Toward a Cooperative Experimental System Development Approach*

Kaj Grønbæk
Morten Kyng
Preben Mogensen

Computer Science Department, University of Aarhus
Ny Munkegade, Bldg. 540, DK-8000 Aarhus C, Denmark
Tel: +45 8942 3261, FAX: +45 8942 3255
E-mail: {kgronbak,mkyng,preben}@daimi.aau.dk

This chapter represents a step towards the establishment of a new system development approach, called Cooperative Experimental System Development (CESD). CESD seeks to overcome a number of limitations in existing approaches: specification oriented methods usually assume that system design can be based solely on observation and detached reflection; prototyping methods often have a narrow focus on the technical construction of various kinds of prototypes; Participatory Design techniques—including the Scandinavian Cooperative Design (CD) approaches—seldom go beyond the early analysis/design activities of development projects. In contrast, the CESD approach is characterized by its focus on: active user involvement throughout the *entire* development process; prototyping experiments *closely coupled* to work-situations and use-scenarios; *transforming results* from early cooperative analysis/design to targeted object oriented design, specification, and realisation; and design for *tailorability*. The emerging CESD approach is based on several years of experience in applying cooperative analysis and design techniques in projects developing general, tailorable software products. The CESD approach is, however, not limited to this development context, it may be applied for in-house or contract development as well. In system development, particularly in cooperative and experimental system development, we argue that it is necessary to analytically separate the abstract concerns, e.g. analysis, design, and realisation from concrete activities and techniques. Thus we introduce a CESD model which provides a framework for handling this separation and at the same time makes it possible to identify and discuss the rich variety of relationships among concrete activities and the main concerns.

INTRODUCTION

Developing useful systems for the workplace is a difficult endeavour, and today it is obvious that our old ways of doing are inadequate [# Insert ref t. LM/MK intro]. Over the last decade several approaches have gained widespread attention by addressing the more serious problems: Prototyping overcomes some of the problems of specification oriented methods which usually assume that system design can be based solely on observation and detached reflection. Object-orientation can make the relations between a system and the work it is to support more explicit and thus simpler to maintain as the context of work changes. Participatory Design, in its many forms, gives voice to the end-users during design when major decisions can be made and changed at a reasonable cost, and not only when the system is to be installed. Several Participatory design approaches, such as the Scandinavian tradition of cooperative design, combines user involvement with prototyping or mock-up techniques [7, 20], but there is still some way to go before more comprehensive approaches emerge. Thus Participatory Design

* From Kyng & Mathiassen (1997). Computers and Design in Context. pp. 201-238.

techniques seldomly go beyond the early analysis/design activities of development projects and Prototyping methods often have a narrow focus on the technical construction of various kinds of prototypes.

In this chapter we outline a Cooperative Experimental System Development (CESD) approach, which seeks to overcome such limitations by supporting: active user involvement throughout the entire development process; prototyping experiments closely coupled to work-situations and use-scenarios; transforming results from early cooperative analysis/design to targeted object oriented design, specification, and realisation; and design for tailorability.

However, in order to present the new, more comprehensive CESD approach we need a better framework for understanding system development. Traditionally system development methods are described by activity flow models, listing analysis, design and implementation activities, typically with some iterations among them. In general, however, such models serve better as a means to get a project manager's view of project activities than as a designer/developers guide on how to understand and accomplish the cooperative development activities in a project. Thus, many Participatory Design researchers, including people within the Scandinavian tradition of cooperative design, have presented their work by telling the story of specific projects and the concrete activities carried out in them [2, 3, 22, 30, 41]. When such accounts are supplemented with presentations of specific techniques (e.g. Mock-up or prototyping techniques [5, 7, 15]) this forms a good basis for designers who want to do Participatory Design early in their projects. However, with our presentation of CESD we want to do more. We want to help people understanding whole projects. As a first step in doing so we show how the traditional concerns of a system development project—like analysis, design and implementation—relates to active user involvement and how such concerns and participation may be catered to throughout the activities of a system development project.

In the rest of this chapter we first outline a set of conceptual dimensions relevant to the understanding of complex development projects and the methods to carry them out such as CESD. Then we briefly present the cases we use throughout to exemplify different aspects of projects following a CESD approach (further details on the projects forming the basis for the development of CESD may be found in [22, 24, 32]). Next we give a presentation and discussion of the concerns of CESD. Finally we present a detailed discussion of the activities in a specific CESD project, and how they relate to the concerns and context of Cooperative Experimental System Development. We use a hypermedia development project which has took place as part of two closely coupled ESPRIT projects EuroCoOp and EuroCODE. EuroCoOp/EuroCODE are large EU Esprit projects involving research institutions as well as industrial partners. The overall project assignment was to develop generic CSCW applications, i.e. packaged software development, based on the user needs in large engineering companies. Great Belt A.S. (GB), a company supervising the construction of the bridge/tunnel across the Great Belt in Denmark, functioned as a prototypical user organization. Thus both projects emphasised strong cooperation between developers and practitioners from Great Belt A.S.

CONCEPTUAL DIMENSIONS

In our presentation and discussion of complex system development in general and of CESD in particular we use four different dimensions: Concerns, Activities, Involved domains, and Project assignment. Below each of these four dimensions is presented briefly.

Concerns

At an abstract level, the category of concerns[1] captures what a system development project is all about: *Management* is directed towards the project itself as a cooperative process, how it is

[1] Our original inspiration to develop the category of concerns owes much to the presentation in [1, 34]. Mathiassen denotes the category Functions. We chose the word concerns – as opposed to functions – since the

established and sustained as work progresses and conditions changes. *Analysis* focuses on the need to understand constraints and potentials in the users' practice with respect to technological possibilities. *Design* focuses on the creation and shaping of visions of technology in use; and *realisation* focuses on the realisation of the visions in technological artefacts and organizational changes. Finally the category of *computer supported work* focuses on the ongoing use and adaptation of computer systems and the work they support.

