman, however, the record of what had happened was incomplete: The original purchase order was missing. A completed receiving document was found with eight items listed on it, all of which had been marked as received. The two invoices found in the paid file showed only two items as paid, however; there was no invoice or record of payment for the other items, yet the vendor reported that the transaction would be completed with payment of the past due invoice for only two of those items that seemingly had not yet been paid. The study then shows how the two actors, the accounting clerk and the auditing clerk, step by step solved the ‘mystery’: Of the invoice for one of the items, only page two was on file; page one was missing. It thus transpired that four other items were invoiced with this item and had already been paid.

This case shows convincingly that orderly records are not necessarily the result of some prescribed sequence of steps and that it may involve the practice of completing a record or pieces of it after the fact of actions taken: ‘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’ (Suchman, 1983, p. 326). Thus, precisely because it is a case of recovery from error, the case gives a vivid impression of the massive heuristic use of standard procedures even in a seemingly abnormal situation. The two actors are able to solve the abnormal problem because of their ‘knowledge of the accounts payable procedure’ (Suchman, 1983, p. 322). Standard procedures can thus be said to 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’ (Suchman, 1983, p. 327).

Taking this interpretation further, Suchman posits that a standard procedure serves as an extraneous and subservient referent for situated action:

‘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)

This interpretation is generalized in Suchman’s seminal book on *Plans and Situated Action* (1987):

‘plans are resources for situated action, but do not in any strong sense determine its course. While plans presuppose the embodied practices and changing circumstances of situated action, the efficiency of plans as representations comes precisely from the fact that they do not represent those practices and circumstances in all of their concrete detail’ (Suchman, 1987, p. 52).

Suchman's thesis that 'plans are resources for situated action' is of fundamental importance to CSCW systems design and has served us as a guiding principle in the development of the concept of coordination mechanisms, but it also leaves a number of unsettling questions unanswered: What is it that makes plans such as production schedules, office procedures, classification schemes, etc., useful in the first place? What makes them 'resources'? Furthermore, 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 there is to it?⁷ What, then, makes one procedure or form or schedule more useful than another for a certain purpose in a specific setting?

Later in the book, Suchman returns to these issues and suggests a rather apt metaphor for the role of artifactually imprinted protocols, namely that of a 'map':

'Just as it would seem absurd to claim that a map in some strong sense controlled the traveler's movements through the world, it is wrong to imagine plans as controlling actions. On the other hand, the question of how a map is produced for specific purposes, how in any actual instance it is interpreted *vis-à-vis* the world, and how its use is a resource for traversing the world, is a reasonable and productive one.' (Suchman, 1987, pp. 188 f.)

While the same irksome questions arise here as well, the 'map' analogy is a fitting condensation of the role of artifactually imprinted protocols that have been described in a number of studies. In Suchman's study of the accounting office, for example, the standard operating procedures were found to be 'formulated in the interest of what things should come to, and not necessarily how they should arrive there.' They were used as a general reference for orientation purposes, not as a prescribed sequence of actions to be taken.

However, other studies lead to quite different conclusions as to how artifactually imprinted protocols are used by actors in everyday work activities.

Checklists. Firstly, consider the relatively simple case of the 'normal checklist.' The checklist is an artifactually imprinted protocol that has been deliberately and carefully designed to *reduce* local control in safety-critical environments. More specifically, a checklist is used to organize tasks whenever it is essential that a set of actions *all* be performed, typically where it is essential that the actions of the performance also be taken in a particular order, to ensure a high level of operational safety. For example, the normal aircraft flight-deck checklist indicates a set of different tasks the pilot must perform or verify during all flight segments in order to configure the aircraft and prepare the flight crew for certain 'macro-tasks' such as ENGINE START, TAXI, TAKEOFF, APPROACH, LANDING, etc. For each one of these macro-tasks there are several 'items' to be accomplished and verified by the crew (Degani and Wiener, 1990).

⁷ 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? Does that mean that the less specific the better? Suchman probably does not intend to imply that.

In his analysis of the checklist, Don Norman observes that ‘The fact that the preparation of the list is done prior to the action has an important impact upon performance because it allows the cognitive effort to be distributed across time and people’ (1991, p. 21). This preparatory task — which Hutchins aptly has dubbed ‘precomputation’ — can be done when more convenient, e.g., when there is no time pressure and no safety and security risk, and by another actor, e.g., by a specialist. ‘In fact,’ Norman observes, ‘precomputation can take place years before the actual event and one precomputation can serve many applications’ (1991, p. 21). The concept of a precomputation of essential aspects of a task is crucial to understanding the role of artifactually imprinted protocols: The flight-deck checklist, for instance, provides a precomputed selection of safety-critical tasks, which all need to be performed at the particular flight segment as well as a precomputed sequence for their execution.

Now, the protocol of the flight-deck checklist does not stipulate the *articulation* of cooperative activities but the activities themselves and it is therefore not a coordinative protocol. For a case of the use of artifactually imprinted protocols that stipulate the articulation of cooperative activities, consider the *kanban* system.

The kanban system. In 1990, Bjarne Kaavé conducted a study of cooperative production control in a manufacturing company we can call Repro Equipment.⁸ The company manufactured specialized optical appliances and covered about 50% of the world market for this category of equipment. At the time of the study, the company produced about 6,000 units a year in fifteen models, each in seven variants.

A manufacturing operation, like the one at Repro Equipment, involves multitude discrete parts and processes that are complexly interdependent: Each product consists of many component parts, in some cases tens or hundreds of thousands of components, and their production may require a number of different processes in a specific sequence. Different processes, such as cutting, bending, welding, etc., typically require specialized tools and skills which are distributed at different workstations and require hugely different set-up times. This is compounded by the fact that, at any given time, a large number of products and their components coexist in the production process at different stages of completion which means that different parts for the same or for different products compete for the same workstations. Thus, in the words of Harrington (1984, p. 4), manufacturing can be conceived of as ‘an indivisible, monolithic activity, incredibly diverse and complex in its fine detail. The many parts are inextricably interdependent and interconnected.’ Accordingly, for a manufacturing enterprise to be able to adapt to changing conditions, the entire enterprise must react ‘simultaneously and cooperatively’ (Harrington, 1979, p. 35).

⁸ This analysis is based on Bjarne Kaavé’s findings as reported in his thesis (Kaavé, 1990) as well as in several joint analysis sessions with one of the present authors.

To deal with this complexity, Repro Equipment had introduced a *kanban* system to coordinate processes in the production of cabinets. *Kanban* is a Japanese word for ‘card’ or more literally ‘visible record’ (Schonberger, 1982, p. 219) and it is now in widespread use in manufacturing to denote a just-in-time production control system where a set of cards acts as the carrier of information about the state of affairs *as well as* production orders conveying instructions to initiate certain activities. The basic idea is that loosely interdependent production processes can be coordinated by exchanging cards between processes. When a new batch of parts or sub-assemblies has been produced and the batch is to be transported ‘down-stream’ from the present work station to the station where it is to be used, for instance, as components for a sub-assembly, a specific card is attached to the container used for the transportation. When the operator at the work station down-stream has processed this batch of parts, the accompanying card is sent back to the operator who produces these parts. To the operator, receiving the card means that he or she has now been issued a production order.

The basic set of rules of a *kanban* protocol is as follows (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.

Setting up a *kanban* system requires a careful configuration of the number of containers per part number and the quantity per container. This configuration, in effect, amounts to a precomputation of tasks in terms of batch size per part number, task allocation in terms of work stations for different part numbers, and task sequences.

However, a *kanban* system is not adequate for coordinating manufacturing operations faced with severe demands on flexibility of volume: a *kanban* system can only handle small deviations in the demand for the end product (Schonberger, 1982, p. 227; Monden, 1983). Accordingly, since Repro Equipment was faced with extreme differences and fluctuations in demand for different models and variants, operators recurrently experienced that the configuration of the *kanban* system (the number of containers per part number and the quantity per container) was inadequate. For instance, in a situation where a particular part number that was only used for a special product variant had all been used, the protocol would automatically generate a production order for this part number, although the part number in question probably would not be needed in months and would thereby absorb production facilities that would be needed for other, more pressing orders.

In such situations, where the *kanban* system is ‘beyond its bounds’ (Roth and Woods, 1989), operators at Repro Equipment would tamper with the *kanban* protocol. For example, having heard of a new rush order from the girl in the order

office, the fork lift operator might put the card for a rarely used part for another model in his back pocket or leave it on the fork-lift truck for a while. Similarly, in order to rush an order, operators would occasionally order a new batch of parts for this order *before* the container had actually been emptied and the card had been released, or they would deviate from batch sizes as specified on the card, etc.

It is crucial to notice that instead of abandoning the *kanban* system altogether, or at least temporarily, the operators *changed the configuration* of the system. That is, when an operator pocketed a *kanban*, he or she was *modifying* the protocol, not switching it off, and when the card was put back in circulation (or released belatedly), the default configuration was in force again. The reason for this is that the *kanban* system incorporates (implicitly, in the configuration of the system) a precomputed model of crucial interdependencies of the manufacturing process (routing scheme, set-up-times, etc.). Thus, even though the *kanban* system at Repro was often used in situations where it was ‘beyond its bounds,’ it was not discarded but merely modified locally and temporarily according to the requirements of the situation.

In order to be usable in a setting like Repro Equipment, the *kanban* system had to be managed (monitored, adapted, modified) continually. This was facilitated by the formation of a network of clerks, planners, operators, fork-lift drivers, and foremen in various functions such as purchasing, sales, production, shipping, etc., who kept each other informed about the state of affairs to be able to control the flow of parts. A member of this network would for example explore the state of affairs ‘up-stream’ 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 was subsumed under a very direct, intelligent, and informal cooperative arrangement. The cooperative ensemble ‘appropriated’ the *kanban* system in order to increase its flexibility. They took over control of the system and controlled production far more closely and effectively than warranted by the design of the *kanban* system.

Coming back to the issue of the status of formal organizational constructs in cooperative work, the *kanban* system illuminates several important points.

Suchman’s contention that the function of abstract representations such as plans ‘is not to serve as specifications for the local interactions, but rather to orient or position us in a way that will allow us, through local interactions, to exploit some contingencies of our environment, and to avoid others’ (Suchman, 1987, p. 188) is not correct as far as the *kanban* system is concerned. When an operator receives a card, he or she will produce the batch as specified by the card, in accordance with the general rules of the protocol, *without actively searching for reasons not to do so* and without deliberating or negotiating whether to do so or not.

