From the social to the systematic
Mechanisms supporting coordination in design

Peter H. Carstensen* and Carsten Sørensen•
**Systems Analysis Department, Risø National Laboratory, Roskilde, Denmark.*
E-mail: peter.carstensen@risoe.dk

• *Department of Infomatics, University of Göteborg, Göteborg, Sweden.*
Visiting Fellow at Warwick Business School, Coventry, United Kingdom
E-mail: carsten@adb.gu.se

Abstract: Large design and manufacturing projects are conducted in elaborate settings. Interdependent specialists work together, building complex systems. A substantial part of their daily work concerns the coordination of distributed work. This paper reports from a field study at Foss Electric, a Danish manufacturing company, where the development of an instrument for testing the quality of raw milk was studied. Scheduled and informal project meetings together with paper-based coordination systems were the primary means of managing the complexity of coordinating work within the project. This paper investigates the origination, use, and function of these coordination mechanisms applying a Coordination Mechanism perspective (Schmidt and Simone, 1996). We argue that the complexity of coordinating distributed work in large design projects result in the adoption of coordination systems. These systems formalize aspects of coordination work through artifacts, procedures for use and conventions.

1. Introduction

Large scale design and manufacturing projects are complex human activities involving many people with different areas of competence. An abundance of decisions have to be made by mutually interdependent actors. When the number of people involved in a project exceeds the limit of a few, they need to examine the state of affairs in the field of work they, for example, need answers to questions such as “how many sub-assembly A’s are currently located in Hall 12?”. The actors involved in the project will need to coordinate their activities, including, for example, meshing, allocating and scheduling others activities, actors and resources (Strauss, 1985; Schmidt, 1994). When work requires the handling of a multitude of intertwined and interdependent activities, the complexity of coordinating these activities increases tremendously. It is impossible to apply *ad hoc* based modes of interaction based on human social skills and practices only.

This paper aims to illustrate the importance of formalizing aspects of coordination work in complex manufacturing projects in order to cope with the need for coordination. We investigate the following hypothesis:

Computer Supported Cooperative Work: The Journal of Collaborative Computing. Vol. **6**, issue 4. pp: 387-413, December, 1996. Kluwer Academic Publishers, Netherlands.

Complex manufacturing projects, where many participants from different areas of competence need to coordinate distributed activities, will invent and adopt artifacts and accompanying procedures in order to handle the complexity of coordinating work.

This hypothesis is substantiated by the analysis of research from fieldwork from a project in one organization. The paper illustrates the invention and use of systems prescribing and formalizing aspects of coordination work through prescriptive procedures and standardized structures formalizing the work flow and through classification schemes establishing common "languages", which reflect relevant structures from the field of work. It is shown how the actors apply the formal systems as means of reducing the complexity of the coordination tasks to be conducted.

Humans are good at applying *ad hoc* solutions to the contingencies of work and any formalization applied might be inappropriate and impose unnecessary discipline (Suchman, 1994). The field study however, clearly shows that the actors involved chose increased formalization of the work processes as a means for handling the required coordination despite the potential problems of increased rigidity and possible control of the work. Kraut and Streeter (1995) have studied coordination in software development and they argue for the importance of informal direct communication among the software designers. They do however, also recognize that more formalized means must also be applied, due to the transaction costs and ephemeral nature of information transferred informally. Our focus is mainly on how the informal means develop into more formal means when the need for coordination increases.

Humans are extremely good at coordinating work by monitoring each other, and by communicating by means of *ad hoc* modes of interaction (cf. e.g., Heath et al., 1993). Problems will however, emerge in highly complex work when, for example:

(1) the cooperative work setting includes many geographically distributed actors;
(2) there are a large number of interdependent activities, actors, or resources;
(3) different areas of competence with different conceptualizations are involved;
(4) different goals are represented;
(5) work is carried out over a long time-span. If the participants wish to avoid coordinating most of the time, different measures can be taken, such as, optimizing the organization of work, applying structured project meetings, introducing standard operating procedures (Mintzberg, 1979).

In order to support the coordination work, symbolic artifacts such as forms, schedules, and classification schemes are often introduced together with conventions and written organizational procedures stipulating the usage of the artifacts. These artifacts and their concomitant procedures and conventions can be seen as mechanisms reducing complexity by stipulating and mediating coordination work (Schmidt, 1994). They stipulate certain aspects of who is doing what, when, where, how, and why. The artifacts will often serve other, more domain specific, purposes as well, besides that of reducing complexity of coordination work.

Much research in the CSCW (Computer Supported Cooperative Work) field has addressed topics related to understanding and supporting the communication among interdependent actors. For example ethnographic studies of how interaction is organized (e.g., Hughes et al., 1992; Heath et al., 1993); support of real-time high bandwidth

communication and Media Space (e.g., Heath and Luff, 1992b; Dourish, 1993; Ishii et al., 1993); and modelling the communication (e.g., Flores et al., 1988; Agostini et al., 1994). Although most of the empirically studies in the CSCW field analyze work settings involving many interdependent actors, they primarily focus on the work of relatively few. The domains investigated are, furthermore, most often characterized by a high degree of (often time-critical) monitoring and regulating activities among the actors (Bentley et al., 1992; Heath and Luff, 1992a; Fillipi and Theureau, 1993; Heath et al., 1993).

This study addresses cooperative aspects of engineering and software design, similar to other studies such as Anderson *et al.* (1993) and Bucciarelli (1984). The research documented here, is based on a field study of a large-scale manufacturing project. The primary methodologies used were semi-structured interviews, project document inspection, and non-participant observation. We have interpreted the data collected using theories on complexity (Simon, 1981; Woods, 1988) and by applying the concept of Coordination Mechanisms (Schmidt et al., 1993; Schmidt and Simone, 1996).

Because we were studying a design project, the field of work was in a number of ways different from, for example, work consisting mainly of monitor and regulation activities. The work we have studied was neither time nor safety critical. It had a very important constructive, as opposed to analytical, element. The overall project goal was to specify an instrument which could be manufactured within a broad range of constraints. Hence, we have focused on the cooperative aspects of a design process carried out over a long time span, in a large scale setting, and involving people with different areas of competence.

Apart from obtaining a general understanding of the work performed, the main objective was to identify and characterize mechanisms supporting coordination work. Further work is concerned with the design of computer-based artifacts supporting coordination of manufacturing work. Others have explicitly addressed and modelled coordination work (e.g., Holt, 1988; Malone and Crowston, 1990; Kaplan et al., 1992; Fitzpatrick et al., 1995), but these are to a very limited degree based on empirical studies of artifacts introduced in order to cope with the complexity of the coordination work.

In the following section we briefly introduce the research approach. We then characterize the company and the work setting studied. Section 4 describe the different mechanisms used to support the coordination work in the project studied, followed by Section 5, which contains an analysis of how these mechanisms support the coordination. The paper concludes with a brief discussion of our findings and some reflections on the implications for the design of computer based coordination mechanisms.

2. Research Approach

Field studies are essential in order to obtain a coherent understanding of, and to design computer-based tools for, manufacturing (Keyser, 1992; Siemieniuch, 1992). The aim of our empirically based effort was to analyze cooperative work in a manufacturing

setting where participants face the complexity and uncertainty of going from a design concept to determining exactly how to manufacture the product. We conducted our study at Foss Electric, a highly specialized company manufacturing complex instruments for measuring quality parameters of agricultural products.

The research approach used for collecting data at Foss Electric can be characterized as qualitative research heavily inspired by both Work Analysis (Schmidt and Carstensen, 1990; Carstensen and Schmidt, 1993), and by ethnographic approaches to studying engineering work (Bucciarelli, 1984).Qualitative research implies collecting data by, for example, interviews and observation, with the purpose of capturing the richness of worldly realism, hence potentially jeopardizing the tightness of control (Mason, 1989).

We conducted a series of interviews and also observed one particular project, the S4000 project. Here a new instrument for analytical testing of raw milk was designed and put into production. The interviews can primarily be characterized as open-ended qualitative interviews (Patton, 1980). 10 short and 11 long interviews (lasting several hours) were conducted, and we participated in approximately 10 project meetings. We have, furthermore, spent about 100 man-hours observing the design and production process. The interviews and observations were conducted over a period of approximately 4 month and were followed up by several meetings with the project members. At these meetings we presented and got feedback on our observations and interpretations of their work.