Activities

Activities are what goes on in a project, the concrete actions and situations: meetings, workshops, implementation of a prototype etc. In papers on Cooperative Design, focus has often been on activities and their abstract representation as techniques such as Future Workshops, Mock-up techniques and Cooperative Prototyping [6, 19, 24, 32, 36]. Activities and techniques are well suited for presentations of cases, for describing what has been done as well as "how to". However, they are less useful in conveying an understanding of "why and when".

Involved domains

Development projects are about change, and to facilitate understanding of change as a project progresses we look at those domains of the project and its context towards which the activities are directed: the developers practice, practice in the user organization, technology, and visions of technology in use. The practice of the developers is the basis for the work in a development project and the users' practice is the substrate for the new, emerging ways of working. The technology, e.g. the software, is traditionally viewed as the result of a development project. We consider technology as the embodiment of visions not only of technology, but of technology in use. The visions are recorded, e.g. in descriptions of functionality and scenarios of future use. Development of technology and visions goes hand in hand with interventions into the practice of the user organization, these interventions are documented for instance in descriptions of current use and computer support, typical and problematic situations, and changes on organization of work.

Project assignment

The project assignment, the task as it is understood by the participants, gives direction to the concerns Three idealised types of project assignments are often distinguished: in-house development, custom/contract development, and packaged software/product development [26]. Below we use these three together with that of development in a research setting. As claimed in the introduction, we find that CESD applies to them all. In the following we mainly consider two types of CESD assignments: custom development, where an organization, the customer, hires a development company to develop a system with them; and packaged software development, where a development company involves itself with one or more user organizations to develop a software package.

Instead of an activity flow model

Next we present the relations between the different dimensions (see Figure 1). We need to be able to discuss the relationships between concerns and activities in cooperative system development projects. Examples are: relations between analysis, design and realisation on the one hand, and on the other activities uncovering important aspects of users current work practice, activities relating prototypes to the users' future work practice through use scenarios, and developing and maintaining an object-oriented model of the future system in alignment with results of prototyping activities.

---

word concerns relate directly to the interests of the people involved. The main difference, however, is that functions, in [1, 34], are introduced as abstracted actions and thus map directly to actions.

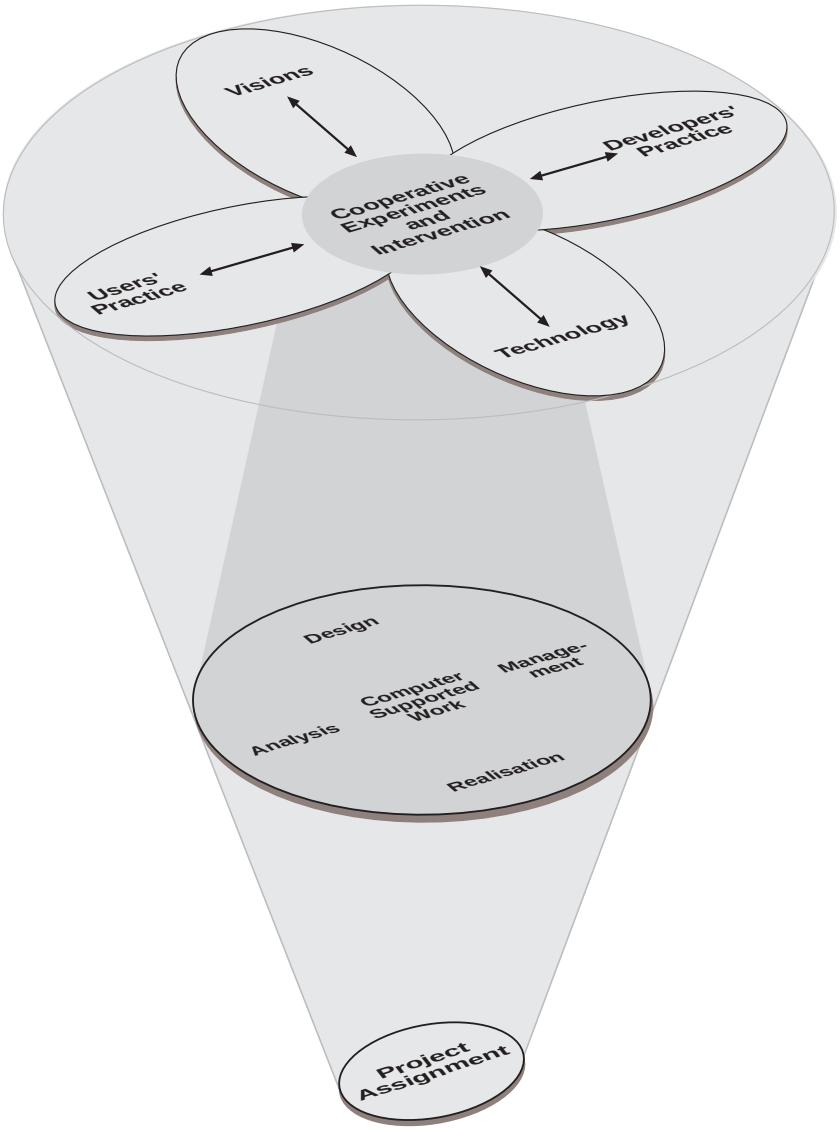

Figure 1: Instead of a project activity model

In the center of the top layer of Figure 1 are the project *activities*. As in Cooperative Design [19], the important, distinguishing CESD activities are workshop based cooperative experiments and interventions. The activities unfold in the context of the involved domains: the practice of the developers and of the users, the technology and the visions of technology in use.