In their individual activities, actors rely on the *kanban* system to issue valid and sensible production orders, unless they have strong reasons to believe that its

unmitigated execution in the particular situation at hand will have undesirable results. Even then, they do not discard the system but alter its behavior by reconfiguring it, after which the system is allowed to ‘switch back’ to the default configuration. That is, in the case of the *kanban* system:

- actors coordinate their distributed activities by *executing* the *kanban* protocol — *unless* they have strong reasons to act otherwise;
- when actors have reasons to doubt the rationality of executing a production order issued by the system, they temporarily reconfigure the system, i.e., respecify the protocol, by withholding cards or introducing false cards;
- by reconfiguring the system, actors do not discard the system but alter its behavior temporarily, upon which the system is allowed to ‘switch back’ to its default configuration.

The *kanban* system thus determines action in a far stronger sense than the map of a traveler determines the traveler’s movements (Suchman, 1987, p. 188 f.; Bucciarelli, 1988, p. 114). In the *kanban* case the protocol conveys a *specific* stipulation in the form of a production order to the particular actor instructing the actor, under the conditions of social accountability, to take the particular actions specified by the card according to the general rules of interpretation laid down in the protocol. It is thus more like a *script* than a *map*. In fact, the *kanban* system works well even though it does not provide a ‘map’ in the form of an overview of interdependencies among processes.

The point is that the *kanban* protocol under normal conditions of operation relieves actors of the otherwise forbidding task of computing myriad — partly interdependent, partly competing — production orders and negotiating their priority. They can, for all practical purposes, rely on the precomputed protocol to issue valid production orders; they take it for granted. Thus, for an actor in Repro Equipment to question the rationality of the protocol at every step in every situation would be an utter waste of effort, and it does not happen.

As a generalization, we find that a protocol stipulates the articulation of distributed activities by conveying affordances and constraints to the individual actor which the actor, as a competent member of the particular ensemble, can apply without further contemplation and deliberation unless he or she, again as a competent member, has accountable reasons not to do so. That is, actors deviate from the stipulations of the protocol if and when they have compelling reasons to do so, and only then.⁹

⁹ ‘Even the simple checklist reduces the semantic distance for its users. Lacking the checklist, the novice must discover the steps that need to be done and an order in which they can be applied. With the checklist, the task is transformed: reading and following instructions take the place of procedural reasoning.’ (Norman and Hutchins, 1988, p. 15)

Proposition 5. A coordinative protocol is a resource for situated action in that it reduces the complexity of articulating cooperative work by providing a *precomputation of task interdependencies* which actors, for all practical purposes, can rely on to *reduce the space of possibilities* by identifying a valid and yet limited set of options for coordinative action in any given situation.

As demonstrated by the conflicting findings from different cases, artifactually imprinted protocols, such as plans, conventions, procedures, and so forth, play different roles in cooperative work. They may, on one hand, play the ‘weak’ role of the ‘map’ of the traveler by providing a codified set of functional requirements which provides a general heuristic framework for distributed decision making. On the other hand, they may play the ‘strong’ role of a ‘script’ that offers a ‘precomputation’ of interdependencies among activities (options, sequential constraints, temporal constraints, etc.) which, for each step, provides instructions to actors of possible or required next steps. Which role is appropriate naturally depends on the extent to which it is possible to identify, analyze, and model interdependencies in advance.

Moreover, the role of a particular protocol may vary according to the situation. Thus, in a situation where a standard operating procedure does not apply, the procedure may merely serve in its weak default capacity as a vehicle of conveying heuristics (as, for instance, in the accounting office). In other cases, however, such as the *kanban* case, the role of the protocol does not vary in the face of contingencies; rather, because of the complexity of the interdependencies of discrete parts production, the *kanban* protocol was not discarded, suspended, nor ‘weakened’ but temporarily respecified (reconfigured) by operators to accommodate the passing disturbance.

Proposition 6. *The role of coordinative protocols varies* from case to case and from situation to situation, according to the fitness and expressive power of the precomputation of interdependencies as represented by the protocol, from weak stipulations, as exemplified by ‘a map,’ to strong stipulations, exemplified by ‘a script.’

However, whether weak or strong, a protocol only conveys stipulations within a certain social context, within a certain community, in which it has a (more or less) certain and agreed-to meaning and it only does so under conditions of social accountability.

Moreover, as pointed out by Suchman with regard to office procedures, protocols are characterized by ‘the inherent and necessary under-specification of procedures with respect to the circumstances of particular cases’ (Suchman, 1982, p. 411). Furthermore, Suchman observes, ‘the vagueness of plans is not a fault,

but is ideally suited to the fact that the detail of intent and action must be contingent on the circumstantial and interactional particulars of actual situations' (Suchman, 1987, pp. 185 f.). However, the degree of vagueness of specific plans is itself contingent:

'While plans *can* be elaborated indefinitely, they elaborate actions just to the level that elaboration is useful; they are vague with respect to the details of action precisely at the level at which it makes sense to forego abstract representation, and rely on the availability of a particular embodied response.' (Suchman, 1987, p. 188)

Thus, it is not only that a protocol, as a linguistic construct (Suchman, 1987, p. 186), is inherently vague compared to the rich details of the actually unfolding activities of the cooperative work arrangement in which it is applied, nor is it only that a protocol is inherently decontextualized, but a protocol is deliberately under-specified with respect to (a) factors that are immaterial for the purpose of the given protocol or (b) factors that can more efficiently and effectively be left unspecified, typically until a later stage. The protocol must, to use the apt phrase of Bowker and Star, be defined at 'an appropriate level of ambiguity' (Bowker and Star, 1991, p. 77).

Proposition 7. As a preconceived plan for the articulation of the distributed activities of a specific cooperative work arrangement, *a coordinative protocol is inexorably under-specified* in the sense that the *nominal* preconception cannot encompass and denote the infinite multiplicity of *actual* circumstances and occurrences unfolding during its situated enactment.

Thus, whether a protocol is weak or strong, its execution involves an unavoidable aspect of situated interpretation and improvisation — which, nonetheless, as in the case of protocols used as scripts, may be inconsequential to competent members.

On the other hand, whether weak or strong, the protocol will, inevitably, encounter situations where it is beyond its bounds, its inherent vagueness and appropriate ambiguity notwithstanding. This is eloquently illustrated by the case of the kanban system. Similarly, the software designers intermittently experienced situations where the bug form protocol they had devised and adopted did not seem to provide adequate stipulations and where the execution of the bug form protocol was modified accordingly. For example, testers would occasionally inform a software designer directly of a detected bug, without filling-in a bug report form and initiating a new instance of the protocol.

Proposition 8. Whether weak or strong, *a coordinative protocol will, inevitably, encounter situations where it is beyond its bounds* and where actors therefore must deviate from or circumvent the execution of the protocol.

Let us now turn to the role of the artifact in coordination mechanisms.

3.2. Coordination mechanisms: the artifact

The role of the artifact in coordination mechanisms is, fundamentally, to objectify and give permanence to the protocol for which it stands proxy. The artifact conveys the stipulations of the protocol in a situation-independent manner. So far, the artifact is conceived of merely as a written record of the protocol, e.g., as a standard operating procedure.

While written language, as observed by Jack Goody, ‘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 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. In the words of Stinchcombe, ‘Written systems can provide a larger number of people with the same information at one time’ and 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.). 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’ (Goody, 1986, p. 6). They are, for all practical purposes, unceasingly publicly accessible.

Proposition 9. The role of the artifact in a coordination mechanism is fundamentally to *objectify and give permanence to the coordinative protocol* so that its stipulations are unceasingly publicly accessible.

Consider, for example, a standard operating procedure or a checklist. The state of the artifact is completely static irrespective of the state of the execution of the protocol it prescribes. Even when the artifactually imprinted protocol is used as a script (actors are following the instructions of the procedure or the items of the checklist step by step), it is entirely up to the actor to produce and maintain the required dynamic representation of the state of the protocol with respect to the unfolding cooperative activities.

In the case of the bug report, however, *the state of the artifact changes* according to the changing state of the protocol. Firstly, the form is transferred from one actor to another and this *change of location* of the artifact in itself conveys, to the recipient, the stipulations of the protocol in a specified form, that is, the change of

location transfers to the particular actor the specific responsibility of taking such actions on this particular bug that are appropriate according to the agreed-to protocol and other taken-for-granted conventions. Secondly, at each step in the execution of the protocol, *the form is annotated* and the thereby updated form retains and conveys this change to the state of the protocol to the other actors — the state of each reported bug is thus reflected in the successive inscriptions on the form made by different actors. That is, a change to the state of the protocol induced by one actor (a tester reporting a bug, for example) is conveyed to other actors by means of a visible and durable change to the artifact. In so far, the artifact can be said to provide a ‘shared space,’ a space with a particular structure that reflects salient features of the protocol. Furthermore, this change is propagated within the ensemble according to the stipulations of the protocol, and the state of the total population of reported bugs is publicly visible in the public repository of bug forms (‘the binder’).

Similarly, in the case of the *kanban* mechanism, the artifact mediates articulation work in the sense that the change of location of a card, that is, the fact that it is transferred ‘up-stream’ from one actor to another, is equivalent to the arrival of a production order at that work station. However, as opposed to the bug report, the inscription on the *kanban* card is not changed and the state of the *kanban* protocol is thus not reflected in any particular card. Hence, state changes to the protocol under execution can not be inferred from the inscription on the cards, only from their location.

In these cases, the artifact not only stipulates articulation work (like a standard operating procedure) but *mediates* articulation work as well in the sense that the artifact acts as an intermediary between actors that conveys information about state changes to the protocol under execution.¹⁰ By serving the dual function of stipulating and mediating articulation work, the artifact is instrumental in reducing the complexity of articulating a vast number of interdependent and yet distributed and perhaps concurrently performed activities.

Proposition 10. The artifact of a coordination mechanism may, in some form and at a particular level of granularity, *dynamically represent the state of the execution of the protocol* and may thereby, among actors, mediate information about state changes to the protocol as it is being executed.

¹⁰ Edwin Hutchins has a related but not identical analysis of artifactually imprinted protocols. In his discussion, Hutchins suggests the term ‘mediating structures’ for artifacts that are not part of the field of work (as tools are) and yet are instrumental in reducing the complexity of work by providing some kinds of constraints to the conduct of the actor (Hutchins, 1986, p. 47). For Hutchins, the artifact or structure serves as an intermediary between an actor planning or defining the protocol for an activity and the actor performing the activity, whereas we, for our purposes, reserve the term ‘mediate’ to denote an artifact serving as an intermediary of *horizontal* propagation of state changes to the protocol, i.e., *within* the cooperative work arrangement at hand. In other words, coordination mechanisms can be conceived of as a special case of ‘mediating structures’, namely artifactually embodied ‘mediating structures’ that are used to *constrain the articulation of distributed activities* in cooperative work settings.