We have, furthermore, developed a set of requirements for computer support of coordination of manufacturing work (Carstensen and Sørensen, 1994; Carstensen et al., 1995).

We have applied Woods (1988) framework to characterize the complexity of the work in the S4000 project. As sources of complexity we basically distinguish between the field of work; the cooperative work arrangement; the environment surrounding and constraining the work arrangement; and the reference dimension of time and space (cf. Schmidt and Simone, 1996).

The approach for analyzing the artifacts identified in the field study is taken from the concept of Coordination Mechanisms (Schmidt et al., 1993; Schmidt and Simone, 1996). A coordination mechanism is defined as a mechanism that, by means of a set of conventions and prescribed procedures and supported by a symbolic artifact with a standardized format, stipulates and mediates the coordination of the distributed activities of large cooperative ensembles (Schmidt and Simone, 1996). It is a conceptual framework for describing artifacts supporting the process of coordinating who is doing what, when, where, how, and why. The concept introduces an analytical distinction between different objects of coordination work. These are:

(1) actors;
(2) roles;
(3) responsibilities and obligations;
(4) tasks, i.e., an operational intention;
(5) activities, i.e., an unfolding course of actions;
(6) conceptual structures, e.g., classification;

(7) informational, material, technical, or infrastructural resources.

The objects of coordination work reflected in a coordination mechanism can be interpreted as conceptualizations of structures in the work arrangement, the field of work, the wider organizational setting, or as being references to time and space. For a detailed description of the concept of Coordination Mechanisms see Schmidt and Simone (1996).

To briefly illustrate the concept of Coordination Mechanisms, let us consider a simple scheduling system supporting people in coordinating the scheduling of meetings by providing a standard calendar form which is routed amongst the participants. They enter suggestions for meetings and for meeting rooms. When meeting rooms or time-slots are double-booked by one of the participants, the relevant people involved will be notified in order to perform re-scheduling. This system can be seen as containing a coordination mechanism. The calendar form is a standardized artifact, perhaps with rows of dates and columns of available meeting rooms. It is symbolic in the sense that booking a room at a particular date only reserves the room, it does not change the state of the room itself. It contains a routing scheme, and perhaps also rules for how to fill in a form, i.e., it contains a protocol. This particular coordination mechanism reduces the complexity of distributed actors coordinating where and when to meet with whom—they perhaps only need to engage in face-to-face meetings or telephone conversations in rare cases.

Examples of coordination mechanisms (paper and board-based artifacts and concomitant conventions and procedures) will be discussed in Section 4 in terms of origination, format of the artifact used, purpose, and usage. The artifacts are subsequently analyzed in terms of the objects of coordination work mentioned above. In order to relate these to the general work setting and situation we will however, in Section 3 briefly introduce the work setting and illustrate the complexity of the S4000 project.

3. Field Study

The field study reported was conducted at the company Foss Electric which develop, manufacture, and market equipment for analytical measuring quality parameters of agricultural products.

3.1 Foss Electric

Equipment for measuring quality parameters of agricultural products is a highly specialized field. There are only a few companies in the marketplace and Foss Electric is among the largest in the world. Both R&D and production are localized in Denmark with subsidiary companies in England and Germany. Sales, service and distributors are located in branches around the world. The Foss Electric holding company employs approximately 700 people. The products manufactured are used for measuring the compositional quality of milk (the fat content, the count of protein, lactose, somatic cells, bacteria, etc.), the composition and micro biological quality of food products, and

for measuring grain quality. The measurement technologies are typically infrared spectography, fluorescence microscopy, or bacteriological testing. The customers are most often laboratories, slaughterhouses, dairies, etc. Due to the high market specialization, few competitors, and an increasing centralization of laboratories, the innovation towards new, better and faster measuring techniques is one of the most important strategic goals for the company. Research and development are hence essential activities.

Foss Electric has implemented concurrent engineering (Harrington, 1984; Helander and Nagamachi, 1992) generating integration between manufacturing functions throughout the development process. The organization is to a large extent structured in terms of projects. The development from concept to final manufactured product involves a number of intermediate products:

(1) a product concept defining the overall architecture and interaction between the technologies involved;
(2) a few functional models (mock-ups);
(3) five to ten prototypes of the instrument used for verifying detailed ideas and designs;
(4) a test series of five to ten instruments in order to test manufacturability of the product.

Foss Electric adopted a prototyping strategy due to the high degree of uncertainty in these types of projects. Our field study concentrated on one of the largest projects at Foss Electric—the System 4000 (S4000) project.

Figure 1: The S4000 system being tested in the Quality Control department.

3.2 The S4000 Project

The objective of the S4000 project was to build a new instrument for analytical testing of raw milk (see Figure 1). It was the first time that Foss Electric had embarked on building a "system" in which two instruments were integrated, sharing pipette unit, conveyor and computer. Compared to previous instruments for testing milk, the S4000 system introduced the measurement of new milk parameters (e.g. urea and critic acid). It was also planned to improve the measurement speed compared to previous products. The S4000 was the first product integrated with an Intel-based 486 PC. The configuration, control, and operation of the instrument should be made via a Windows user interface, i.e., a graphical user interface and use of mouse and keyboard. The instrument is grouped into a number of functional units, for example cabinet, PC, pipette unit, conveyer, flow-system, and measurement unit. In total the instrument consists of more than 8000 components.

The project duration was 2 1/2 years, designing the first version of the S4000 system, and more than 50 people were involved in the project. The core personnel included a number of designers from mechanic design, electronic design, software design, and chemistry. In addition there was a handful of draught-persons and several individuals from the production, model shop, marketing, quality assurance, quality control, and service departments. Senior management was also involved.

The following illustrates how the S4000 project involved a series of highly complex situations which had to be managed on a daily basis. The complexity of the S4000 project can be characterized, applying the four dimensions inspired by Woods' (1988) framework:

(1) Dynamism in work situations is often caused by the need to handle a number of concepts, requirements, etc. which are dynamic by nature. There is, for example, inherently dynamism in mechanical design and process planning at Foss Electric. The use of existing machines is optimized, and new machines are introduced. The utilization of the production machinery, and thereby the set of manufacturing processes that the production function can offer is constantly changing.

(2) Many highly interacting parts: The field of work is constituted by a large number of interconnected parts, components, etc. Thus, a failure can potentially be caused by one of many circumstances, and require one of many solutions. The problem is ill structured (Simon, 1973). In order to reduce the problem space, the actors use heuristics and general strategies. Just to illustrate, by way of two examples, in the S4000 system there were close interaction between more than 20 different software applications in the software complex (more than 200.000 lines of C-code). The software and hardware was also closely connected with the mechanical and chemical processes in the flow and measurement system. As one of the software designers stated:

> "The problem we have right now is that the software architecture is difficult to decompose so much that one designer can handle a component. We are all working on several components and work on one component involves two to four people, and perhaps even some of the electronic designers too."

(4) Uncertainty exists in many complex work situations. The actors are often confronted with missing, incomplete, ambiguous, erroneous and contradictory information and they must act on their cooperate judgement (Woods, 1988). The

problem itself is often not evident (Dery and Mock, 1985). Uncertainty is usually caused by external occurrences, or it can arise through failures, noise, time delays, influence of previous events, unclear requirement, etc. from the field of work. In the S4000 project measurement of completely new parameters in raw milk (e.g., urea and citric acid) had to be developed. Also the software controlling the instrument was to be implemented as a Microsoft Windows application. This involved developing with a programming language, an application run-time platform, a user interface, and a set of basic communication protocols which the software designers had no previous experience of.

(4) Mutually interdependent actors: Highly complex work situations are most often handled by mutually interdependent actors. This includes a number of secondary activities for handling distributed activities, such as dividing, allocating, coordinating, scheduling, meshing, and interrelating (Strauss, 1988). The S4000 project was a large project involving many actors with different competencies. It was significantly more complex than the usual projects, as characterized by on of the software designers:

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

Because of the concurrent engineering strategy, different conflicting requirements is exposed and negotiated early in the manufacturing process. Although this is an overall advantage in relation to the quality of the finished product, it can lead to a significant burden in relation to the coordination work demanded in the design process.