In the middle layer, the *concerns* are shown. As indicated by the grey cone, concerns are realised through activities. One activity typically contributes to more than one concern, but usually an activity has one concern as its main focus.

Figure 1 also illustrates that project management is mainly—but only mainly—directed towards the developers practice, analysis towards users' practice, design towards visions of technology in use and realisation towards technology.

Finally, the project *assignment*, the task as it is understood by the project participants, gives direction to the concerns.

Distinguishing activities and concerns

Two main features distinguish the above model: - the analytical separation of activities and concerns, acknowledging that an activity may contribute to several concerns and - the

contextualization of the activities, emphasising the relations with the practice of the users and the other involved domains.

In most presentations of system development methods the category of concerns is merged with that of activities and/or abstracted activities in terms of techniques. Thus, for example, analysis is identified with one or more techniques and in a concrete project with specific activities. In addition these concerns are usually associated with a time-sequence of so-called phases separated by detailed documents. Thus it is often the expectation that e.g. analysis is finished and fully documented before design is begun. However, it is a main point in our CESD model that concrete activities in a system development project contribute to several concerns and vice versa—i.e. any one concern is realized through a number of activities. Thus analysis cannot in general be 'finished' before design is begun, simply because some of the activities carried out to cater for the design concern will also contribute to the analysis concern.

A well-known example of a method identifying concerns and activities is the waterfall model, based on detailed documents and simple feedback loops between neighbouring phases [40]. Several authors have subsequently addressed different problematic aspects of the waterfall model. Focus has been on problems with sequence and simplistic feedback loops, e.g. on the impossibility of finishing analysis before beginning realisation. The answers have been circles or spirals of iteration covering most phases several times, and one of the well-known examples of models supporting such iteration is the spiral model of Boehm [8]. However, adding iteration in order to cope with complex situations where the waterfall model is inadequate, do not address what we consider to be the source of the difficulties with current models: that they ignore that the same activity may contribute to more than one concern—as when a workshop where prototypes are tried out by the users both improves our understanding of the users' practice (analysis) and produce ideas for improving the prototype (design).

The recent interest in object-orientation—which we share—seems to have had a rather negative impact on the interest in investigating the relations between on the one hand concerns such as analysis, design and realisation and on the other activities and techniques. Current discussions on object-oriented methods usually do not distinguish activities and concerns, and they are based on a rather linear progression through analysis, via design and realisation to use (e.g. [10]), with only local iterations within the main activities. In very simple situations, such a view of development projects suffices, but for the kind of projects that we are interested in it is too simplistic to be useful in understanding what is going on.

In the section "The Concerns of CESD" we discuss the different concerns one by one to better understand each concern and how it may be catered to in a CESD project. Then in the last section we use an activity/concern matrix to present and discuss how the different activities of a CESD project over time contribute to the different concerns.

Differences to other participatory and experimental models

When we turn to the other distinguishing feature of the model in Figure 1, the contextualization of project activities, we note that this aspect certainly isn't new to Participatory Design. In a way the interaction with the practice of the users is really what PD is all about. What is new to PD is trying to make a framework or model to support discussions of whole development projects. Other, related approaches have done this. Many of the most recent project models for system development incorporate some notion of not only iterative design, but also prototyping. However, very few models make user involvement explicit.

In [9] a number of flow descriptions of different prototyping and evolutionary development models are proposed and discussed. The models, for instance, outline that a 'Prototype' is output from a 'Prototype Installation' activity and input to activities like 'Organizational Integration' and 'Evaluation', but the book gives very little account of how prototypes are related to the current and future work practices of users and on how users are involved in the

development process. Rather, the main focus is on classification of prototypes and technical considerations about how to produce the prototype software.

The STEPS [18] methodology proposes a project model based on iteration. This model explicitly outlines high-level activities carried out by users, by developers, and cooperatively. Thus the user involvement is explicit in the model, but the model only deals with artefacts like 'System specification' and 'System version' and does not give explicit room for exploratory prototyping. Similar to the approach described by [9], there is only little focus on the actual participatory activities and the relation to the users' practice.

Finally, the above mentioned Spiral Model by Boehm [8] combines risk analysis, prototyping and specification in an iterative model. This model is explicit about using prototypes to reduce the overall uncertainty in the project, but the role of users and their practice in relation to the development activities is hardly mentioned in the paper.

Before we introduce the project examples and begin our discussion of the concerns of CESD, we briefly comment on a couple of aspects that are not dealt with explicitly in our model.

Planning activities

Planning of activities is not covered in detail by our model. The activities of a project are folded into the center of the top layer of Figure 1. Anyone who wants to plan a CESD project will need to unfold them to make activity plans, and this is no simple matter. As pointed out by Lucy Suchman [42] plans are necessary and useful resources or tools in development projects, but only if those using them have a good understanding of what projects are all about. It is our hypothesis that such understanding requires knowledge of concrete projects, including specific project activity plans, of techniques, and of methods on the level represented in Figure 1. In the last part of this chapter we use the model of Figure 1 to present the activities of a concrete project as they unfold over time and discuss how they relate to the concerns of the project.

Trade union influence

Trade union influence is an aspect of system development which has been covered in detail in our earlier work preceding CESD [2, 7]. But this aspect is not explicitly built into the model itself. The CESD approach presented in this chapter is based on work in a number of projects within the Scandinavian tradition of cooperative design, the Collective Resource Approach, CRA [14]. CESD shares with CRA the aim of supporting democratic influence on and in system development, but this concern is not in focus in this chapter. We focus on the system development project itself, not on the interface to worker/union activities—or to managerial activities for that matter.