By virtue of the artifact's mediation of the changing state of the protocol between actors, the coordination mechanism not only conveys the general stipulations of the protocol but *specifies the stipulations* in the sense that the individual actor is instructed that it is he or she that has to take this or that specific action at this particular point in time. In other words, by representing and conveying the changing state of the protocol, the artifact also mediates the transition from 'nominal' to 'actual' in the enactment of the protocol.

Proposition 11. By mediating the changing state of the protocol, the artifact of a coordination mechanism *specifies* the general stipulations of the protocol.

To provide permanence to the coordinative protocol and serve as a mediator, the artifact upon which the protocol is imprinted must be *distinct from* the field of work (Proposition 1). An artifact may, of course, be subjected to all sorts of unforeseen use and an artifact that is involved in the transformation processes of the work may at the same time be used for coordinative purposes, perhaps to support a coordinative protocol. For example, actors writing a joint report may have adopted a convention according to which their coordinative interactions (comments to the evolving text as well as records of responsibilities, for example) be conveyed in and through the text of the report. That is perfectly feasible, and may indeed be very effective, but what happens, for instance, to the records of responsibilities and schedules when the text of the report is changed, for instance reorganized? They may have vanished as 'the snows of yesteryear,' to abuse the words of François Villon. That is to say, artifacts that are part of the field of work or coupled to the state of the field of work may be unreliable and treacherous as a material carrier of a coordination mechanism.

Proposition 12. The artifact of a coordination mechanism is *distinct from* the field of work in the sense that changes to the state of the field of work are not automatically reflected in the state of the artifact and, conversely, changes to the state of the artifact are not automatically reflected in the state of the field of work.

Moreover, due to its mediating role with respect to the state of the execution of the protocol, the artifact may support the development and maintenance of mutual awareness among the actors within the cooperating ensemble. By reflecting the state of the execution of the protocol, the artifact may convey information about occurrences within the ensemble from which actors can make inferences about likely or possible problems and develop an overview of the state of the protocol in its totality. This potential is clearly illustrated by the case of software testing, especially due to the successive inscriptions on the bug forms and the systematic assembly of (copies of) bug forms in 'the binder.' The *kanban* system, on the other hand, does not provide a facility for obtaining such an overview. In fact, the

information conveyed by the transfer of cards up-stream is drastically filtered and distorted by the successive translations from card to card. The only interface to the state of the protocol across the total population of cards in circulation is the (ever changing) *location* of the myriad cards in the distributed manufacturing system. That is, the *kanban* system does not provide facilities allowing actors to develop and maintain a mutual awareness so as to, for instance, anticipate disturbances and obtain an overview of the situation within the cooperative ensemble at large; they are, so to speak, enveloped by an overwhelming and inscrutable quasi-automatic coordination mechanism. — In fact, in the *kanban* system, changes to the state of the artifact are strongly coupled to state changes in the field of work. Information only propagates ‘up-stream’ as parts are used down-stream: the speed and pattern of propagation of information are thus restricted by the rate and pattern of changes to the field of work at large. The *kanban* system does allow operators to control the execution of the protocol, however, since that control is ultimately in the hands of the operators: it is the operator who has used the parts in a particular container who takes the card and sends it up-stream; it is the truck driver who delivers it; it is the operator further up-stream who receives the card and decides to act on it. That is, due to the operators’ control of the execution of the *kanban* protocol, the direct coupling of the *kanban* system to the field of work can be severed whenever they deem it appropriate to exercise that control.

An artifact is, of course, more than a permanent symbolic construct; it has a specific material format which, in itself, is of importance to its use. For example, consider the simple checklist again. The checklist can be conceived of as an artifactually imprinted protocol that has been deliberately and carefully designed to reduce local control, typically in safety-critical environments. The use of the checklist requires the actor to employ a strategy for sequential execution which permits him or her to ensure that the steps are done in the correct order and that each step is done once and only once. The material format of the checklist as an artifact may be of assistance to the actor in ensuring this:

‘The fixed linear structure of the checklist permits the user to accomplish this by simply keeping track of an index that indicates the first unexecuted (or last executed) item. Real checklists often provide additional features to aid in the maintenance of this index: boxes to tick when steps are completed, a window that moves across the checklist, etc.’ (Hutchins, 1986, pp. 47 f.; cf. also Norman and Hutchins, 1988, p. 9)

In a similar vein, Jack Goody, in a discussion of the specific affordances provided by the *material format* of written text, observes that writing introduces certain spatio-graphic devices such as lists, tables, matrices by means of which linguistic items can be organized in abstraction from the context of the sentence (Goody, 1987) and points out that the spatio-graphic format of an artifact can stipulate behavior by reminding an actor of items to do and directing attention to missing items: ‘The table abhors a vacuum’ (Goody, 1987, p. 276). This is, again, eloquently illustrated in the case of the bug form where the bug report form

provides a set of fields which match crucial points of the bug handling protocol and which are to be filled in by the different actors in the course of the bug's life (cf. Figure 1).

Proposition 13. By reflecting salient features of the protocol, the *material format* of the artifact conveys coordinative stipulations and may provide a 'shared space,' structured accordingly, for mediating changes to the state of the protocol in compliance with the protocol.

By way of concluding this discussion, it is important to keep in mind that an artifact only conveys stipulations within a certain social context, within a certain community, in which the protocol and any change to the state of the protocol have a (more or less) certain and agreed-to meaning and that it only does so under conditions of social accountability. The point we want to make here, however, is that the specific structural and behavioral properties of the artifact (its material format as well as its protocol, if such have been incorporated in the artifact) are formed to serve the purpose of conveying specific stipulations within this particular context by constraining and forcing the actors' behavior.

We can summarize our analysis of the constituent parts of the protocol-cum-artifact dyad, by attempting a formal definition of coordination mechanisms:

Proposition 14. A *coordination mechanism* is a specific organizational construct, consisting of a *coordinative protocol* imprinted upon a *distinct artifact*, which, in the context of a certain cooperative work arrangement, *stipulates* and *mediates* the articulation of cooperative work so as to *reduce the complexity of articulation work* of that arrangement.

3.3. Coordination mechanisms: alignment

Consider, again, the case of software testing in the S4000 project. As observed previously, in addition to the bug report mechanism, the software designers introduced and used a variety of protocol-cum-artifact dyads to handle the complexity of coordinating the distributed activities of software testing, each of them devised to serve specific purposes in the setting. However, the different protocols intersect and must therefore be aligned somehow by the actors.

For example, a project schedule in the form of a spreadsheet was used to capture and display the relationships between actors, responsibilities, tasks, and schedules. In handling bug reports, participants would consult the project schedule to obtain information about who would be responsible, as 'platform master,' for verifying the corrected bug; this was indicated in the bug report form by the number of the platform period. Similarly, an integration period number inscribed in the bug report form also indicated a deadline for the correction task to be finished. Neither the

name of the platform master nor the deadline were explicitly stated but could be derived from the project schedule, a spreadsheet where the name of the platform master and the date for that integration period number would be inscribed, at some point. That is to say, from the point of view of the involved coordination mechanisms, one coordination mechanism (the bug report mechanism) subscribed to the specification of a role and a date to be provided by another mechanism (the project schedule). Thus, an array of multiple protocols-cum-artifacts that intersect at various points makes it possible to instantiate a particular protocol (e.g., a particular bug report) while it still not completely specified; the missing specifications can be ‘filled-in’ later by consulting another artifact. Thus actors do not need to specify explicitly what can be inferred from other mechanisms at some point in time.

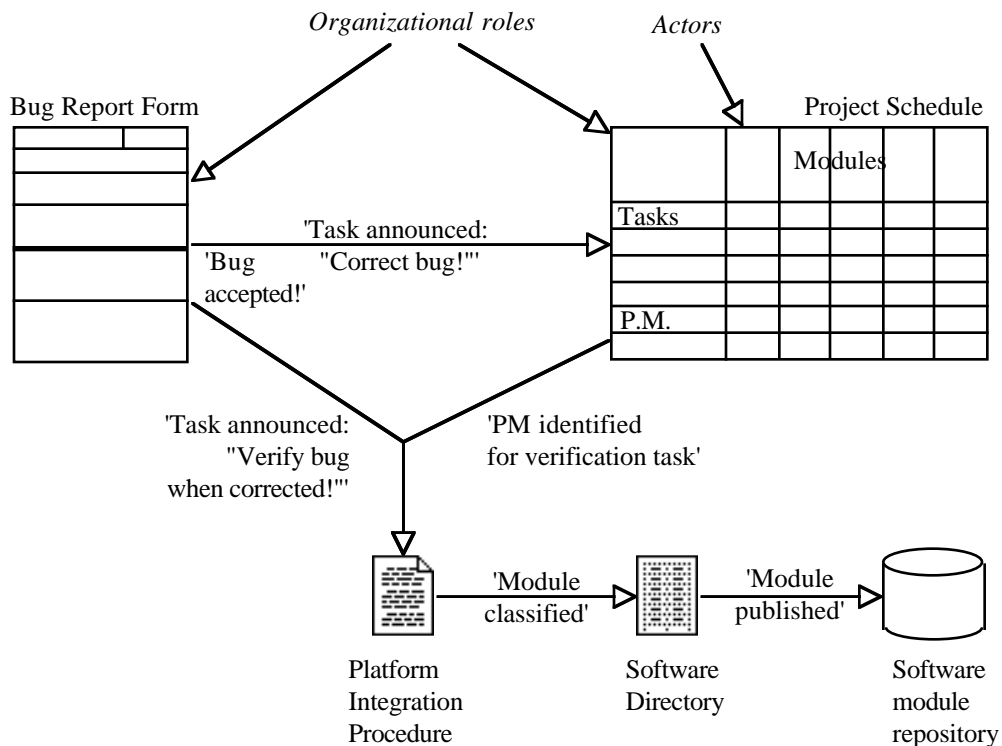


Figure 3. The interacting coordination mechanisms of the software testing case. ('P.M.' denotes the Platform Master, i.e., the actor in charge of the integration of modules and verification of corrections at the end of the current platform period).

The case also demonstrates a more active form of alignment between protocols, namely in the form of one protocol inciting the execution of another when a certain condition occurs (see Figure 3). For example, when a reported bug was accepted as a bug, a new task was announced and inscribed in the project schedule. That is, the change of the state of one mechanism (the bug report) instructed actors to make certain inscriptions on the artifact of another mechanism (the project schedule). Similarly, when a bug was reported to have been corrected yet another task were to be announced, namely the task of verifying the correction.