4. Mechanisms Supporting Coordination Work

As briefly illustrated above, the S4000 project was an extremely complex project in many respects. One of the most predominant complexity factors was the huge amount of coordination work required. There were however, at least the following four types of countermeasures, reducing the complexity of coordination work in the project:

(1) a project oriented matrix organization was implemented at Foss Electric. Projects were organizational units with the project manager serving as a "head of department" and all participants were physically located in the same area;
(2) to ensure the overall goals were met, the project implemented a set of scheduled meetings, some weekly and most twice per month. In the most intense phases of the project 27 different weekly or bi-monthly meetings, involving from 6 to 26 participants, were scheduled;
(3) besides the scheduled meetings an abundance of informal, unplanned meetings were held. Typically one or two participants met together;
(4) the amount of detailed information that needed to be communicated, coordinated, and negotiated required more formalized measures for the daily operation. In the following we elaborate on these measures.

Different types of coordination mechanisms were used to keep track of the integration or the state of affairs, and to schedule relations and dependencies among involved actors, tasks, and resources. Some of the mechanisms were invented in the S4000 project, some were a result of redesign of previous mechanisms, and others were adopted. Examples from the manufacturing domain were:

(1) an augmented bill of materials (ABOM) supporting the integration between mechanical design, process planning, and production;
(2) a product classification scheme used for classifying CAD models, in order to support distributed storage and retrieval of these;
(3) a Cause and Effect Diagram with the Addition of Cards (CEDAC board) supporting registration of defects, shortcomings, problems, and suggested improvements identified in the production of prototypes and the test series;

The ABOM and the classification scheme mentioned above were closely related to each other. Each new component described in the ABOM was related to its CAD model and classified by means of a look-up in the product classification scheme and by filling in the related indexation form. They were furthermore interrelated with the MRP II system (COPICS) and the CAD models database used. This is illustrated in Figure 2.

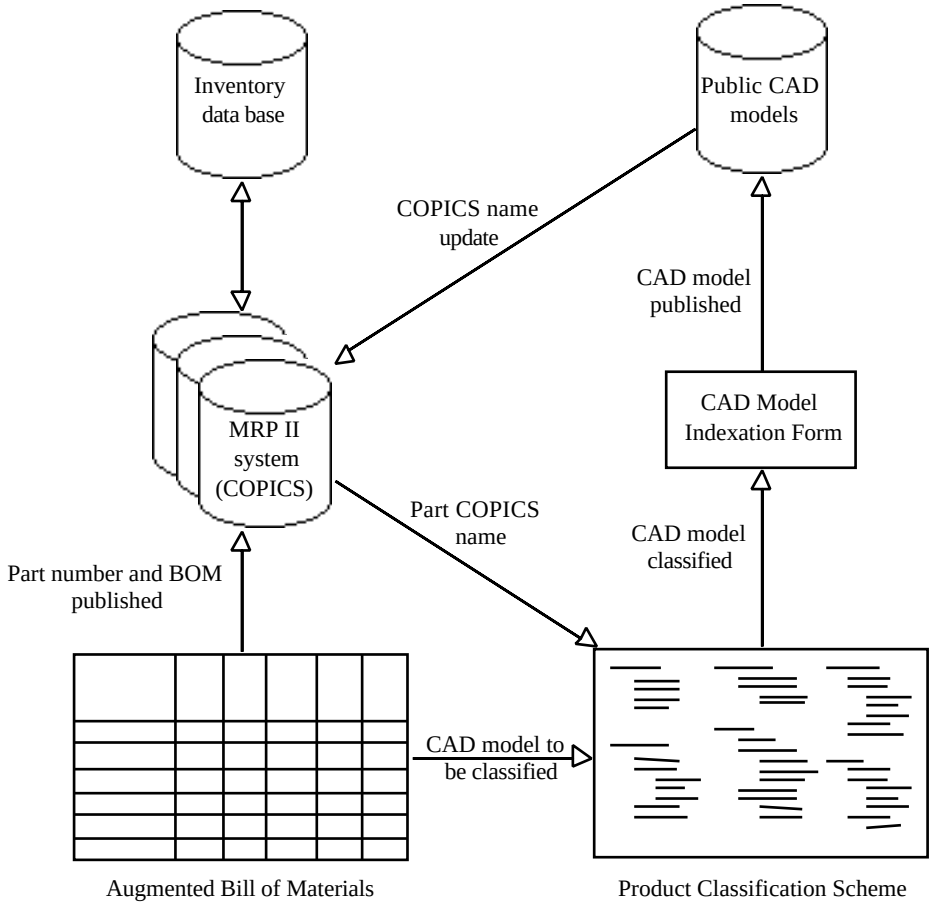

Figure 2. The ABOM and the product classification scheme were linked to each other and to the CAD repository, the MRP II system (COPICS), and the inventory database (Schmidt et al., 1995).

From the software development domain, we have the following four examples of coordination mechanisms:

(4) a fixed software design working cycle (or working rhythm) including artifacts supporting the coordination and integration of the software;
(5) a software project plan spreadsheet supporting the allocation of resources and scheduling of tasks and responsibilities;
(6) a directory structure reflecting the architecture of the software and used to support the integration of the software developed; and
(7) a bug report form coordinating activities concerning registration, diagnosis, and correction of software bugs.

These four mechanisms were also interrelated, illustrated in the following examples and shown in Figure 3: The working cycles were specified in the planning spreadsheet; an accepted bug in a bug report indicated that a task should be included in the spreadsheet; and the directory structure specified how an integration period in a specific working cycle should be started.

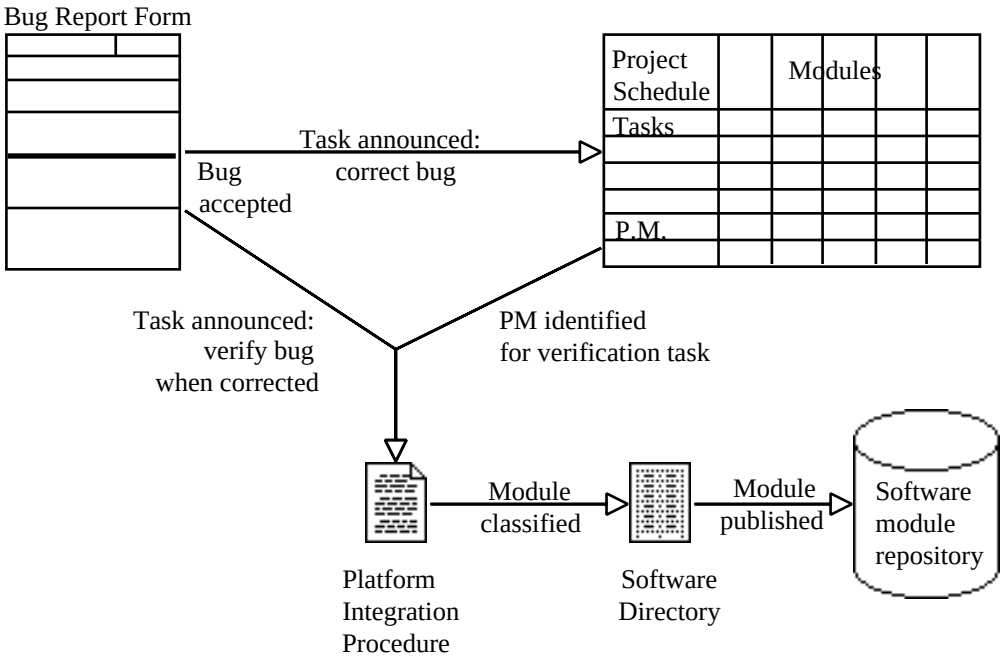

Figure 3. An illustration of how the coordination mechanisms used in software design interact. 'PM' denotes the Platform Master, i.e., the actor in charge of the integration of modules and hence verification of corrections at the end of the current working cycle (Schmidt et al., 1995).

Due to limited space we describe, in the following section four of the mechanisms listed above. The first two of these are from the manufacturing domain, the last are from software development. The selected mechanisms show a span between, on the one hand, mechanisms providing pure procedural support, and on the other hand, mechanisms providing classification structures. We have also selected mechanisms which illustrate support of both distributed work and a shared workspace. Among the four examples

there are, furthermore, three mechanisms developed at Foss Electric as well as one adopted from others.