Our aim is to contribute to an approach to system development that is open to democratic influence, supports participation of end-users on their own terms, and is open-ended in its creation of visions. This has led to our focus on cooperative experimental development, CESD. In our use of cooperative techniques we have a lot in common with Cooperative Design as presented in e.g. [19]. However, in our coverage also of the realisation concern in a system development project, CESD goes further than Cooperative Design, which often stops before any coding is done in the target environment.

Our conjecture is that CESD type projects are more amenable to worker/end-user influence than traditional projects—and are more likely to succeed in non-ideal political settings, with rather traditional demands on deliverables etc. than projects modelled more closely over the schema of the Utopia project [4]. However, worker/union participation in any project in a traditional company/managerial context should be based on supplementary, worker/union activities in parallel with the development work, activities that are based on the interests of the workers and controlled by them. Otherwise, such participation is unlikely to challenge the traditional managerial goal-setting and may not further democratic worker influence. In a

Scandinavian context, direct negotiations between unions and management about project goals and other major issues have been used [14].

THE CONCERNS OF CESD

In this section we discuss the concerns of CESD, one by one. The presentation is not intended as a project model, but to further a more conceptual understanding of the concerns. Readers unfamiliar with Scandinavian Cooperative Design might consult [2, 19, 22, 24], where more concrete case-based material is presented.

Project Management: establishing and sustaining cooperation

This concern is directed toward the project itself as a cooperative process. As any other project a CESD project needs to be established and sustained in order to function well. In CESD special attention must be paid to a number of aspects associated with cooperation.

People from all the different categories of (potential) end-users need to be involved in a CESD process

Most important, and often difficult, is to get the message through: that CESD implies cooperation between developers and (potential) end-users, not their managers or in-house IT staff. In most cases the first contact between user organization and developers is at the management level[2], and the expectation usually is that ideas and visions of management combined with the technical knowledge of IT personnel in the user organization are sufficient as contributions from the user organization. However, the different kinds of (potential) end-users need to be involved, need to be able to spend time on project activities. The reason for this is that concrete knowledge of and experience with work-processes are the basis for end-user contributions in CESD–an issue that has to be explained and illustrated throughout the first period of a project to be accepted and appropriated by the participants.

In order to take care of the day to day arrangements, specific contacts need to be set up between the user organization and the developers. Such contacts supplement the structures needed to evaluate project progress and make go/no go decisions. Usually we ask for one contact person from each group of end-users involved.

Project plans should accommodate the new role of the practitioners from the user organization including time off from ordinary work.

A project plan should be sketched for the entire project, with more detailed plans for the first period. Stipulations of resource spending, including involvement of end-users in cooperative activities, are important, since this is typically underestimated by the user organization. In order to get and sustain end-user commitment to project activities it is usually beneficial to reduce their normal work-load.

To facilitate a CESD project it is useful to make both project aims, plans and participants known in the user organization. Company-wide information meetings supported by small leaflets are useful and should be produced throughout the life of a project, since only end-users directly involved in project activities can be expected to know what is going on.

When we look specifically at custom development projects the most obvious distinguishing characteristic is that management of the user organization have decided that they want a project. The aim of the project will be that the organization benefits, and at least part of management will be committed to the project.

---

[2] Noticeable exceptions are the Scandinavian Trade Union projects such as Utopia, where the first contact was to the local unions within the user organizations.

However, top level commitment does not guarantee end-user commitment. This is something that has to be established explicitly. Similarly, organizational benefit, as defined by management, does not necessarily imply end-user benefit.

When a development organization wants to involve one or more user organizations in packaged software development, it is usually a major effort to get the initial commitment. A user organization can expect to learn about constraints and possibilities in its use of technology of the same kind as the software package. However, when more substantial contributions are sought from the user organization this is usually not enough. Either the package must be so mature that it can be used for real in a short time frame, or some other form of compensation must be found, e.g. in the form of paying the users for their time. For example, GB was paid with project funds in order to get acceptance from the organization as a whole and in addition substantial resources were spent to ensure benefits for individual end-users. This kind of contract with user organizations is recommend for projects aiming at developing packaged software if the user organization may have to deliver resources to the project which do not pay off immediately to the organization.

Finally non-disclosure of sensitive information to third parties and other unauthorised use should be dealt with up front. An obvious issue concerns restrictions on disclosure to people outside the user and developer organizations respectively, but in many cases it is equally important to establish some kind of protection of end-users against disclosure of their opinions etc. to superiors.

Analysis

The overall objective of CESD is change. Therefore, analysis is directed towards investigating the given use-practice in relation to possible changes. Analysis is seen as facilitating taking action in order to bring about change. The concern, thus, is both to investigate current practice in the user organization as it is, and to investigate its inherent constraints and potentials for considered changes.

Analysis in CESD shares the concern to understand the complexity of current systems with, for example traditional descriptive approaches [10, 46], and may utilise descriptive techniques from these. It shares the concern of getting a detailed understanding of current work practice with, for example, ethnographical(-ly inspired) approaches [28, 29], and utilises observational techniques inspired by these approaches, particularly in the initial stages. Furthermore, the analysis concern in CESD emphasises the overall objective of change within practice with the people from practice. Therefore, analysis in CESD [36, 37] is often realised through cooperative, experimental, and intervening activities.

Analysis in CESD supports a cooperative learning process with common learning agendas for practitioners from the user organization as well as developers.