In the case of software testing, the continual alignment of multiple, specialized mechanisms seemed to be seamless and achieved effortlessly. This can largely be attributed to the fact that all designers would assume all roles, simultaneously or in turn. They could thereby develop and maintain a high degree of awareness of the intersections of the many mechanisms involved. In a larger or more complex arrangement this would not be as easily achieved and the alignment of the involved coordination mechanisms may thus require more effort and be less seamless.

Finally, while the array of protocol-cum-artifact dyads intersect at various points and therefore need to be aligned by the actors in the course of their work, they are only loosely coupled. Each mechanism addresses a very narrow set of coordinative activities. Because of that, each coordination mechanism can be constructed relatively independently of the others. Thus, in designing a mechanism for a specific purpose, one does not need to have an overview of the totality of work processes in the setting at large. In fact, because coordination mechanisms can be designed and introduced to serve limited purposes, the array of coordination mechanisms of the work arrangement at large can be designed and maintained in a distributed and bottom-up manner. The concept of coordination mechanisms thus suggests an approach to the design of workflow systems which, in line with Davenport's suggestions (Davenport, 1993), may result in composite workflow systems that are far less brittle in the face of the vicissitudes of contemporary business environments than workflow systems often seem to be.

Hence,

Proposition 15. Coordination mechanisms are *specialized constructs* that are devised to support certain aspects of the articulation of a specific category of distributed activities within a particular cooperative work arrangement and the use of a coordination mechanism may therefore require that it is *aligned with other mechanisms* devoted to different aspects of the articulation of those activities or to related activities.

4. Computational coordination mechanisms

As noted above, coordination mechanisms based on paper artifacts (e.g., forms, catalogues, time tables) have been around for ages and are used on a massive scale in modern work settings. While mundane and unassuming, they have crucial affordances: (a) the artifact can represent and convey stipulations among actors in a permanent and publicly accessible form; (b) the protocol and the artifact can be defined and specified by the actors themselves — operators, clerks, managers, auditors, etc. — by means of the ordinary skills of their professions; (c) actors have total control of the interpretation and execution of the protocol and can, under conditions of social accountability, modify or deviate from the protocol; (d) the

artifact can dynamically represent state changes to the protocol and mediate these among actors, and (e) multiple coordination mechanisms can be aligned, seamlessly and smoothly, by actors.

Nonetheless, such mechanisms have serious inherent limitations: (a) state changes to the protocol are conveyed by paper and similar unwieldy artifacts and the speed and pattern of propagation of changes to the state of the protocol are thus severely limited; (b) the protocol is only immediately visible to actors to the extent that the protocol is mapped onto the symbolic construct of the artifact that serves as an intermediate; (c) modifications to the protocol only take effect when or if actors become aware of them through other channels; (d) maintaining a conventional, paper-based coordination mechanism involves a plethora of mind-numbing operations; (e) it involves massive housekeeping efforts and it may thus be practically impossible for actors to obtain an overview of the state of the protocol, and (f) the seamless and smooth alignment of multiple intersecting coordination mechanisms is only feasible insofar as the same actors are involved with the coordination mechanisms in question.

These limitations with conventional coordination mechanisms become increasingly problematic as modern industrial, service, and administrative organizations need to be able to operate in a radically flexible and adaptive and yet highly coordinated fashion. In view of these issues, it seems obvious to explore whether it is possible to construct *computational coordination mechanisms* in which the allocation of functionality between actor and artifact is changed in such a way that the coordination mechanism, as a software device, incorporates the artifact in a computational form *as well as* aspects of the protocol which, again in a computational form, operates on the artifact.¹¹

Proposition 16. A *computational coordination mechanism* can be defined as a software device in which *the artifact* (in the sense of a permanent symbolic construct) *as well as aspects of the protocol* are incorporated in such a way that changes to the state of the protocol induced by one actor are conveyed, in accordance with the protocol, by the computational artifact to other actors.

Notice that, as far as the computational protocol is concerned, only *aspects* of the protocol are incorporated. As observed above, a protocol only has its (more or less) certain and agreed-to meaning within a certain social context, and it is inescapably under-specified. A computational coordination mechanism is invariably embedded within a social context of conventions and routines which competent members take

¹¹ The term artifact can create misunderstandings since it can refer to two different phenomena here: On one hand we have a computational artifact in the previously defined narrow sense of a permanent symbolic construct. But on the other hand one can, of course, conceive of the computational coordination mechanism *as a software artifact* which then would incorporate the computational protocol. For the sake of clarity, we will in this context restrict the use of the term 'artifact' to denote the permanent symbolic construct.

for granted but which are not amenable to incorporation in a computational protocol. Hence, a computational protocol cannot simply replace members' more or less tacit conventions and social competencies. For every computational coordination mechanism, there will exist facets of the protocol which are not incorporated in the computational protocol. That is, in the construction of a computational coordination mechanism, the protocol is split into a computational protocol and a residual non-computational or 'social' protocol. Furthermore, the precise allocation of functionality between human actors and computational coordination mechanism is a non-trivial analysis and design task. A computational coordination mechanism is not an immaculate reincarnation of a coordination mechanism.

Proposition 17. The specific *allocation of functionality* between human actors and a given computational coordination mechanism reflects the extent to which it is feasible to incorporate the various aspects of the conventions and routines of the social context into a computational protocol.

Now, 'no representation of the world is either complete or permanent' and coordination mechanisms are thus 'local and temporary closures' (Gerson and Star, 1986). That is, no computational coordination mechanism will be able to handle all aspects of articulation work in all work domains. Particular computational coordination mechanisms will be designed to support cooperating actors in specific aspects of their articulation work which are particularly complex and which, most likely, are specific to the given work domain. A computational coordination mechanism should thus be conceived of as a specialized software device that, while it is distinct from the field of work, interacts with 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.

Proposition 18. A computational coordination mechanism should be conceived of as *a specialized software device which interacts with a specific software application* so as to support articulation work with respect to the field of work as represented by the data structures and functionalities of that application.

On the basis of the analysis of the empirical studies of artifactually imprinted protocols in general and in particular coordinative protocol-cum-artifact dyads, a set of requirements for computational coordination mechanisms can be identified. The requirements can be organized into two categories: 'malleability' and 'linkability.'

4.1. Malleability

Since coordination mechanisms are ‘resources for situated action’ (Suchman, 1987), a computational coordination mechanism must be *malleable* in the sense that users are supported in defining its behavior.

Organizational demands and constraints change, and procedures and conventions change accordingly. In the case of the bug form mechanism, for example, the entire mechanism — the artifact as well as the procedures and conventions — was designed from scratch by the actors themselves and the actors were later on discussing various modifications to the protocol, for example the introduction of the role of a project manager in the protocol. It should thus be possible for actors to design and develop new computational coordination mechanisms and to make lasting modifications to existing ones. Accordingly:

Proposition 19. Actors should be able to *define the protocol* of a new coordination mechanism as well as to *redefine it* by making lasting modification to it, so as to be able to meet changing organizational requirements.

On the other hand, in view of the inexorably contingent nature of work, actors must be able to control the execution of the protocol, for instance by suspending a step, and to make local and temporary changes to the protocol, for instance by bypassing a step, by ‘rewinding’ a procedure, by escaping from a situation, or even by restarting the protocol from another point. For example, the bug form protocol was deviated from during its execution as erroneous classifications of bugs were discovered, as designers rejected the responsibility ascribed to them, etc. In other words, actors must be able to exercise *local control* over the execution of the protocol.

Proposition 20. A computational coordination mechanism must be constructed in such a way that actors are able to control its execution and make *local and temporary modifications* to its behavior to cope with unforeseen contingencies.

More generally stated, the specification (or instantiation) of an already defined protocol should not be conceived of as a singular act of creation. As stated in Proposition 7, a protocol is in principle under-specified. Thus, a protocol will typically be specified incrementally, at least to some extent, while it is executed in the course of the work. Furthermore, a protocol can be invoked implicitly, without any explicit announcements, for instance by certain actors taking certain actions (Strauss, 1985; Schäl, 1996). Thus, in order to allow for implicit understanding of certain aspects of articulation work as well as incomplete and not-yet complete specification, and also in order not to force actors to have to specify a coordination

mechanism more explicitly than deemed necessary, a computational mechanism must be constructed in such a way that a *partial specification* of the protocol is possible. That is, it should be feasible for attributes of the protocol specification to be left un-specified and for the missing specification to be provided, at a later stage, perhaps by another mechanism or by inference from actions taken by actors (cf. Proposition 15). For example, if actor A starts performing task *a*, it may be assumed that he or she is committed to accomplish task *a* and it may also be inferred that he or she has assumed the role *x* defined as responsible for task *a*.

Hence,

Proposition 21. A computational coordination mechanism must be constructed in such a way that *its behavior can be specified while it is being executed*, at least partially, so as to allow for incomplete initial specification of the protocol.

From these requirements (Proposition 19-21) follows that the definition and specification of the protocol must be ‘visible’ to actors, not only in the sense that it is accessible but also, and especially, that it *makes sense* to actors in terms of their articulation work:

Proposition 22. In order for actors to be able to define, specify, and control the execution of the mechanism, the protocol must be perspicuous, i.e., accessible and intelligible to actors *at the semantic level of articulation work*.

We are not here addressing the issue of which modality of presentation is most appropriate: graphs, trees, nets, matrices, or standardized prose. The point is that the protocol must be perceptible at a semantic level, at a level of granularity, and in a modality which is appropriate for the specific work domain at hand. That is, the objects and functional primitives available to actors for defining or specifying the protocol must be expressed in terms of categories of articulation work such as roles, actors, tasks, activities, conceptual structures, resources, and so on that are meaningful to the participants involved in terms of their everyday work activities.

Moreover, as a specialized software device supporting the articulation of distributed activities with respect to a particular field of work, as represented by the data structures and functionalities of a particular application, a computational coordination mechanism must be distinct from the other software components of that application (Proposition 12) in order for the mechanism to be malleable to actors at the semantic level of articulation work. If the coordination mechanism cannot be defined and specified independently of the other components of the system, malleability cannot be bounded and actors will thus be confronted with a vast space of possibilities at innumerable semantic levels, which will lead to utter confusion.