Section 4.2 presents the Product Classification Scheme which basically is a structure allowing for distributed storage and retrieval. Section 4.3 analyzes the CEDAC Board—a classification structure embedded in a shared workspace. Section 4.4 presents the Software Design Working Cycles, primarily a procedure stipulating the division of labour in the software development process. Section 4.5 presents the Bug Report Form, a paper-based workflow system combining a form containing classification structures, and a formal routing procedure.

4.1 Product Classification Scheme

The product classification scheme provided a means for storage and retrieval of Computer Aided Design (CAD) models in a data management system. The CAD system was introduced at the same time as the S4000 project was initiated. One of the advantages of using CAD instead of a traditional paper-based system is an improved opportunity of reusing old components in new products. This approach was time-saving and, among other things, supported the use of standardized components. In order to optimize the potentials of reuse, it is, however, pertinent that the CAD specifications can be retrieved across projects. Although browsing through a file-cabinet with drawings is a time-consuming task, it is possible. When work is based on CAD models stored in a database, browsing is not a feasible strategy for reuse of existing specifications. If, however, CAD models are categorized, existing specifications can more easily be retrieved. Hence, in relation to introducing CAD, the company developed a classification scheme capturing components and units for all instruments produced (see Figure 4).

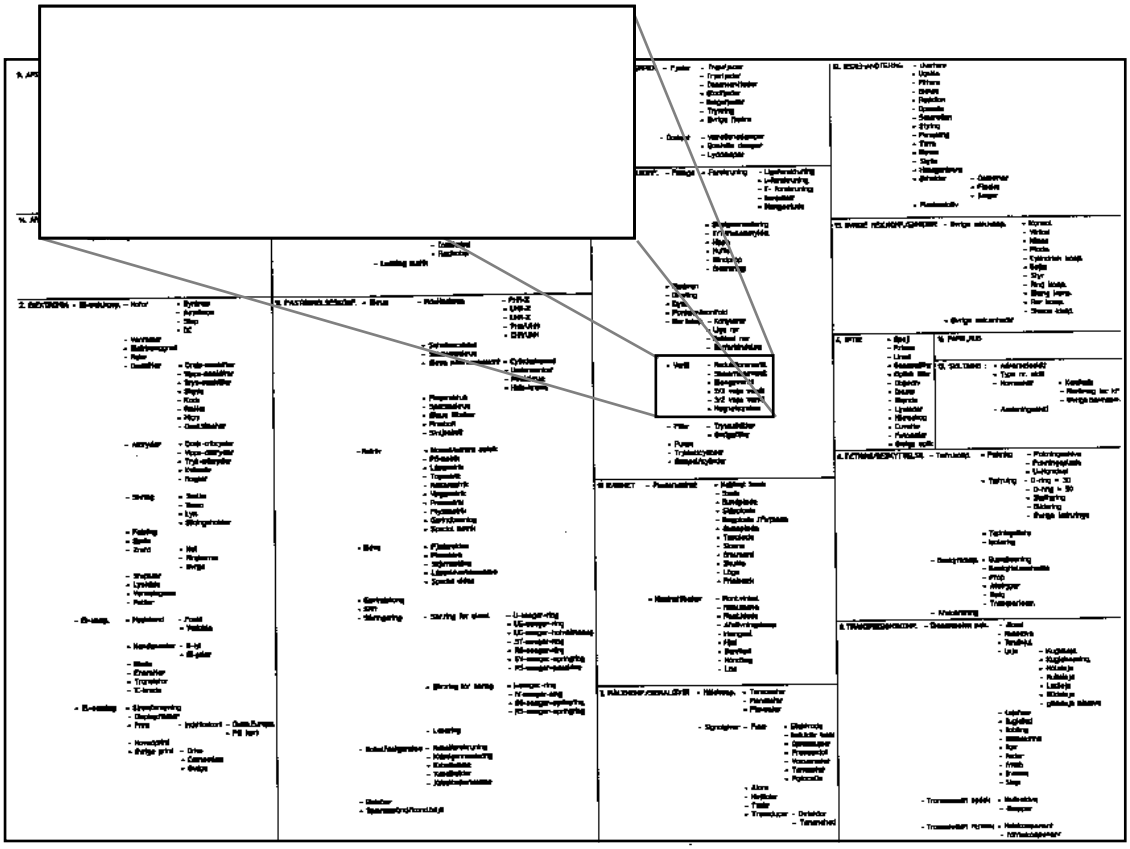

Figure 4: The product classification scheme — the different sub-categories belonging to the category of valves, which are of the class hydraulic and pneumatic components. The small fraction of the product classification scheme illustrates the different sub-categories belonging to the category of valves, which are of the class hydraulic and pneumatic components.

When a CAD model for a component had been specified, the draught-person classified it according to the classification scheme. The classification was an extra attribute in the data management system. The classification scheme was partly paper-based and partly computer supported, and its use was not stipulated in an organizational procedure, but only by conventions.

The classification was most often determined based on consulting an A3 size scheme which the draught-person had on the desk. The classification was then entered into the data management system by selecting from a list in a menu on the CAD system. Here, the categories in the classification scheme were alphabetically ordered. The paper-based scheme was a print of the classification scheme specified in the database. It was ordered in a tree-structure with classes, categories, sub-categories, and sub-sub-categories. There were 16 classes, and approximately 340 different categories, sub-categories, and sub-sub-categories. As an example, class number 5 was hydraulic and pneumatic components, which had 11 categories. One of these was valves, which has 6 sub-categories and no sub-sub-categories (see Figure 4). The categories were from time to time modified, and new categories were added in order for the scheme to represent the type and function of components specified. Changes to the scheme were results of negotiations between designers and draught-persons at designated meetings. The classes

and categories in the scheme were based on a mix of functional and geometrical properties of components and units. Some of the categories reflected the practical problems of classifying components. There was, for example, a class named *"other mechanical components and units"* containing categories such as: console, plate, cylindrical component, tube component. This gave the draught-person a means of classifying very irregular components, which otherwise would have been impossible to fit into the scheme.

The product classification scheme supported the coordination work solely by providing a conceptual structure making it possible for draught-persons and engineering designers to perform distributed storing and retrieval of CAD models. It reflected a common standard for categorization of the components and units in any instrument produced at Foss Electric. The 16 classes and about 340 categories can be viewed as the negotiated order of how an instrument could be formally described. It is enforcing a standardized format for filing and retrieving CAD models. New components and units were designed all the time, and it was often difficult for the draught-person to perform the classification. This did however, not result in constant changes to the classification scheme. Because the scheme was used by many different people, changes had to be negotiated. The main rationale behind the classification scheme was very far from the philosophy of classification systems used in libraries. It was not a primary requirement that any CAD specification stored by one person should be easily retrievable by another. The idea was that the specifications of the most commonly used components and sub-assemblies should be retrievable. These components were most often characterized by a very well defined functionality. Since the classification scheme primarily was based on functionality, these components were very easily classified. In other words, the classification scheme is based on an 80% principle, where the 20% of specifications, which were very specific components, might have be classified in such a way that they were very difficult to retrieve. The initial scheme was designed by a six-person task force who spent a couple of weeks designing it in their spare time—work causing major discussions. As in the case of the International Classification of Diseases re-interpreted by Schmidt (1994), use and management of the product classification scheme can be characterized as a struggle of carefully finding *"the appropriate level of ambiguity"*.

4.3 The CEDAC Board

The idea of the CEDAC (Cause and Effect Diagram with the Addition of Cards) System was originally developed in Japan in the seventies (Fukuda, 1989). It was, subsequently, adopted by North American and European companies. The purpose of the CEDAC System (see Figure 5) is to reduce the number of manufacturing defects through continuous improvements, by enabling people to make use of their accumulated knowledge and experience. The system was introduced at Foss Electric in 1990 after representatives from the company had visited companies in Japan. The main purpose was to register and overcome defects, shortcomings, problems, and suggestions found in production in connection with manufacturing prototypes and test series instruments.