Most approaches to analysis implies that only the analysts learn (as the ones mentioned above). CRA focused on mutual learning in the sense that developers learn about use-practice and practitioners learn about technological possibilities [13]. The concern of analysis in CESD supplements these in focusing on learning processes with common learning agendas, i.e. ‘analysts’ as well as practitioners from the user organization investigate current practice. People are cooperating on the same issue—current practice (organizational as well as technical conditions). Both have to deepen their understanding of the given current practice. The system developers because they are outsiders, and the practitioners for two reasons:

- one’s own practice is to a large extent taken for granted: Entangled in everyday work supervisors at GB, for example, usually do not consider what competencies they use, what constitute patterns in non-conformances, when are proposed solutions sufficient, etc.

- the practitioners are in many respects also outsiders: The manager does not know, at least in detail, what the secretaries are doing, and the secretary does not know what the supervisor does, and the supervisor does not know….

On the other hand, the common subject matter—current practice—is approached with very diverse competencies, perspectives, and backgrounds, and different groups will acquire different outcomes.

Challenging the established with alternative possibilities is a primary means to investigate constraints and potentials for change within current use-practice.

When the overall objective of CESD is change, it is not enough to investigate how things are currently, it is also important to investigate the dynamics within the current practice of the users. For example, in a prototyping session when current practice was challenged by new possibilities, it triggered a prolonged discussion regarding "invisibility" of office work, management's demand for accountability, the overhead of registration work, and how they would and could be changed. For a detailed account of how the use of prototypes may trigger new discussions about current practice and disclose some of its inherent constraints, see [37].

In other words, analysis in CESD focuses both on investigating practice as it is and on the constraints and potentials for change within the practice.

Experimentation is a primary means in an analysis of actual practice, contrary to formal explanations or 'espoused theories' about it.

The aim of experimentation in analysis is twofold. One is to draw attention to the otherwise not articulated or 'invisible'. The other is to challenge the existing, the taken-as-given. From the fact that things are and have been accomplished in a certain way for years it does not necessarily follow that it should continue that way, nor that it will continue that way. As a supplement to traditional approaches such as interviewing, observing, making surveys, etc., CESD 'asks the practice'—it experiments with alternatives to see where clashes between current practice and alternatives occur.

In general, cooperative analysis suggests to analyse constraints and potentials for changes in current practice by experimenting with alternatives. Dilemma games is an example of such experiments [3, 36]. They are accomplished by the participants acting through scenarios that expose dilemmas. It is led by one or more 'provocateurs' who on the basis of flexible scripts introduce scenarios and urge people to take action. The scenarios develop according to the actions chosen by the participants—actions have consequences—and may thus reveal problematic aspects of "easy" solutions.

Another approach to explore relationships between current conditions and possible alternatives is to use prototypes. Prototypes are (early) embodiments of abstract visions. Usually, they have been used in order to get early feed back into design. However, they might also be used to further the analysis concern, in that their use may represent a concrete alternative illuminating current practice. This is one of the reasons why early realisation of envisioned possibilities is at the core of CESD.

At the same time the cooperative nature of experimentation makes it possible also to address a range of issues usually outside the scope of analysis by outsiders: by drawing on the knowledge of the participating users, the developers may investigate the reasons for specific actions, why one option is preferred over another etc.

As a complement to users' input, CESD design takes advantage of existing systems, research results, and standards.

PD and CD are sometimes criticised for reinventing the wheel by letting design decision rely only on users' ideas and current work practice. However, in most development projects current technology, standards, and user interface guidelines play a significant role in shaping the design

of new systems. CESD pays explicit attention to this aspect by studying existing and analogous systems. CESD developers use their computer oriented competencies in searching for relevant (alternative) technology bases, standards, user interface paradigms etc. that can be transformed into visions of technology in use. And they use their current understanding of the problem domain and their experience to select existing technological concepts and systems which can be brought in as thought-provoking artefacts in cooperative workshops extending the participants' understanding of alternatives as well as current practice, see e.g. [35, 36].

For instance, in the EuroCoOp/EuroCODE project past research in hypermedia was studied intensively, with the result that the Dexter Hypertext Reference Model [27] came to play a significant role in shaping the core functionality of the system, whereas the user workshops played a significant role in pushing the design towards an open architecture where hypermedia functionality could be brought to the users favourite applications [24] instead of living in a monolithic system as originally assumed by the reference model.

The starting point for analysis in custom development and product development is often quite different. Although there is not one kind of custom development nor one kind of product development, the two stereo-types below may indicate typical differences.

In the case of developing custom software, the user organization and its conceived problems are more or less given and the goal is to find and agree upon alternatives: an organizational change including new or changed computer systems.

The concern in this setting is to investigate the given practice and to explore its inherent constraints and potentials in relation to a number of proposed alternatives.

The situation in product development can be seen as the reverse situation to custom development: the solution, the package, is given, whereas the problem, the intended uses, is more unknown. The concern of analysis in this setting is to investigate the constraints and potentials in a number of practices for *this* possibility, the given package, to become a realistic and likely possibility within these practices.

Design

Design concerns visions of technology in use, i.e. focus is on creation of common understandings of realistic technological visions related to specific domains and use situations. According to many system development methods design is an activity that takes over a requirement specification and perhaps a domain model from analysis and then converts the overall requirements for functionality to a specification of the internal technical structure of the new system being designed. The outcome of design is typically a detailed specification that is assumed to be given to a group of programmers who in turn implements the specification.

In contrast, much of the CD literature [4, 12, 19] view design as the main concern in system development. In design users and designers meet in a series of creative and constructive workshops undertaking experiments with possible futures based on artefacts such as mock-ups and prototypes. In the PD and CD literature focus is on techniques to facilitate common user and designer creativity.