On the other hand, articulation work is always fundamentally conceived of with respect to the common field of work and in terms of the specific ordering of objects and processes constituting this field of work. The bug report protocol, for example, refers to entities of the field of work such as ‘module name,’ whereas the *kanban* protocol refers to such entities as ‘part name’ and ‘number of parts,’ etc. Accordingly, a computational coordination mechanism must be constructed in such a way that its stipulations can be related to and expressed in terms of the objects and processes of the field of work. For example, a computational coordination mechanism interacting with a collaborative-writing application to support the coordination of the flow of distributed activities of writing, editing, evaluating, reviewing, proofreading, and accepting contributions to a technical report would need to be able to relate to the usual data structures of the word processor application: text strings, formatting instructions, document components (paragraphs, sections, headings, tables, headers, footnotes, etc.). The same, of course, applies to the data structures and functionalities of the various domain-specific information systems such as MIS, OIS, CIM, and CASE systems which are part of the (wider) field of work of the cooperative work arrangement in question.

Proposition 23. A computational coordination mechanism must be constructed in such a way that actors, in defining and specifying the mechanism, can establish *relationships between components of the mechanism and the field of work* as represented by the data structures and functionalities of the target applications.

Since articulation work, as we noted earlier (Proposition 2), is a recursive function, changing a coordination mechanism may itself be done cooperatively, as part and parcel of the cooperative effort. That is, changing a coordination mechanism (permanently or temporarily) may itself be a cooperative activity which may need to be supported. Accordingly, it should be possible for actors to control the propagation of changes in terms of factors such as: When should a given change take effect? Which instances of the (previous) protocol should be affected, and how? Which actors should be notified, and how? Which complementary actions should be taken pursuant to the change, by whom? And so forth.

Proposition 24. Since a computational coordination mechanism must be malleable, it must be constructed in such a way that actors are supported in *controlling the propagation of changes to the protocol* within the cooperative work arrangement.

4.2. Linkability

Coordination mechanisms are local and temporary closures, as we have frequently noted. A given cooperative work arrangement will — in all but the most extreme circumstances — be working with multiple CSCW applications and they will need to articulate their distributed activities with respect to these different applications (as well as to many other aspects of their environment, of course). For example, in the domain of engineering design, the cooperative ensemble may be using applications such as project management tools, CAD tools, and process planning tools as well as generic ‘groupware’ tools such as departmental calendar systems and collaborative writing tools. To regulate articulation work with respect to these application an array of specialized coordination mechanisms may be devised. And, to confound matters, multiple mechanisms may be required to address specific aspects of articulation work with respect to each application. This may pose problems.

In the case of paper-based coordination mechanisms, the artifact is completely inert and any changes to the state of the protocol or the artifact are exclusively the result of actions by human actors (even when, as in the *kanban* case, it may appear the result of a monstrous, distributed machinery). Because actors are totally involved in the execution of paper-based mechanisms — they are completely ‘in the loop’ —, they are also relatively well placed to align the different coordination mechanisms with respect to each other, at least in so far as the different actors generally are equally involved in the use of the different mechanisms. When the allocation of functionality between artifact and actor changes, however, as a result of the introduction of computational coordination mechanisms, the ability of actors to align protocols in the former intuitive way may deteriorate.

Thus, as multiple coordination mechanisms are introduced to regulate articulation work with respect to multiple applications users will be inundated with overhead activities of aligning the different mechanisms: aligning each mechanism with changes to other mechanisms. In order not to create an impedance between the multitude of interlaced — individual and cooperative — coordinative activities, it should be possible for actors to link different coordination mechanisms addressing the different applications to facilitate a seamless alignment of articulation work with respect to these applications.

Proposition 25. A computational coordination mechanism should be constructed in such a way that it *can be linked to other coordination mechanisms* in its organizational context.

The requirement of linkability is not limited to links to other computational coordination mechanisms in the strict sense but applies to the relationship of a computational coordination mechanism to computational representations of the

organizational context in which the given cooperative work arrangement is embedded. In constructing a coordination mechanism it may for instance be appropriate to provide links to indices to common repositories (previous designs, components, work in progress, drawings, patents, etc.), indices to technical resources (processes, tools, machinery), indices to available personnel (skills, competencies, schedules), indices to statutory constraints, and so on (De Michelis and Grasso, 1993; Fuchs and Prinz, 1993; Prinz, 1993). The challenge is, as Ellis and Keddara aptly put it, to make groupware ‘organizationally aware’ (Ellis et al., 1995).

Since computational coordination mechanisms must be able to interact in a concerted fashion, they must be constructed by means of the same set of elements, at the same semantic level. To ensure that, a *general notation* for constructing computational coordination mechanism is required.

Proposition 26. To ensure comprehensive linkability of computational coordination mechanisms, a *general notation* for constructing computational coordination mechanism is required.

Now, malleability and linkability are evidently contradictory requirements, since the former hinges upon the possibility of *changing* the behavior of a mechanism while the latter hinges upon the *stability* of the behavior of other mechanisms. This conflict is unavoidable since no representation of the world is either complete or permanent. Nevertheless, the conflict can be alleviated in different ways: Basically, coordination mechanisms can be made tolerant of limited and relatively trivial modifications of other mechanisms by means of interface agents. However, when modifications are too radical to be handled by such interfaces, and the problem therefore recurses (Bowker and Star, 1991), the affected cooperative ensembles of course have to sort out the mess and negotiate a new arrangement. In line with the recursive nature of articulation work, such negotiations may themselves be governed by a suitable coordination mechanism. In that case, one coordination mechanism would take another cooperative work arrangement and perhaps also its coordination mechanisms as its field of work. This suggestion is not the fruit of idle speculation on our part but is grounded in the field study evidence. For example, in a study of the cooperative production of technical documentation in a Danish manufacturing company we have observed cases where one coordination mechanism, a ‘construction note’ which was normally used for governing the distributed process of negotiating proposed changes to designs, was now used for *governing the process of negotiating proposed changes to another coordination mechanism*, namely a ‘classification scheme’ that was used to govern the distributed production and dissemination of the technical documentation within the company (cf. Schmidt et al., 1995). In other words, the problem recurses, but so does the solution.

In lieu of a conclusion: The Ariadne notation

On the basis of the conceptual framework outlined in the preceding sections of this paper, especially the general requirements for computational coordination mechanisms (Proposition 19–25), a general notation for constructing computational coordination mechanisms has been implemented under the name Ariadne. As noted in the introduction, a proper description of the Ariadne notation is beyond the scope of this paper and has been published elsewhere (Simone and Schmidt, 1994; Simone et al., 1995a). Nonetheless, it seems appropriate to conclude our exposition by sketching very briefly the general shape of the Ariadne notation as derived from the empirical investigations and theoretical analyses.

The crucial point in developing a notation for constructing computational coordination mechanisms is to determine a repertory of elemental categories, at the semantic levels of articulation work (Proposition 22), by means of which coordinative protocols can be expressed. Taking Anselm Strauss' quite informal lexicon of articulation work (who, what, where, when, how, how soon, for how long, etc. (Strauss, 1985)) as our baseline, a set of elemental categories was derived from the studies of how artifactually imprinted protocols are designed and used by actors in everyday work activities, as shown in the table of Figure 4.¹² These categories represent the minimal set of categories required to express the protocols examined in these field studies.

Two qualifications are required here. Firstly, for the purpose of developing the Ariadne notation it was assumed that the set of elemental categories of articulation work identified in Figure 4 is complete and definite. Nevertheless, this set of categories has been derived empirically through an iterative process of induction, and the repertory is undoubtedly neither complete and nor definite. For the construction of a computational notation the assumption of completeness and definiteness is necessary, however, but this does not preclude the lexicon from evolving over time, as long as the set of categories at any given point in time can be taken to be finite. Secondly, this lexicon is not intended to be a particularly useful or comprehensive, terminology for ethnographic field work or for other kinds of empirical investigations of cooperative work (and, in fact, it is far too crude for such purposes). The set of categories of Figure 4 has been derived solely for the purpose of defining coordinative protocols with a view to constructing computational protocols.

[Figure 4 about here]

¹² A similar idea of selecting objects and related operations has been suggested by Malone and others (Malone and Crowston, 1990) as an initial foundation for an interdisciplinary 'coordination theory'. This effort has evolved into the current attempt to define 'tools for inventing organizations' (Malone et al., 1993).

In the table of elemental categories of articulation work, the categories and predicates are ordered along two dimensions: vertically, categories of articulation work with respect to the *cooperative work arrangement* versus *the field of work* (Proposition 1); horizontally, categories of *nominal* versus *actual* articulation work (Proposition 7). Two aspects require elaboration, albeit very briefly:

(1) The category termed ‘conceptual structures’ denotes the various constructs needed to express the conceptualizations of the field of work (definitions, classifications, etc.) that the members of the cooperative ensemble have adopted to be able to refer to the multifarious objects and processes of their common field of work in an orderly fashion.

(2) The distinction between nominal and actual (Proposition 7) identifies categories pertaining to the *definition* and *specification* of the computational coordination mechanism, respectively (propositions 19 and 20-21). The point of this distinction is that the transition from nominal to actual status is not merely a *refinement*, since the categories are qualitatively different. An *activity*, for example, denotes a work process as an unfolding course of action in terms of those aspects of a work process that are relevant to *doing* the work with the currently available resources. By contrast, a *task* denotes an operational intent, irrespective of how it is implemented (Andersen et al., 1990). In other words, a *task* is expressed in terms of *what*, an *activity* in terms of *how*.

The categories of articulation work are the basic building blocks made available by Ariadne whereas the elemental predicates are used to identify, and to define the meaning of, the attributes which characterize the categories and relations among them within the Ariadne notation.

The provision of a notation at the semantic level of articulation work distinguishes Ariadne from many proposed environments for the development of CSCW applications. These environments are typically based on partial repertoires of categories of articulation work. In some case, though, CSCW environments offer languages at the semantic level of the manipulation of general-purpose objects (e.g. Malone et al., 1992). To use such an environment for constructing a coordination mechanism, actors have to specialize these general purpose objects, that is, define a set of objects at the level of articulation work. This transformation involves the effort of defining an *ad hoc* model of articulation work and often leads to the definition of partial articulation work models, since the effort normally is undertaken in the context of developing a specific application. The *ad hoc* and partial character of these models becomes problematic when the resulting computational coordination mechanisms need to be modified and linked during the distributed and evolutionary construction process of their life cycles. By contrast, Ariadne provides a finite and expressive framework for developing a shared understanding across the activities of the distributed and evolutionary construction process. In this framework, the lexicon of articulation work plays the fundamental

role of governing the design of computational coordination mechanisms, of allowing for their malleability and linkability, of governing the impact of changes, of supporting the handling of partial specifications and, finally, of making the notation perceptible and hence usable to all categories of actors.