This resulted in an improved productivity and product quality by involving the employees in solving the problems. The CEDAC System was materialized in various boards either mounted on wheels or on walls. The particular one we studied was a 1 x 2 meter steel board placed on a wall at the shop floor. When a problem was encountered or a suggestion for improvement found, a card describing it was attached to the board with a magnet. Examples on topics were: poor and insufficient drawing specifications which could make it impossible to produce the product, inappropriate tool specifications, or unsuitable process selections. Cards could be classified in the following categories:

(1) of interest;
(2) in progress;
(3) proposal not possible;
(4) being tested;
(5) successfully tested; and
(6) the test yielded a poor result.

People from the different functions participate in a weekly CEDAC meeting, i.e., mechanic and electronic designers, draught-persons, the project leader, process planners and a foreman.

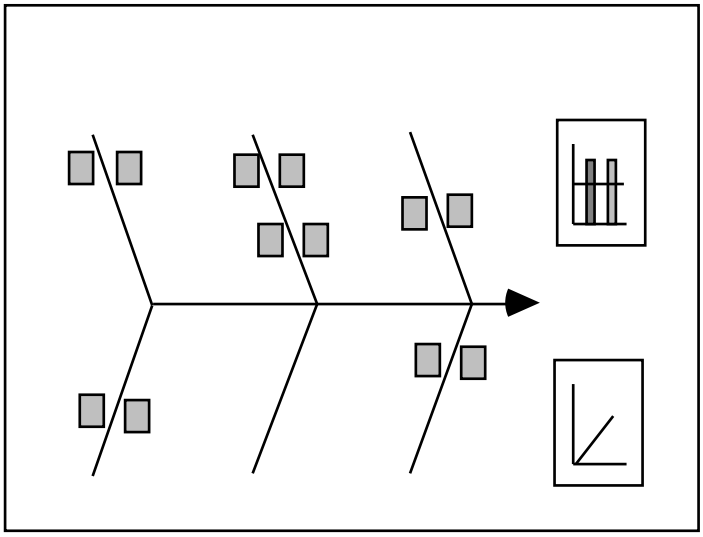

Figure 5: The CEDAC board as implemented in engineering design, process planning and production of parts for functional models and prototypes at Foss Electric.

Workers on the shop floor had since the previous meeting placed cards on the CEDAC board describing defects and suggestions. Statistics on the accumulation and processing of errors and suggestions were compiled and placed on the board. The CEDAC System ensured that defects and good ideas for improvement were collected and continuously negotiated and tested on a weekly basis. Hence, it supported the integration between the engineering design and process planning functions. The CEDAC System was not primarily meant to be an integration tool, but a tool to

maintain and improve product quality. At Foss Electric, CEDAC Systems were also implemented in the assembly department and in the quality control department.

The frequency of CEDAC meetings varied through the project. In some phases there were a large amount of errors and observations to be discussed, and in others there would only be one meeting each week. In the assembly department there could, in the early production stages, be daily meetings in order to discuss assembly problems. The system would occasionally break down, or be overruled. If, for example, weekly meetings on Thursday morning had been arranged, handling a serious error observed right after the meeting would not be postponed until the following meeting. The CEDAC procedure would be overruled and the problem would be taken care of in a separate meeting.

If we interpret the CEDAC board purely in terms of an artifact supporting coordination by stipulating and mediating aspects of coordination work, one important aspect is the references to the field of work and the work arrangement represented in the artifact. The CEDAC board was basically an artifact supporting coordination by providing an asynchronous communication channel and two conceptual structures of the classification scheme type. Had the board just been an ordinary notice-board with no restrictions as to whom placed what, it would only have provided an easier way of performing *ad hoc* modes of interaction by providing an asynchronous communication channel. The first classification scheme was used to classify notes into types of problems and suggestions, i.e., blue-print errors, mechanic design, electronic design, part defects, assembly, and other. The second classification scheme was used to classify the state of the problem or suggestion on each card. Everyone could put up a note, but only the CEDAC foreman could symbolically show the state of the solutions by classifying the card. The board without any classification schemes would only have supported cooperation by providing a shared workspace where observations would be accumulated between meetings. The CEDAC board, however, reduced complexity of coordination work by combining the two conceptual structures and a symbolic representation of the field of work on notes, and because it was accompanied by an organizational procedure stipulating the division of responsibility among the different functions involved.

4.4 Software Design Working Cycles

During the design of the software in the S4000 project, the designers realized that they had severe problems in coordinating and integrating both their activities and the software modules. According to their own account, they needed prescriptions for how to control and coordinate the process of integrating and meshing of the software in the S4000 project. The idea and concept of what they called “software platforms” was invented by the designers themselves in order to support monitoring and controlling the integration of software pieces and modules. The software platform is a concept including a number of artifacts, written procedures and conventions. Originally a software platform was just a point in time at which all software designers stopped all design activities and started integrating the modules. Later in the project artifacts and

organizational procedures were included. Two of these were the software project plan spread-sheet and the directory structure mentioned the beginning of Section 4.

The period between two software platforms—the period in which the software designers developed, coded, and tested their modules—was typically 3–6 weeks. Version 1 of the S4000 system covered approximately 15 platform periods. After a platform period, the developers spent a week integrating the software modules. During this period no designer was allowed to continue design work until all had approved the integrated software complex. When the integration was brought as far as it was considered possible, and all known problems were written down as tasks to be accomplished, the complete software complex was released as platform for the departure of all new design activities. In the latter part of the project—after having established a first running version of the total software system—the integration period was reduced to two and half days.

According to the designers, the software platform concept was considered an absolute necessity for the S4000 project. Structures and procedures stipulating the coordination of the software integration was needed, otherwise the work on the software would never have been finalized. An interesting observation was that over the last one and a half years of the project, the software group was working without a manager—decisions on what to do and how to organize work were made by the group themselves.

One of the disadvantages of the platform concept was, of course, that some designers were more or less inactive during the integration period, and there were no structures for handling major integration problems that appeared unexpectedly in the middle of a work period. This type of problem occasionally caused a need for re-scheduling the platform periods and immediate attainment of an extra platform integration period.

In each integration period one of the software designers was appointed as Platform Master for the following integration period. He was then responsible for collecting all information on changes (new development, redesign, error correction, etc.) made to the software, and for ensuring that the software was tested and corrected before it was released again. He was also responsible for updating a complex spreadsheet containing the revised plans and activities before the software was released for further development.

The platform working cycles is an interesting invention. Most of the designers were not familiar with projects involving as many software designers as the S4000 project. They were not familiar with the requirement for more coordination than could be managed on an *ad hoc* basis. They were not using CASE tools, encyclopaedia tools, version control tools, or other tools supporting the integration. Generally, one way to ensure both coherence in the project, and a common understanding of the work and the direction for progress, was to establish specific points in time, and at these points “force” the actors to establish a common basis for future activities. This simple idea was recognized as the best solution to an overwhelming problem for the software designers. The idea was adopted and refined in terms of written procedures for organizing the integration process, identification of different roles to be fulfilled, and invention of forms to support the integration test process. Thus, the working cycles supported the coordination of the software design work by establishing a formalization of the work.

They also established a forum in which designers were made mutually aware of each others work. This was implemented through the establishment of standardized structures. The procedures stated that the integration period should be finalized with a meeting at which all the designers were supposed to inform each other on relevant changes they had included in “their modules”. It was obvious that this way of informing each other was insufficient. The designers had no support for illustrating to each other the consequences of their changes. Thus, changes in a module often resulted in needs for extra *ad hoc* communication and coordination during the design and coding periods.

Despite these problems however, and despite the fact that more efficient and effective approaches could have been devised, the concept proved to be very successful. So successful that it has been developed into a company standard for organizing all projects at Foss Electric.

A central artifact supporting the working cycle processes was the software platform spreadsheet. This was a rolling project plan for the software development related to the working cycles containing information on:

(1) tasks;
(2) estimated time per module per task;
(3) responsibilities;
(4) relations between the tasks and platform periods;
(5) the total planned work hours per platform period for each software designer.