Experimentation with possible futures, based on hands-on experience with mock-ups and prototypes is a central feature of CESD design.

To design cooperatively, to develop visions of technology in use, it is important to give these visions a form that allows the users to apply their knowledge and experience as competent professionals in the process. In Scandinavian CD, and in CESD, this is accomplished by simulating future use of the emerging designs [6, 14]. To do this, alternative designs are embodied in mock-ups/prototypes, example material are produced and potential use scenarios are sketched as a basis for the simulations. Ideas and focus for this work emerges from analysis,

where problematic or otherwise interesting work situations are identified, and from explicit idea-generation activities such as the Vision phase of Future Workshops. It was, for example, the first prototype in EuroCoOp/EuroCODE that enabled people at GB to experience what hypermedia could be in practice, as opposed to our previous attempts to explain it.

Preparation of efficient transformation of design artefacts such as mock-ups and prototypes into documentation and application code in the target programming environment is crucial in design.

The CESD design concern is much in line with the CD point of view on design but it differs in its emphasis on capturing the design results. Design is both concerned with techniques to stimulate cooperative creativity and techniques to capture design results in a collection of artefacts that support an efficient implementation of a system meeting the needs of the users. In CESD, CD techniques are supplemented and supported with object oriented tools and techniques to enable a smooth transformation of design artefacts to application code.

It is important that experiments as mentioned earlier are carried out with flexible tools that allow the joint user and designer team to rapidly change the design artefacts according to new ideas evolving during workshops. Flexibility of tools in line with cardboard and paper or software tools such as HyperCard, Visual Basic, and various Lisp based environments is given high priority in CESD. The ideal situation with respect to continued use of prototypes for evolutionary development [17], however, is to obtain a development environment, that can be used both for flexible prototyping and as target programming environment. Currently object-oriented development environments offer the best opportunity for cooperation in system development, cf. also the next section on realisation. In an object-oriented environment prototypes can be unfolded to object oriented specifications such that major parts of prototype code can be reused as application code.

With respect to mock-ups, it is beneficial to go beyond pure interface mock-ups and include representations of that which is to be handled by the mock-up/emerging system. Such mock-ups can support the end-users understanding of the emerging system as well as give valuable input to an object oriented specification of the system. In several of our mock-ups dealing with materials such as technical drawings, pictures and PERT diagrams we mocked-up databases with these materials and used them to support work on multiple copies, exclusive write-access etc. And we illustrated conflicting changes, version control on drawings etc.—all of which would have been extremely difficult with pure interface mock-ups. At the same time this work clarified several of the issues involved in deciding upon a suitable object-structure for the subsequent database prototypes.-

Realisation

Realisation concerns technical implementation of design visions and organisational change. Traditionally realisation appears late in the life cycle of a development process and involves mainly programming of the system from design specifications. In CESD there may be realisation concerns quite early and throughout the process, i.e. programming of early prototypes and conversion of example material may appear in early stages, e.g. in the EuroCoOp/EuroCODE project this happened during the first exploratory prototyping. In CESD, the realisation concern also covers changes to work procedures and the user's practice related to the system being developed. Thus, we view realisation in the broad sense similar to [1], where realisation is seen as an abstract function with a close interrelationship with design and analysis also including organisational change, e.g. user education, change of work procedures, and data conversion.

Technical implementation

Whereas PD and CD literature rarely pays attention to cooperation under the realisation concern, CESD sees cooperative and experimental aspects closely related to realisation.

Technical implementation requires understanding of design visions in addition to programming competencies.

The primary competencies needed for the technical implementation are understanding of the design vision created so far and programming expertise. Thus actors concerned with technical implementation should be professional analysts, designers and programmers, i.e. activities contributing to CESD implementation are not undertaken only by programmers. Artefacts such as prototypes, mock-ups, scenario descriptions, and various paper based specifications are used as basis for final realisation. But an overlap between members of the analysis and design groups and the realisation group is crucial to the success of any development project, since it is impossible to convey all information needed for implementation through the artefacts and descriptions [11, 38]. In the EuroCoOp/EuroCODE project we identified a number of co-operation scenarios regarding GB materials in course of the first prototyping experiments. At that time we only had a single user hypermedia prototype, but the scenarios were used as important input to the technical design and implementation of a hypermedia database which could support the cooperation scenarios by means of advanced transaction and awareness notification mechanisms. In order to convey the idea to the group implementing the database cooperation features, we generalised the scenarios and rephrased them in terms that mapped directly to the object structures of the hypermedia system being designed. The scenarios then became a common point of reference between the analysis and design oriented actors and the technical implementation oriented actors.

To improve cost effectiveness of CESD, object oriented tools which reduces or eliminates the CASE-gap are recommended for implementation.

In many development projects CASE tools are applied to capture the technical aspects of design in terms of e.g. SA/SD diagrams [46]. This, however, implies both a conceptual and a tool related gap ("CASE-gap") between analysis, design and implementation, e.g. data flow diagrams are used in SA/SD and the C programming language and an object oriented user interface library based on X-windows are used for implementation. In such development projects it becomes almost impossible to keep specifications updated with implemented prototypes and system versions, both of which often change due to new/unrecognised user requirements.

The traditional CASE-gap can partly be eliminated through the use of object oriented techniques that can provide a uniform conceptual basis throughout the activities that contribute to analysis, design and realisation. Emerging OO-CASE tools [31] and interface builders [21] can handle object oriented user interfaces and code specifications in both diagrammatic and textual form using the same common representation, such that consistency can be maintained automatically.