Finally, to meet the requirements of malleability and linkability the Ariadne notation has been given an internal structure organized into three levels that are called α , β , and γ , as illustrated in the central part of Figure 5.

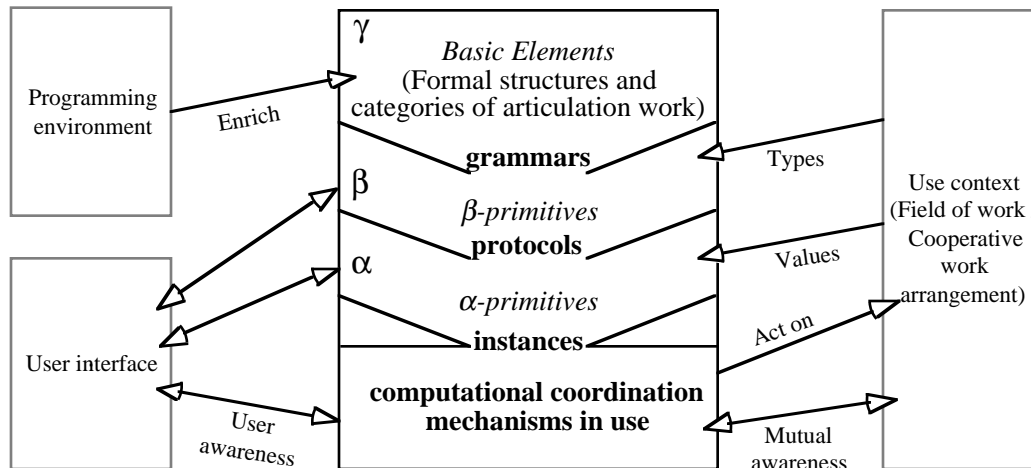


Figure 5. The architecture of the Ariadne notation.

The three levels are not hierarchical, and going from one level to the lower level is not identical to a refinement, since the information handled at each level is of a different nature. Rather, each level defines the ‘space of possibility’ for the lower one: the γ -level provides the grammars that can be applied at the β -level to define protocols, whereas the protocols defined at the β -level can be specified at the α -level. More specifically:

At the γ -level, it is possible to define or modify a grammar which can then be used to construct an infinite variety of protocols at the β -level. Building a grammar means determining the expressive power of a language for defining a class of computational coordination mechanisms, that is, the components that will constitute the computational coordination mechanism together with their structural interrelationships (for instance, to construct workflows or classification schemes) as well as the operational semantics associated to the elements of the grammar. The ‘space of possibility’ within which grammars can be defined at the γ -level is determined by the set of categories of articulation work underlying Ariadne at any point in time, and by the available set of formal structures for representing various relations (causal, hierarchical, instrumental, etc.) among the categories (the Basic Elements of Figure 5).

At the β -level, it is possible to define or modify the protocol itself according to the chosen grammar. For instance, a bug report protocol can be specified with more

or less emphasis on its distributed features by choosing an appropriate grammar. In this context, the user can determine the allocation of functionality between human actor and protocol, select methods for handling partial specifications, and make permanent changes to an existing protocol as part of its evolutionary design.

At the α -level, finally, it is possible to instantiate and activate the protocol in a particular situation and to do so in an incremental fashion (Proposition 21). The same protocol, e.g., the bug form protocol, can be executed repeatedly and concurrently by different actors. Moreover, the primitives at this level allow for the management of local changes (Proposition 20).

The different levels of Ariadne will typically be accessed by users with different skills. At the α - and β -level, the use of the notation does not require specialized skills, at least not skills more specialized than those required to use a spreadsheet application to construct a model, for example a household budget. That is, at these levels the use of the notation will merely require the ability of selecting and combining predefined items according to the rules of a relevant grammar and the associated semantics. The α - and β -levels are typically needed by end-users who, as part of their everyday work activities, use and design coordination mechanisms. The γ -level, on the other hand, is typically the realm of the ‘application designer’ or, in our framework, actors who define grammars needed by a particular community of end-users for defining their protocols.¹³

While the distinction between the β -level and the α -level of the notation can be recognized in almost all recent CSCW applications, the γ -level is unique. There are two reasons for the introduction of the γ -level. First of all, the γ -level makes it possible to define an appropriate language for constructing a family of computational coordination mechanisms, for instance for a particular work domain. The aim is to overcome a certain limitation of workflow systems, and of CSCW applications in general, namely that they impose a particular modelling approach. For example, the dynamic aspects of the protocols are in most applications and environments described through partial models of articulation work, of which the most common focus on the flow of objects across organizational units (individuals, tasks, etc.), on the flow of control across actions, on some communication patterns among roles (negotiation), or on some predefined combination of these (Ellis, 1979; Cook, 1980; Shepherd et al., 1990; Medina-Mora et al., 1992; Swenson et al., 1994). Since the adequacy of the language strongly depends on which aspects of articulation work are to be supported and on the organizational environment in which the planned coordination mechanisms are intended to operate, Ariadne does not impose a preconceived modeling approach. Rather, Ariadne allows designers to

¹³ Notice that while the construction of grammars could be performed by end-users with a specific aptitude and competence for modelling, the design of new elements for the notation is a pure programming activity that requires specialized technical skills.

adopt a particular interpretation of the underlying repertory of categories of articulation work.

The second reason for introducing a γ -level of the notation is the requirement of linkability of computational coordination mechanisms. In fact, the γ -level of Ariadne offers an Interoperability Language that the designer can make an integrated part of the language for constructing computational coordination mechanisms (Divitini et al., 1995; Simone et al., 1995a).

Referring again to Figure 5, the arrows on the left-hand side connect the basic elements at the γ -level with the framework in which they are developed. The connection to the Programming Environment is intended to stress that the notation is in a dynamic but disciplined relationship with its development environment, in the sense that any increase of the expressive power of Ariadne is realized by enriching the sets of basic elements of the notation while otherwise preserving the properties of the notation.

The notation also has an obvious connection with the user interface service. The design of Ariadne is concerned with the definition of the information necessary to design computational coordination mechanisms (that is, the appropriate expressive power of the language) and not with how this information is requested by or presented to the users. By this choice we are not denying the crucial role of the ‘material format of the artifact,’ as it is represented at the user interface. To the contrary, we realize and acknowledge that the design of the material format of computational coordination mechanisms requires a specialized and demanding research effort that is beyond the scope of Ariadne. However, some requirements of the user interface service are obvious. Firstly, of course, appropriate multi-modal representations must be devised for the various types of elements of the notation, the syntactic rules of their combination as well as their behavior. These representations should exhibit the same properties that characterize the notation, namely compositionality, malleability, linkability. Secondly, in accordance with the layered structure of the notation, the representations should be tailorable to different organizational roles in different application domains.

The arrows on the right-hand side of Figure 5 connect the computational coordination mechanism to the field of work and to the cooperative work arrangement, that is, to its context of use (Proposition 23). At the γ -level there are no connections to the context of use. In fact, the grammars are independent of the particular contexts of use in that they simply define the expressive power of the grammars to be used to define specific protocols. By contrast, the definition and specification of a protocol are related to a given (class of) field of work and work arrangement. The objects of the field of work and of the work arrangement are related to the categories of articulation work: as ‘types’ at the β level and as ‘instances’ at the α level. In protocols, ‘types’ are imported together with the related ‘methods’ which are then conveyed to the ‘instances’ in the standard way.

Thus, Ariadne explicitly requires a clear interface between the computational coordination mechanisms and their context of use and provides facilities for defining such an interface.

We conclude this section with some comments on the realization of Ariadne. In the initial development of the Ariadne notation, a considered decision was made to postpone the implementation and concentrate on developing a formal specification of its elements and on evaluating it against the requirements and scenarios derived from field studies. This strategy was adopted, consciously and explicitly, in order to avoid having the notation influenced, in an implicit and uncontrollable manner, by the inevitable limitations of currently available implementation platforms. The formal specification showed that it was feasible to construct malleable coordination mechanisms by means of the notation. Subsequently, a ‘concept demonstration’ of the formal specification of the notation was implemented in an environment which is particularly suitable to managing relational structures and their behavior. This partial implementation has established that the layered structure and compositionality of the Ariadne notation are workable (Simone et al., 1995a) and has demonstrated that an agent-based architecture is most suitable for Ariadne (Divitini et al., 1995; Simone et al., 1995b). A new implementation of Ariadne, based on such an architecture, is envisioned.

Acknowledgments

The development of the framework outlined in this paper was supported by ESPRIT Basic Research through Action 6225 (COMIC or Computer-based Coordination mechanisms in Cooperative Work) and by the Danish Scientific Research Council. The framework has been developed in collaboration with our colleagues in the COMIC project, especially our colleagues at Risø and Milano: Hans Andersen, Peter Carstensen, Monica Divitini, Betty Hewitt, Alberto Pozzoli, Tuomo Tuikka, and Carsten Sørensen. We are especially indebted to Peter Carstensen and Bjarne Kaavé for generously sharing their field study findings with us and to Liam Bannon and many others for invaluable comments on the manuscript.

Nominal		Actual	
Elemental categories of articulation work	Elemental predicates	Elemental categories of articulation work	Elemental predicates
<i>Articulation work with respect to the cooperative work arrangement</i>			
Role	assign to [Committed actor]; responsible for [Task, Resource]	Committed-actor	assume , accept, reject [Role]; initiate [Activity];
Task	point out, express; divide, relate; allocate, volunteer; accept, reject; order, countermand; accomplish, assess; approve, disapprove; realized by [Activity]; to be aligned with [Task]	Activity/Action	[Committed actor] initiate; [Actor-in-action] undertake, do, accomplish; realize [Task]; [Actor-in-action] makes publicly perceptible, monitors, is aware of, explains, questions; aligned with [Activity]
Personnel	locate, allocate, reserve;	Actor-in-action	does [Activity];
<i>Articulation work with respect to the field of work</i>			
Conceptual structures	categorize: define, relate, exemplify relations between categories pertaining to [Field of Work];	State of field of work	classify aspect of [State of field of work]; monitor, direct attention to, make sense of, act on aspect of [State of field of work];
Informational resource	locate, obtain access to, block access to;	Informational resources-in-use	show, hide content of; publicize, conceal existence of;
Material resource	locate, procure; allocate, reserve to [Task];	Material resources-in-use	deploy, consume; transform;
Technical resource	locate, procure; allocate, reserve to [Task];	Technical resources-in-use	deploy; use;
Infrastructural resource	reserve;	Infrastructural resources-in-use	use;

Figure 4. Elemental categories of articulation work model: The table identifies the elemental categories of articulation work and their predicates.