The spreadsheet supported the coordination work by providing a conceptual structure that scheduled tasks, actors, and deadlines, by relating development activities to relevant software modules and to responsible actors. The spreadsheet stipulated how progress regarding the integration of the software should be achieved, and which actors and modules needed to be involved when problems or changes in plans occurred. It furthermore, supported the actors’ awareness of the current state of affairs and future tasks. It also facilitated the frequent negotiations on allocation and reallocation of resources. The main problem with the spreadsheet was that it was only accessible to one of the designers. The others only had paper copies. This caused problems regarding negotiations among the designers, when these negotiations were based on out-dated versions of the plans.

Another artifact related to the working cycles was a directory structure developed to support the software integration process. Each of the designers then had to place their software modules in pre-specified locations in the directory structure before each working cycle (platform period). A set of software routines were implemented. These were used by the Platform Master for automatically collecting, compiling, and integrating all software modules. In order to support the use of the directory mechanism, a set of check lists, standards, and procedures were established. The directory structures and the software routines simplified the meshing of the software by providing a structure stipulating how several actors’ modules were to be integrated. Hence, the directory structure supported distributed software development by providing a classification scheme for concepts and structures in the software. This structure established a common standardized conceptualization of the software architecture which all designers needed to relate to when communicating and coordinating their activities.

4.7 The Bug Report Form

In the early designing, coding, and testing phases, there was a predominant lack of overview and coordination in the S4000 project. The problem was characterized by one of the software designers in the following way:

> "With the number of developers involved, it is extremely important that all bugs are registered, otherwise they just 'disappear' [..] An important derived product then is a list of problems reported as fixed but not yet tested. Based on the lists and the problem descriptions the platform master can check and then report the problem as being corrected. Originally the intention was to produce statistics of the number of known-but-not-yet-fixed problems and use this as a management tool. But we realized that as a management tool this can only be used if you have a stable product. We didn't!"

The form (see figure 6) and the list were refined into a tool for registering bugs, diagnosing and prioritizing the bugs, coordinating the correction tasks, and verifying the corrections reported.

A bug report form could be filled in by everybody involved in testing the software, e.g. software designers, other designers, people from quality assurance, and marketing people. The originator described and classified the problem. A team of three software designers (called the spec-team) then added information about affected modules, responsible designer, platform period, and an importance priority (as viewed from a software reliability perspective). All forms were filed in a central bug forms file using the categories:

(1) non-corrected catastrophe;
(2) non-corrected semi-serious problems;
(3) non-corrected cosmetic problems;
(4) postponed;
(5) rejected;
(6) corrected but not yet tested;
(7) corrected problems.

Forms were successively re-classified as a result of, for example: Decisions made by the spec-team; messages from designers concerning specific problems; results from the platform integration.

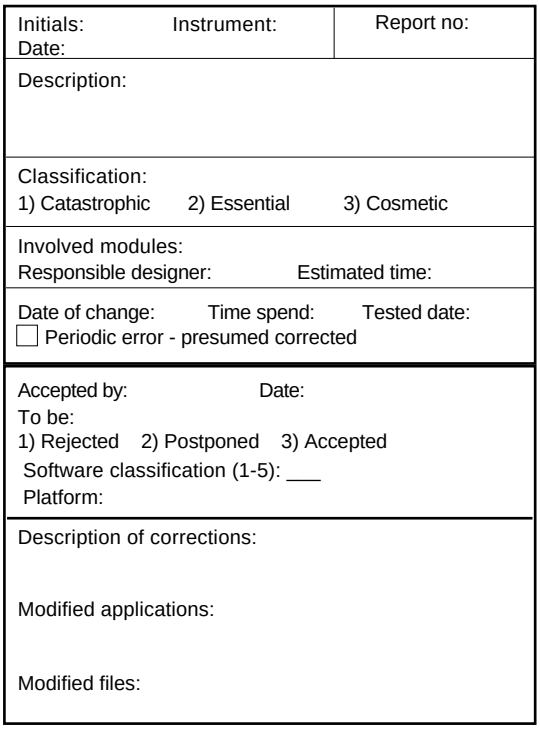

Figure 6: A translated version of the bug report form and the overall procedure for registering, diagnosing, and correcting bugs. CFM means 'central file manager' and PM is 'platform master'. The form is a sheet of A4 paper printed on both sides. The numbers indicate who were supposed to fill in which information in the form.

The list of unresolved problems was continuously produced and accessed by the software designers. Each designer was responsible for fixing the problems (handling bug forms) and reporting back to the platform master. The flow of forms, acknowledgements, OK-reports, etc. were very complicated. A designer described it in a state-transition diagram containing more that 15 states. A thorough description of the bug report form and its use are provided in Carstensen (1996). A simple model of how the forms flowed among the actors is shown in figure 7. The flow structure were successively negotiated and changed due to, for example, involvement of new designers, recognition of mis-classification of bugs, rejections of responsibility for a specific problem.

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

The bug report form was used to collect, classify, and manage errors and suggestions. It mediated coordination information and stipulated coordination by means of conceptualizations of tasks, actors, software modules, and responsibilities. The main purpose of the bug report forms, the central file, the problems list, and the procedures for classifying, correcting, and reporting on the problems, was three-fold: Firstly, it ensured that all problems were registered and filed. Secondly, it established and exhibited a clear and visible organization of designers' responsibilities. Thirdly, it clearly stipulated how a problem was diagnosed, how the correction responsibility was

delegated, and how a problem was reported corrected. Furthermore, the problems list provided designers with an awareness of the state of affairs in the total software system. It provided designers with visibility of activities in modules of which they were not responsible, but with which their modules interacted.

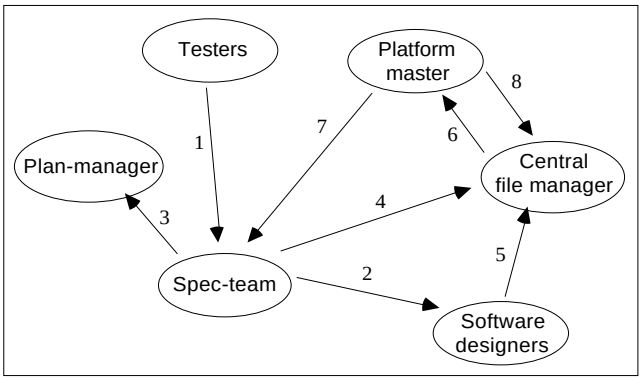

Figure 7: The different roles involved in the coordination of software testing and correction in the S4000 project, and the stipulated flow of bug report forms between them. The arrow numbers indicates the sequence when following the standard work flow (adapted from Carstensen et al., 1995).

The invention and use of the bug report form and the problems list reduced the complexity of the coordination activities concerning bug registration and correction by allowing distributed registration and classification of software errors. The standardized format and the classification established a common language for reporting on bugs and for classifying the problems, both from a usage perspective and from a software design perspective. This made it possible to distribute the test activities, since the need for *ad hoc* communication and coordination was reduced. This was done through introducing a more formalized paper-based work-flow system. The structure for classifying problems proved, however, to be insufficient in a number of situations. Having only three categories of problems, and not being able to relate the classification to certain aspects of the product (e.g., usability or maintenance), resulted in further discussions and negotiations on how to classify and interpret a classification of a bug.

The bug form system made all known problems visible to software designers and others involved in the testing work. However, the designers found it increasingly difficult to obtain an overview since the binder eventually contained several hundred bug forms.

The last and perhaps most important feature of the bug handling mechanism was that it clearly stipulated the flow of the reporting, diagnosing, correcting, and verifying activities. When an activity was finalized the following one could be initiated more or less automatically, and by means of the central file it was possible to trace each bug reported and to conceive its current status.

5. Discussion

Based on the findings presented above, several motives for inventing, designing, and using mechanisms supporting the coordination of work can be derived. When the number of mutually interdependent actors involved in a project exceeds the limit of a few, they need to examine the state of affairs in the field of work. This is, however, impossible to do by means of *ad hoc* modes of interaction only. More formalized structures, procedures and artifacts are required. Activities have to be conducted, separated in time and space. This requires standardized structures, classifications, and conceptualizations reflecting relevant structures both in the field of work, the work arrangement, and the wider context. Since not all interaction can be conducted synchronously, human ability to correct communication errors and misunderstandings "on the fly" is not sufficient. A commonly understood and more clearly specified "language" is needed. Furthermore, when work requires the management of many intertwined and interdependent activities, the complexity of meshing and coordinating these activities increases tremendously. In such situations the field study clearly illustrated the need for support of the structuring and control of how the interrelated activities should be meshed.