Prototypes and mock-ups are important design artefacts in CESD, and it is crucial to make efficient transformation of these into documentation and application code in the target programming environment.

Both prototypes and mock-ups need to be analysed from a technical point of view to break them down into modules and program constructs which can be organized in the target programming environment. This is a kind of 'reverse' design or redesign activity where prototype code and mock-ups are turned into object oriented specifications which becomes the application code. This transformation is a process that requires competence from both those who constructed the prototypes and mock-ups and from programmers who can produce the target environment implementation. With most traditional prototyping tools this is a manual process. However, if the CESD prototypes have been constructed with tools as those mentioned above, the code from prototypes can be reused in a reverse design process where OO design diagrams are generated (automatically) from the code of the prototypes. Since these

design diagrams are in fact a different *view* on the underlying representations of the code the diagrams can be used to carefully reorganising and modifying this code of which major parts can be turned into efficient and maintainable code and reused in a targeted implementation.

In the EuroCoOp/EuroCODE project, the code from early prototypes of the hypermedia system we developed [22, 24, 25] was reorganised, optimised and reused using reverse design techniques with support from the Mjølner OO-CASE tool, cf. Chapter 24 in [31].

Transforming a mock-up into application code is always a manual effort. The concepts used in the mock-up may, however, quite efficiently be translated into an object oriented domain model using an OO-CASE-tool.

Use of object oriented techniques for CESD throughout analysis, design and realisation supports traceability of user oriented concepts across all design artefacts.

When applying object oriented techniques many phenomena and concepts related to the users' practice will be mapped to classes and objects in design diagrams which in turn appear in the user interface and in the application code. This enables the developers to discuss the design and the application code in terms of user oriented concepts, thus making it easier to trace which objects will be affected by a design change initiated by users in say a prototyping session. Object oriented CASE tools make it possible to maintain code, such that concepts related to the problem domain and use situations are easy to identify and filter from the application code [31]. Thus it becomes easier to extend and modify the application code when new requirements are uncovered and formulated by the users at late stages of development or in later use of the systems.

Organizational realisation

A CESD process always implies some degree of organisational change related to the use of the new computer system being developed [34]. Some of this change is unpredictable and happens as a consequence of system installation and some is planned as part of the CESD design activities and is implemented in interplay with other activities in the CESD process. Space only allow us to cover a fraction of this organisational realisation concern. However, organizational change in a CD context is discussed by several authors [12, 14, 16, 36, 39].

In projects with a priory identified user groups and organizations, the CESD user representatives can help disseminate the new system in parallel with development.

With respect to the organisational implementation of a system, the user representatives who have been directly involved in the CESD process have gained a lot of experiences with the new system through participation in workshops and work with prototypes of the system. The user representatives can—at least in in-house and contract development projects—give direct feedback to their organisation. They can use their experiences from working with prototypes and the like in teaching, informing and discussing with colleagues about the new system [24, 39, 43]. Such interaction between user representatives and colleagues may also result in new contributions to analysis and design.

CESD supports an incremental organisational implementation of new systems which can give feedback to analysis and design.

In CESD, it is important to also learn from real use, thus it is recommended in any large development effort to break down the development assignment into smaller sub-projects corresponding to sub-tasks performed by the user organization [43]. Then the sub-projects can be completed such that the organisational implementation of the larger system takes place in smaller steps. This enables the CESD process of later sub-projects to take advantage of real use experience from earlier sub-projects already resulting in an organisational implementation [3]. In the EuroCoOp/EuroCODE projects several incremental introduction strategies were

identified [33]. Examples are: introduction of a new system in a specific department, introduction for a specific function across departments, introduction for a specific project starting more or less from scratch with new data material. The strategy to choose depend on the project assignment and other conditions for the project.

Computer supported work

When considering the entire life cycle of a computer system, the main concern is use. The system—if successful—is adapted into the work practices of its users. Use, however, often requires customisation/tailoring of computer systems, particularly when the actual users haven't been directly involved in the development. New needs for continued development are also quite frequently uncovered after a period of use. Such needs may be triggered by changes to work not caused by the system, or they may be due to circumstances not being uncovered during the main development cycle. To handle such new or uncovered needs, a CESD process should on the one hand be followed by a responsive support arrangement and on the other hand allow the users to tailor the delivered system to meet new needs.

The support staff needs to be familiar with the use context of the delivered system, and a short turn-around time for bugfixes and updates is crucial.

The support staff should be familiar both with the delivered system and important issues regarding the users' work. This may be achieved by involving support staff in parts of the CESD process or by letting support staff familiarize to typical use settings and tasks in the use domain after the system is delivered. Such familiarity enables the support staff to understand the users' problems. Also quick response from support is important, this makes it worthwhile for the users to report problems and inconveniences instead of just introducing less efficient "workarounds". Reports to support staff may contribute also to the analysis and design concerns.

In CESD needs for tailoring and continued development are seen as the rule rather than the exception, and depending on the development and use context 'use' is seen as an ongoing development process with a less frequent intervention by developers.

In CESD, cooperative analysis and design techniques may be applied during use to identify needs for continued development or tailoring.

For instance, Future Workshops can be undertaken to uncover problems and identify and develop users' visions related to a system in use. In the EuroCoOp/EuroCODE project, we conducted such a workshop as part of the evaluation of the third prototype which was evaluated by the users at GB in a period of three months, most of the time without developers being present. New features can be mocked up manually or prototyped using e.g. a scripting tool build into the installed system.