References

- Andersen, Hans H. K. (1994): Classification schemes: Supporting articulation work in technical documentation. In H. Albrechtsen (ed.): *ISKO '94. Knowledge Organisation and Quality Management, Copenhagen, Denmark, June 21-24, 1994*.
- Andersen, N. E., F. Kensing, J. Lundin, L. Mathiassen, A. Munk-Madsen, M. Rasbech, and P. Sørsgaard (1990): *Professional Systems Development — Experience, Ideas, and Action*. Englewood-Cliffs, New Jersey: Prentice-Hall.
- Bittner, Egon (1965): The Concept of Organization. *Social Research*, vol. 32, pp. 239-255.
- Bogia, Douglas P., William J. Tolone, Celsina Bignoli, and Simon M. Kaplan (1996): Issues in the Design of Collaborative Systems: Lessons from ConversationBuilder. In D. Shapiro, M. Tauber, and R. Traünmüller (eds.): *The Design of Computer Supported Cooperative Work and Groupware Systems*. Amsterdam: North Holland, pp. 401-422.
- Bogia, Douglas P., William J. Tolone, Simon M. Kaplan, and Eric de la Tribouille (1993): Supporting Dynamic Interdependencies among Collaborative Activities. In S. Kaplan (ed.): *COOCS '93. Conference on Organizational Computing Systems, Milpitas, California, November 1-4, 1993*. New York: ACM Press, pp. 108-118.
- Bowker, Geoffrey and Susan Leigh Star (1991): Situations vs. Standards in Long-Term, Wide-Scale Decision-Making: The Case of the International Classification of Diseases. In J. F. Nunamaker, Jr. and R. H. Sprague, Jr. (eds.): *Proceedings of the Twenty-Fourth Annual Hawaii International Conference on System Sciences, Kauai, Hawaii, January 7-11, 1991*. IEEE Computer Society Press, vol. IV, pp. 73-81.
- Bucciarelli, Louis L. (1988): Engineering Design Process. In F. A. Dubinskas (ed.): *Making Time. Ethnographies of High-Technology Organizations*. Philadelphia: Temple University Press, pp. 92-122.
- Carstensen, Peter (1994): The bug report form. In K. Schmidt (ed.): *Social Mechanisms of Interaction*. Lancaster, UK: Computing Department, Lancaster University, pp. 187-219. - [COMIC Deliverable 3.2. Available via anonymous FTP from ftp.comp.lancs.ac.uk].
- Carstensen, Peter H., Carsten Sørensen, and Henrik Borstrøm (1995a): Two is Fine, Four is a Mess: Reducing Complexity of Articulation Work in Manufacturing. *COOP '95. International Workshop on the Design of Cooperative Systems, Antibes-Juan-les-Pins, France, 25-27 January 1995*. INRIA Sophia Antipolis, France, pp. 314-333.
- Carstensen, Peter H., Carsten Sørensen, and Tuomo Tuikka (1995b): Let's Talk About Bugs! *Scandinavian Journal of Information Systems*, vol. 7, no. 1, pp. 33-54.
- Cook, Carolyn L. (1980): Streamlining office procedures - An analysis using the information control net model. *National Computer Conference, 1980*, pp. 555-565.
- Davenport, Thomas H. (1993): *Process Innovation: Reengineering Work through Information Technology*. Boston, Mass.: Harvard Business School Press.
- De Michelis, Giorgio and M. Antonietta Grasso (1993): How to put cooperative work in context: Analysis and design requirements. In L. Bannon and K. Schmidt (eds.): *Issues of Supporting Organizational Context in CSCW Systems*. Lancaster, UK: Computing Department, Lancaster University, pp. 73-100. - [COMIC Deliverable 1.1. Available via anonymous FTP from ftp.comp.lancs.ac.uk].
- Degani, Asaf and Earl L. Wiener (1990): *Human Factors of Flight-Deck Checklists: The Normal Checklist*. National Aeronautics and Space Administration, Ames Research Center, Moffett Field, California, May, 1990. [NASA Contractor Report 177549; Contract NCC2-377].
- Divitini, Monica, Carla Simone, Kjeld Schmidt, and Peter Carstensen (1995): *A multi-agent approach to the design of coordination mechanisms*, Roskilde University, DK-4000 Roskilde, Denmark, 1995. [WPCS-95-5].

- Egger, Edeltraud and Ina Wagner (1993): Negotiating Temporal Orders: The Case of Collaborative Time Management in a Surgery Clinic. *Computer Supported Cooperative Work (CSCW). An International Journal*, vol. 1, no. 4, pp. 255-275.
- Ellis, Clarence A. (1979): Information Control Nets. *Proceedings of the ACM Conference on Simulation, Measurement and Modeling, Boulder, Colorado, August 1979*.
- Ellis, Clarence A., Karim Keddara, and Grzegorz Rozenberg (1995): Dynamic Change Within Workflow Systems. In N. Comstock, C. Ellis, R. Kling, J. Mylopoulos, and S. Kaplan (eds.): *COOCS '95. Conference on Organizational Computing Systems, Milpitas, California, August 13-16, 1995*. New York: ACM Press, pp. 10-21.
- Fitzpatrick, Geraldine, William J. Tolone, and Simon M. Kaplan (1995): Work, Locales and Distributed Social Worlds. In H. Marmolin, Y. Sundblad, and K. Schmidt (eds.): *ECSCW '95. Proceedings of the Third European Conference on Computer-Supported Cooperative Work, 10-14 September 1995, Stockholm, Sweden*. Dordrecht: Kluwer Academic Publishers, pp. 1-16.
- Flores, Fernando, Michael Graves, Brad Hartfield, and Terry Winograd (1988): Computer Systems and the Design of Organizational Interaction. *ACM Transactions on Office Information Systems*, vol. 6, no. 2, pp. 153-172.
- Fuchs, Ludwin and Wolfgang Prinz (1993): Aspects of Organizational Context in CSCW. In L. Bannon and K. Schmidt (eds.): *Issues of Supporting Organizational Context in CSCW Systems*. Lancaster, UK: Computing Department, Lancaster University, pp. 11-47. - [COMIC Deliverable 1.1. Available via anonymous FTP from ftp.comp.lancs.ac.uk].
- Gerson, Elihu M. and Susan Leigh Star (1986): Analyzing Due Process in the Workplace. *ACM Transactions on Office Information Systems*, vol. 4, no. 3, pp. 257-270.
- Goody, Jack (1986): *The Logic of Writing and the Organization of Society*. Cambridge: Cambridge University Press.
- Goody, Jack (1987): *The Interface Between the Written and the Oral*. Cambridge: Cambridge University Press.
- Harper, Richard H. R. and John A. Hughes (1993): 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 G. Button (ed.): *Technology in Working Order. Studies of work, interaction, and technology*. London and New York: Routledge, pp. 127-144.
- Harper, Richard R., John A. Hughes, and Dan Z. Shapiro (1989): *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.
- Harrington, Joseph (1979): *Computer Integrated Manufacturing*. Malabar, Florida: Krieger.
- Harrington, Joseph (1984): *Understanding the Manufacturing Process. Key to Successful CAD/CAM Implementation*. New York: Marcel Dekker.
- Heath, Christian, Marina Jirotko, Paul Luff, and Jon Hindmarsh (1995): Unpacking Collaboration: the Interactional Organisation of Trading in a City Dealing Room. *Computer Supported Cooperative Work (CSCW). An International Journal*, vol. 3, no. 2, pp. 147-165.
- Heath, Christian and Paul Luff (1992): 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, pp. 69-94.
- Holt, Anatol W. (1985): Coordination Technology and Petri Nets. In G. Rozenberg (ed.): *Advances in Petri Nets 1985*, vol. 222. Berlin: Springer-Verlag, pp. 278-296.
- Hutchins, Edwin (1986): Mediation and Automatization. *Quarterly Newsletter of the Laboratory of Comparative Human Cognition [University of California, San Diego]*, vol. 8, no. 2, pp. 47-58.

- Johnson, Philip (1992): Supporting Exploratory CSCW with the EGRET Framework. In J. Turner and R. Kraut (eds.): *CSCW '92. Proceedings of the Conference on Computer-Supported Cooperative Work, Toronto, Canada, October 31 to November 4, 1992*. New York: ACM Press, pp. 298-305.
- Kaavé, Bjarne (1990): *Undersøgelse af brugersamspil i system til produktionsstyring*. M.Sc diss. Technical University of Denmark, 1990.
- Kaplan, Simon M., William J. Tolone, Douglas P. Bogia, and Celsina Bignoli (1992): Flexible, Active Support for Collaborative Work with Conversation Builder. In J. Turner and R. Kraut (eds.): *CSCW '92. Proceedings of the Conference on Computer-Supported Cooperative Work, Toronto, Canada, October 31 to November 4, 1992*. New York: ACM Press, pp. 378-385.
- Kreifelts, Thomas, Elke Hinrichs, Karl-Heinz Klein, Peter Seuffert, and Gerd Woetzel (1991a): Experiences with the DOMINO Office Procedure System. In L. Bannon, M. Robinson, and K. Schmidt (eds.): *ECSCW '91. Proceedings of the Second European Conference on Computer-Supported Cooperative Work, 24-27 September 1991*. Amsterdam: Kluwer Academic Publishers, pp. 117-130.
- Kreifelts, Thomas, Frank Victor, Gerd Woetzel, and Michael Weitass (1991b): A Design Tools for Autonomous Agents. In J. M. Bowers and S. D. Benford (eds.): *Studies in Computer Supported Cooperative Work. Theory, Practice and Design*. Amsterdam: North-Holland, pp. 131-144.
- Malone, Thomas W. and Kevin Crowston (1990): What is Coordination Theory and How Can It Help Design Cooperative Work Systems *CSCW '90. Proceedings of the Conference on Computer-Supported Cooperative Work, Los Angeles, Calif., October 7-10, 1990*. New York, N.Y.: ACM press, pp. 357-370.
- Malone, Thomas W., Kevin Crowston, Jintae Lee, and Brian Pentland (1993): Tools for inventing organizations: Toward a handbook of organizational processes. *Proceedings of the 2nd IEEE Workshop on Enabling Technologies Infrastructure for Collaborative Enterprises, Morgantown, West Virginia, April 20-22, 1993*.
- Malone, Thomas W., Hum-Yew Lai, and Christopher Fry (1992): Experiments with Oval: A Radically Tailorable Tool for Cooperative Work. In J. Turner and R. Kraut (eds.): *CSCW '92. Proceedings of the Conference on Computer-Supported Cooperative Work, Toronto, Canada, October 31 to November 4, 1992*. New York: ACM Press, pp. 289-297.
- Malone, Thomas W., Hum-Yew Lai, and Christopher Fry (1995): Experiments with Oval: A Radically Tailorable Tool for Cooperative Work. *ACM Transactions on Office Information Systems*, vol. 13, no. 2, pp. 177-205.
- Medina-Mora, Raul, Terry Winograd, Rodrigo Flores, and Fernando Flores (1992): The Action Workflow Approach to Workflow Management Technology. In J. Turner and R. Kraut (eds.): *CSCW '92. Proceedings of the Conference on Computer-Supported Cooperative Work, Toronto, Canada, October 31 to November 4, 1992*. New York: ACM Press, pp. 281-288.
- Monden, Yasuhiro (1983): *Toyota Production System. Practical Approach to Production Management*. Norcross, Georgia: Industrial Engineering and Management Press, Institute of Industrial Engineers.
- Norman, Donald A. (1991): Cognitive Artifacts. In J. M. Carroll (ed.): *Designing Interaction. Psychology at the Human-Computer Interface*. Cambridge: Cambridge University Press, pp. 17-38.
- Norman, Donald A. and Edwin L. Hutchins (1988): *Computation via Direct Manipulation*. Institute for Cognitive Science, University of California, San Diego, La Jolla, California, 1 August, 1988. [ONR Contract N00014-85-C-0133].