The motives mentioned above were all exemplified in the field study. In most cases the first solution when a problem was recognized was to increase the use of *ad hoc* coordination, i.e., have more formal and informal communication and meetings. In many situations this appeared insufficient and ineffective. Thus, formal structures and mechanisms of different kinds were invented, introduced, and used. An overall result from this study is, thus, that when confronted with an abundance of detailed decisions and activities that need to be coordinated, organizations invent and adopt mechanisms that partly mediate and stipulate the coordination of the work. All the mechanisms of this type observed in the field study were characterized by increased rigidity and were more prescriptive than those they replaced. For example, the bug form system stipulated the distributed activities to take place where a bug was identified, this replaced weekly meetings. The directory structure and automatic linking procedures replaced an agreement among the software designers on how the software components should be integrated.

The following summarizes the common characteristics of the mechanisms described in the previous section. We identify the overall function and purpose of the artifacts and their concomitant conventions and procedures. Our approach aims at providing input for a process using the analysis results as a background for designing computer-based support. Some implications for the design of computer-based coordination mechanisms are discussed here. More elaborate discussions on requirements for computer support can be found in Carstensen et al.(1995) or Carstensen and Sørensen (1994).

The coordination mechanisms described in Section 4 supports the coordination of work in several ways.

First, they support the involved actors in examining the state of affairs of both the field of work and the work arrangement. Although the central bug form binder, for example, provided poor browsing facilities, it still enabled all designers and testers to

browse information on all registered bugs by classification. The CEDAC board offered a board visible to all actors and including two classification schemes for classifying problems and their state. The board made it easier to view state of affairs at a glance. The project plan spreadsheet provided facilities for examining the software architecture and for relating this to design and correction tasks. It, furthermore, provided an overview which stipulated relationships between tasks, responsible designers and platform periods. Computer based coordination support will typically have to include functionality for: Viewing the state of affairs; reporting changes to the state of affairs; starting and stopping procedures and processes; and simulating the consequences of running a specific procedure. The structures offered by the computer based mechanisms will probably have to be quite similar to the structures embedded in the existing artifacts, i.e., conceptualizations of central structures in the work arrangement and the field of work.

Second, the mechanisms supported conducting activities in parallel and separated in both time and space. The conceptual structures and a negotiated order of how to classify and interrelate conceptual structures, provided the actors with an opportunity to work with the same structures and resources concurrently, decentralized and distributed in time and space. The bug report, its embedded classification structure, and the procedure for its use, supported distributed software testing and correction. The product classification scheme made it possible for actors to perform distributed storage and retrieval of CAD models in a common data management system. In a similar fashion, the software directory structure enabled the software designers to work in parallel without constant interaction and negotiation on the allocation of facilities in the software modules.

Third, the mechanisms provided standardized structures with shared meanings, which supported the communication and negotiation of decisions, allocation of resources, etc. The conceptual structures reflected the central objects of articulation, e.g. the involved actors, the tasks to be accomplished, and the components of the instrument. Furthermore, the artifacts provided a classification used for categorization of the objects of coordination work. The classification reflected a negotiated order of how to formally characterize, for example, the instrument, the tasks and their relations to the actors, deadlines. The product classification scheme, the classification structures in the bug report form, and the conceptualization of the software architecture in the bug report form are all examples of such standardized descriptions.

Fourth, some of the mechanisms supported the stipulation and control of interrelated activities. The most obvious example is the software design working cycles. The platform concept clearly specified which roles and activities were needed when integrating the software components in the end of each platform period. It, furthermore, prescribed how the activities should be conducted, and who was responsible. The importance of this stipulation of the work was highlighted by several designers. They stated that if they had not established the working cycles, the software project would have failed. They would simply not have been able to control the integration process. Another example is the procedures related to the bug report form. These clearly specified the flow each process of registering, diagnosing, correcting, and verifying the software bugs should follow. This procedure eliminated many problems facing the

software designers when handling software bugs. The directory structure and the related procedures, furthermore, stipulated how the software modules were to be integrated.

Fifth, the mechanisms supported modifications to and refinements of the mechanisms—changes in the way they appeared and were used. The procedures, structures, schemes, and embedded classification categories were used by many actors and had to be changed frequently due to the emergence of new problems, such as: Redesign of the instrument; redesign of the production planning and process; changes in the requirements for the software; and changes in the organization of the development team. Examples of such changes are new dimensions added to the product classification scheme, new modules added to the software architecture and thereby to the directory structure, changes to or addition of actors and software structures in the project plan spreadsheet. The procedures stipulating how the platform cycles should be organized, and the definition of the different roles involved, were also refined. Hence, computer-based coordination mechanisms should provide some degree of local control and the embedded structures and protocols must be visible to the actors. Actors must be able to asses the status of the mechanism in order to decide on the further course of action.

In general, we can consider the described mechanisms as examples of sets of conventions and prescribed procedures with related symbolic artifacts with a standardized format that stipulates and mediates the coordination of the distributed activities, i.e., as examples of what Schmidt and Simone (1996) define as coordination mechanisms. All of the artifacts contained structures that can be described as conceptualizations of structures from the field of work and the work arrangement.

6. Conclusion

This paper is based on a field study in a large scale manufacturing design project involving many, partly distributed, actors with different areas of competence, over a long period of time. The people involved were mutually interdependent, the product consisted of a multitude of interacting parts, and the project took place in an environment characterized by both uncertainty and dynamism.

We have argued that one of the primary contributors to a high degree of complexity in the project was the large number of people with different skills involved in a concurrent engineering project. Hence, the coordination work they needed to engage in was a main source of complexity. We found support for the hypothesis that large scale manufacturing projects invent and adopt mechanisms stipulating and mediating coordination work. We also found that the actors tend to increase the level of formalization embedded in the mechanisms when the complexity of the coordination work increases. Examples of coordination mechanisms used in the project were presented and analyzed based on the concept of Coordination Mechanisms (Schmidt and Simone, 1996). Although the study addressed manufacturing we have reasons to expect the hypothesis to be valid in other domains characterized by either many distributed actors with different areas of competence, a large number of intertwined activities, or work carried out over a long time-span.

Due to the limited space, neither the project and coordination mechanisms studied, nor aspects of computerization of the mechanisms are discussed in great detail. The aim has been, by way of examples, to provide ideas for a sound starting point for further research on how to provide computer support for coordination work in manufacturing projects.

We fully realise the human capability to communicate, coordinate, and negotiate by means of discussions, meetings, and memos. Our claim is merely that in complex situations there are a need for more formalized mechanisms mediating and stipulating aspects of the coordination work in order to reduce the complexity to a manageable level. Although we claim a need for more formalized measures, we realise that work situations are situated and can be enormously contingent (Suchman, 1987). It is, therefore, an essential requirements for coordination mechanisms that they provide a high degree of local control to the individual actors. We propose to consider developing such coordination mechanisms based on computers. Although we have commenced requirement specification work and development of preliminary prototypes we do realise that only a point of departure has been established. Much further work is required in order to develop a more coherent understanding of the cooperative aspects of complex work and in formulating constructive measures are required.

Acknowledgements

First of all, this research could not have been conducted without the invaluable help of numerous people at Foss Electric A/S, Hillerød, Denmark. Thanks for letting us pry in your work! We would especially like to thank Marianne Malmstedt, Ole Pflug, Jørn Ørskov, Steen Kjær Andersen, and Anne Poulsen for helping us coordinating the field study. Thanks to Henrik Borstrøm who participated in the project through three years. We have had many fruitful discussions with Kjeld Schmidt during the analysis of our findings. Also a big thanks to Maxine Robertson for carefully proof-reading the manuscript. Three anonymous reviewers provided us with constructive comments and suggestions. The research documented in this paper has partially been funded by the Esprit BRA 6225 COMIC project, and the CODEM project sponsored by the Danish Technical Research Council. This work has also been supported by the Swedish Transport & Communications Research Board (Kommunikationsforskningsberedningen) through its grant to the ”Internet project”. All errors in this paper naturally remain the responsibility of the authors.