- Prinz, Wolfgang (1993): TOSCA: Providing organisational information to CSCW applications. In G. De Michelis, C. Simone, and K. Schmidt (eds.): *ECSCW '93. Proceedings of the Third European Conference on Computer-Supported Cooperative Work, 13-17 September 1993, Milan, Italy*. Dordrecht: Kluwer Academic Publishers, pp. 139-154.
- Pycock, James (1994): Mechanisms of interaction and technologies of representation: Examining a case study. In K. Schmidt (ed.): *Social Mechanisms of Interaction*. Lancaster, UK: Computing Department, Lancaster University, pp. 123-148. - [COMIC Deliverable 3.2. Available via anonymous FTP from ftp.comp.lancs.ac.uk].
- Pycock, James and Wes Sharrock (1994): The fault report form: Mechanisms of interaction in design and development project work. In K. Schmidt (ed.): *Social Mechanisms of Interaction*. Lancaster, UK: Computing Department, Lancaster University, pp. 257-294. - [COMIC Deliverable 3.2. Available via anonymous FTP from ftp.comp.lancs.ac.uk].
- Roth, Emilie M. and David D. Woods (1989): Cognitive Task Analysis: An Approach to Knowledge Acquisition for Intelligent System Design. In G. Guida and C. Tasso (eds.): *Topics in Expert System Design. Methodologies and Tools*. Amsterdam: North-Holland, pp. 233-264.
- Schäl, Thomas (1996): System Design for Cooperative Work in the Language Action Perspective: A Case Study of The Coordinator. In D. Shapiro, M. Tauber, and R. Traünmuller (eds.): *The Design of Computer Supported Cooperative Work and Groupware Systems*. Amsterdam: North Holland, pp. 377-400.
- Schmidt, Kjeld (1991a): Cooperative Work. A Conceptual Framework. In J. Rasmussen, B. Brehmer, and J. Leplat (eds.): *Distributed Decision Making. Cognitive Models for Cooperative Work*. Chichester: John Wiley & Sons, pp. 75-109.
- Schmidt, Kjeld (1991b): Riding a Tiger, or Computer Supported Cooperative Work. In L. Bannon, M. Robinson, and K. Schmidt (eds.): *ECSCW '91. Proceedings of the Second European Conference on Computer-Supported Cooperative Work, 24-27 September 1991*. Amsterdam: Kluwer Academic Publishers, pp. 1-16.
- Schmidt, Kjeld (1994): *Modes and Mechanisms of Interaction in Cooperative Work*. Risø National Laboratory, P.O. Box 49, DK-4000 Roskilde, Denmark, 1994. [Risø-R-666(EN)].
- Schmidt, Kjeld and Liam Bannon (1992): Taking CSCW Seriously: Supporting Articulation Work. *Computer Supported Cooperative Work (CSCW). An International Journal*, vol. 1, no. 1-2, pp. 7-40.
- Schmidt, Kjeld, Carla Simone, Monica Divitini, Peter Carstensen, and Carsten Sørensen (1995): *A 'contrat sociale' for CSCW systems: Supporting interoperability of computational coordination mechanisms*, Roskilde University, DK-4000 Roskilde, Denmark, 1995. [WPCS-95-7].
- Schonberger, Richard J. (1982): *Japanese Manufacturing Techniques. Nine Hidden Lessons in Simplicity*. New York: Free Press.
- Selznick, Philip (1948): Foundations of the Theory of Organization. *American Sociological Review*, vol. 13, pp. 25-35.
- Shepherd, Allan, Niels Mayer, and Allan Kuchinsky (1990): Strudel - An Extensible Electronic Conversation Toolkit. *CSCW 90, Los Angeles, CA, October 7-10 1990, New York, N.Y.* ACM press, pp. 93-104.
- Simone, Carla, Monica Divitini, and Kjeld Schmidt (1995a): A notation for malleable and interoperable coordination mechanisms for CSCW systems. In N. Comstock, C. Ellis, R. Kling, J. Mylopoulos, and S. Kaplan (eds.): *COOCS '95. Conference on Organizational Computing Systems, Milpitas, California, August 13-16, 1995*. New York: ACM Press, pp. 44-54.

- Simone, Carla, Monica Divitini, Kjeld Schmidt, and Peter Carstensen (1995b): A Multi-Agent Approach to the Design of Coordination Mechanisms. In V. Lesser (ed.): *Proceedings of the First International Conference on Multi-Agent Systems, San Francisco, Calif., USA, June 12-14, 1995*. Menlo Park, Calif.: AAAI Press.
- Simone, Carla and Kjeld Schmidt (eds.) (1994): *A Notation for Computational Mechanisms of Interaction*. Lancaster, UK: Computing Department, Lancaster University. - [COMIC Deliverable 3.3. Available via anonymous FTP from ftp.comp.lancs.ac.uk].
- Sørensen, Carsten (1994a): The augmented bill of materials. In K. Schmidt (ed.): *Social Mechanisms of Interaction*. Lancaster, UK: Computing Department, Lancaster University, pp. 221-236. - [COMIC Deliverable 3.2. Available via anonymous FTP from ftp.comp.lancs.ac.uk].
- Sørensen, Carsten (1994b): The CEDAC board. In K. Schmidt (ed.): *Social Mechanisms of Interaction*. Lancaster, UK: Computing Department, Lancaster University, pp. 237-245. - [COMIC Deliverable 3.2. Available via anonymous FTP from ftp.comp.lancs.ac.uk].
- Sørensen, Carsten (1994c): The product classification scheme. In K. Schmidt (ed.): *Social Mechanisms of Interaction*. Lancaster, UK: Computing Department, Lancaster University, pp. 247-255. - [COMIC Deliverable 3.2. Available via anonymous FTP from ftp.comp.lancs.ac.uk].
- Stinchcombe, Arthur L. (1974): *Creating Efficient Industrial Administrations*. New York and London: Academic Press.
- Strauss, Anselm (1985): Work and the Division of Labor. *The Sociological Quarterly*, vol. 26, no. 1, pp. 1-19.
- Strauss, Anselm (1988): The Articulation of Project Work: An Organizational Process. *The Sociological Quarterly*, vol. 29, no. 2, pp. 163-178.
- Strauss, Anselm (1994): *Continual Permutations of Action*. New York: Aldine de Gruyter.
- Strauss, Anselm, Shizuko Y. Fagerhaugh, Barbara Sucek, and Carolyn Wiener (1985): *Social Organization of Medical Work*. Chicago and London: University of Chicago Press.
- Suchman, Lucy A. (1982): Systematics of Office Work. Office Studies for Knowledge-Based Systems, Digest. *Office Automation Conference, San Francisco, April 5-7, 1982*, pp. 409-412.
- Suchman, Lucy A. (1983): Office Procedures as Practical Action: Models of Work and System Design. *ACM Transactions on Office Information Systems*, vol. 1, no. 4, pp. 320-328.
- Suchman, Lucy A. (1987): *Plans and situated actions. The problem of human-machine communication*. Cambridge: Cambridge University Press.
- Suchman, Lucy A. and Eleanor Wynn (1984): Procedures and Problems in the Office. *Office: Technology and People*, vol. 2, pp. 133-154.
- Swenson, K. D., R. J. Maxwell, T. Matsumoto, B. Saghari, and K. Irwin (1994): A business process environment supporting collaborative planning. *Collaborative Computing*, vol. 1, no. 1, pp. 15-24.
- Winograd, Terry (1986): A language/action perspective on the design of cooperative work. *CSCW '86. Proceedings. Conference on Computer-Supported Cooperative Work, Austin, Texas, December 3-5, 1986*. ACM, New York, N. Y., pp. 203-220.
- Winograd, Terry and Fernando Flores (1986): *Understanding Computers and Cognition: A New Foundation for Design*. Norwood, New Jersey: Ablex Publishing Corp.
- Wynn, Eleanor (1991): Taking Practice Seriously. In J. Greenbaum and M. Kyng (eds.): *Design at Work: Cooperative Design of Computer Systems*. Hillsdale, New Jersey: Lawrence Erlbaum, pp. 45-64.
- Wynn, Eleanor H. (1979): *Office Conversation as an Information Medium*. Ph. D. dissertation, University of California, Berkeley, , 1979.

- Zerubavel, Eviatar (1979): *Patterns of Time in Hospital Life: A Sociological Perspective*. Chicago and London: University of Chicago Press.
- Zimmerman, Don H. (1966): *Paper work and people work: A study of a public assistance agency*. Ph.D. Dissertation, University of California, Los Angeles, Los Angeles, 1966.
- Zimmerman, Don H. (1969a): Record-Keeping and the Intake Process in a Public Welfare Agency. In S. Wheeler (ed.): *On Record: Files and Dossiers in American Life*. New York: Russell Sage Foundation, pp. 319-354.
- Zimmerman, Don H. (1969b): Tasks and Troubles: The Practical Bases of Work Activities in a Public Assistance Agency. In D. A. Hansen (ed.): *Explorations in Sociology and Counseling*. New York: Houghton-Mifflin, pp. 237-266.