Bibliography

Agostini, Alessandra, Giorgio De Michelis, Stefano Patriarca, and Renata Tinini (1994): A Prototype of an Integrated Coordination Support System. *CSCW*, vol. 2, no. 4.

Anderson, Bob, Graham Button, and Wes Sharrock (1993): Supporting The Design Process Within An Organisational Context. In *ECSCW ’93. Proceedings of the Third European Conference on Computer-Supported Cooperative Work, Milan*, ed. Giorgio De Michelis, Carla Simone, and Kjeld Schmidt. Kluwer Academic Publishers, pp. 47–59.

Bentley, R., J. A. Hughes, D. Randall, T. Rodden, P. Sawyer, D. Shapiro, and I. Sommerville (1992): Ethnographically-informed systems design for air traffic control. In *CSCW '92. Proceedings of the Conference on Computer-Supported Cooperative Work, Toronto, Canada, October 31 to November 4, 1992*, ed. Jon Turner and Robert Kraut. New York: ACM Press, pp. 123-129.

Bucciarelli, Louis L. (1984): Reflective practice in engineering design. *Design Studies*, vol. 5, no. 3, July, pp. 185-190.

Carstensen, Peter and Kjeld Schmidt (1993): Work Analysis-Perspectives on and Requirements for a Methodology. In *Human-Computer Interaction: Applications and Case Studies*, ed. M. J. Smith and G. Salvendy. Amsterdam: Elsevir, pp. 575-580.

Carstensen, Peter and Carsten Sørensen (1994): Requirements for a Computational Mechanism of Interaction: An example. In *A Notation for Computational Mechanisms of Interaction*, ed. Carla Simone and Kjeld Schmidt. Lancaster, England: University of Lancaster, pp. 33-80.

Carstensen, Peter H. (1996): *Computer Supported Coordination.* Writings in Computer Science (No. 61). Roskilde, Denmark: Department of Computer Science, Roskilde University.

Carstensen, Peter H., Carsten Sørensen, and Tuomo Tuikka (1995): Let's talk about bugs! *Scandinavian Journal of Information Systems*, vol. 7, no. 1, pp. 33-53.

Dery, David and Theodore J. Mock (1985): Information Support Systems for Problem Solving. *Decision Support Systems*, vol. 1, pp. 103-109.

Dourish, Paul (1993): Culture and Control in Media Space. In *ECSCW '93. Proceedings of the Third European Conference on Computer-Supported Cooperative Work, 13-17 September 1993, Milan, Italy*, ed. Giorgio De Michelis, Carla Simone, and Kjeld Schmidt. Dordrecht: Kluwer Academic Publishers, pp. 125-137.

Fillipi, Geneviève and Jacques Theureau (1993): Analyzing Cooperative Work in an Urban Traffic Control Room for the Design of a Coordination Support System. In *ECSCW '93. Proceedings of the Third European Conference on Computer-Supported Cooperative Work, 13-17 September 1993, Milan, Italy*, ed. Giorgio De Michelis, Carla Simone, and Kjeld Schmidt. Dordrecht: Kluwer Academic Publishers, pp. 171-186.

Fitzpatrick, Geraldine, Wiiliam J. Tolone, and Simon Kaplan (1995): Work, Locales and Distributed Social Worlds. In *Proceedings of the Fourth European Conference on Computer Supported Cooperative Work - ECSCW'95, 10-14 September, 1995, Stockholm, Sweden*, ed. Hans Marmolin, Yngve Sundblad, and Kjeld Schmidt. Kluwer Academic Publishers, pp. 1-17.

Flores, Fernando, Michael Graves, Brad Hartfield, and Terry Winograd (1988): Computer Systems and the Design of Organizational Interaction. *TOIS*, vol. 6, no. 2, April, pp. 153-172.

Fukuda, Ryuji (1989): *CEDAC - A Tool for Continuous Systematic Improvement.* Cambridge, Massachusetts, USA: Productivity Press.

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

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

Heath, Christian and Paul Luff (1992a): Collaboration and Control. Crisis Management and Multimedia Technology in London Underground Control Rooms. *CSCW*, vol. 1, no. 1-2, pp. 69-94.

Heath, Christian and Paul Luff (1992b): Media Space and Communicative Asymmetries: Preliminary Observations of Video Mediated Interaction. *Journal of Human-Computer Interaction.*

Helander, Martin and Mitsou Nagamachi, ed. (1992): *Design for Manufacturability — A Systems Approach to Concurrent Engineering and Ergonomics*. London: Taylor & Francis.

Holt, Anatol (1988): Diplans: A New Language for the Study and Implementation of Coordination. *TOIS*, vol. 6, no. 2, April, pp. 109-125.

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

Ishii, Hiroshi, Kazuho Arita, and Takashi Yagi (1993): Beyond Videophones: TeamWorkStation-2 for Narrowband ISDN. In *ECSCW '93. Proceedings of the Third European Conference on Computer-Supported Cooperative Work, 13-17 September 1993, Milan, Italy*, ed. Giorgio De Michelis, Carla Simone, and Kjeld Schmidt. Dordrecht: Kluwer Academic Publishers, pp. 325-340.

Kaplan, Simon M., William J. Tolone, Douglas P. Bogia, and Celsina Bignoli (1992): Flexible, Active Support for Collaborative Work with Conversation Builder. In *CSCW '92. Proceedings of the Conference on Computer-Supported Cooperative Work, Toronto, Canada, October 31 to November 4, 1992*, ed. Jon Turner and Robert Kraut. New York: ACM Press, pp. 378-385.

Keyser, Véronique De (1992): Why field studies? In *Design for Manufacturability — A Systems Approach to Concurrent Engineering and Ergonomics*, ed. Martin Helander and Mitsou Nagamachi. London: Taylor & Francis, pp. 305-316.

Kraut, Robert E. and Lynn A. Streeter (1995): Coordination in Software Development. *Communications of the ACM*, vol. 38, no. 3, pp. 69-81.

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

Mason, Richard O. (1989): MIS Experiments: A Pragmatic Perspective. In *The Information Systems Research Challenge: Experimental Research Methods*, ed. Izak Benbasat, vol. 2. Boston Massachusetts: Harvard Business School Research Colloquium, Harvard Business School, pp. 3-20.

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

Patton, M.Q. (1980): *Qualitative Evaluation Methods*. USA: Sage Publications.

Schmidt, Kjeld (1994): *Modes and Mechanisms of Interaction in Cooperative Work*. Risø National Laboratory.

Schmidt, Kjeld and Peter Carstensen (1990): *Arbejdsanalyse. Teori og praksis [Work Analysis. Theory and Practice]*. Risø National Laboratory.

Schmidt, Kjeld and Carla Simone (1996): Coordination mechanisms: Towards a conceptual foundation of CSCW Systems Design. *Computer Supported Cooperative Work. The Journal of Collaborative Computing*, vol. 5, no. 2-3.

Schmidt, Kjeld, Carla Simone, Peter Carstensen, Betty Hewitt, and Carsten Sørensen (1993): Computational Mechanisms of Interaction: Notations and Facilities. In *Computational Mechanisms of Interaction for CSCW*, ed. Carla Simone and Kjeld Schmidt. Lancaster, England: University of Lancaster, pp. 109-164.

Schmidt, Kjeld, Carla Simone, Monica Divitini, Peter Carstensen, and Carsten Sørensen (1995): *A 'contract sociale' for CSCW systems: Supporting interoperability of computational coordination mechanisms*. Centre for Cognitive Informatics, Roskilde University.

Siemieniuch, Carys (1992): Design to product — A prototype of a system to enable design for manufacturability. In *Design for Manufacturability—A Systems Approach to Concurrent Engineering and Ergonomics*, ed. Martin Helander and Mitsou Nagamachi. London: Taylor & Francis, pp. 35-54.

Simon, Herbert A. (1973): The Structure of Ill Structured Problems. *Artificial Intelligence*, vol. 4, pp. 181-201.

Simon, Herbert A. (1981): *The Sciences of the Artificial*. Second edition; First edition 1969. Cambridge, Mass.: The MIT 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.

Suchman, Lucy (1994): Do Categories Have Politics? The language/action perspective reconsidered. *Computer Supported Cooperative Work*, vol. 2, no. 3, pp. 177-190.

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

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