In the context of custom development or contract development, tailoring may be performed by the original developers using the original development environment. But it is more efficient to tailor a system in use, if it has been prepared for tailoring [45]. The packaged development context requires built-in tailoring mechanisms to allow people who were not on the development team and who don't have access to source code and environment to tailor the system.

To maintain evolving opportunities for use, it is important to pay explicit attention to creation of open points for tailoring, flexible system architectures and tools for tailoring, during analysis, design and implementation.

To deliver a tailorable system the architecture has to be prepared for tailoring. It is rarely possible to deliver a fully tailorable system with a user understandable development environment that allow users to modify any inconvenient aspect of the system. Instead, , the ambition in CESD is to deliver a system which is prepared for tailoring at some central places,

"open points", where variations in use are anticipated. The tailoring facility should allow the user to think in use oriented terminology when performing tailoring of a system. Unfolding selected objects and classes identified in the (object-oriented) analysis, will allow the user to recognize his familiar terms in the context of the computer system, and with an appropriate tailoring tool be able to modify the behaviour.

To deliver a tailorable system, tailorability must be paid attention to in all concerns: analysis, design and realisation [44]. An important source for designing open points for tailoring is to carefully consider places, where several alternative design solutions have been considered feasible. In such places it may be beneficial to generalise a solution which can be specialised to either of the alternatives by users. It is also important to consider the tools for tailoring during development. Many new operating system architectures provide a notion of scripting: AppleScript, DDE and Macros, TCL/TK which by means of light weight interpreters can open up applications to be tailored without having access to the original development environment [23]. In the EuroCoOp/EuroCODE project, we utilised such features of several standard packages such as Microsoft Word and Excel to integrate these applications with our hypermedia system.

ACTIVITIES AND THEIR RELATIONSHIPS TO CONCERNS AND CONTEXT

In the previous section we described and discussed CESD at the level of concerns without an explicit timeline. A development project, however, unfolds in a concrete process over some (limited) span of time, typically in a context containing several organizations. Such a process can be seen as a series of activities, e.g. meetings, workshops, design sessions, prototyping sessions, and program editing, which to varying degrees are dominated by the five concerns discussed in the previous section.

It is a main point in our CESD model that the 'concern' and the 'activities' levels, cf. Figure 1, should be seen as separate analytical levels, since a detailed analysis of activities in a system development project will show that most identifiable activities contribute to several concerns and vice versa—i.e. any one concern is realized through a number of activities. Ideally, an activity plan for a CESD project should thus not be formulated in concern terms like analysis, design, etc., but rather in terms like meetings and workshops corresponding to the different (CESD) techniques to be used. This allows us to view it as a normal situation that a prototyping session may contribute to all five concerns; and that a four hour programming activity may turn out to be a major contribution to the design concern and only a marginal contribution to the realisation concern.

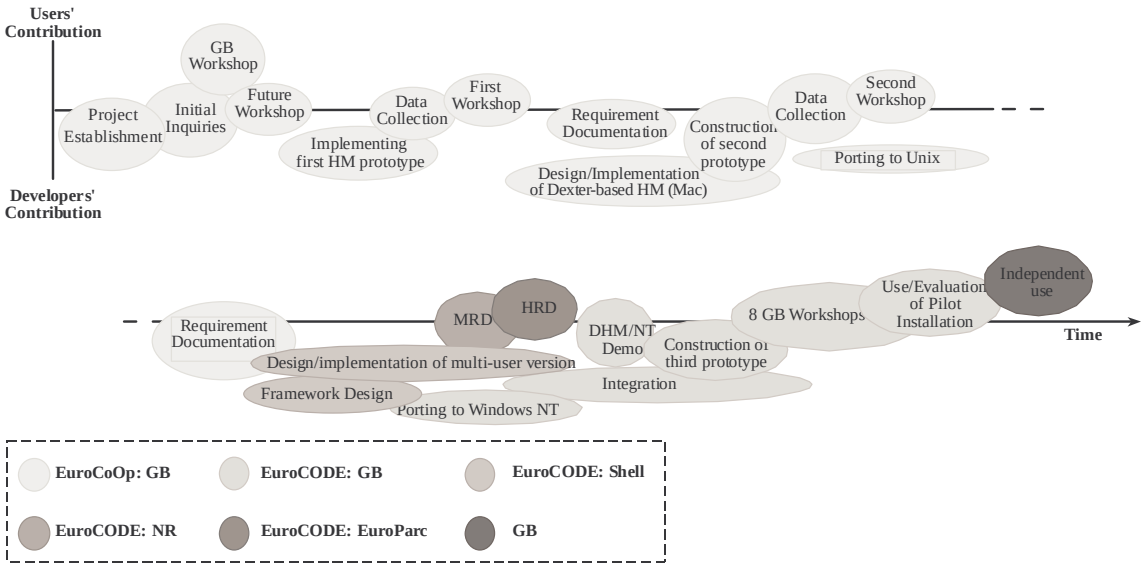

Figure 2: Actual flow of activities in the EuroCoOp/EuroCODE projects over a four year period[3].

In this section we illustrate how our main example, the EuroCoOp/EuroCODE project, was organized at the activity level and how the different activities focused on different domains and were dominated by different concerns. Figure 2 illustrates the sequence of activities at a level of detail corresponding to the high-level plan for the project. The next seven subsections discuss the activities with a little less detailed distinction of sub-activities. We do *not* present the many techniques developed over the years which can support CESD. Interested readers can find elaborate discussions of a number of techniques in [2, 3, 19, 20, 36]. These techniques (e.g. future workshops, mock-up design, cooperative prototyping, dilemma games, and organizational games) could in turn also be unfolded to illustrate how they contribute to the different concerns. Here we only do so implicitly through discussions of activities where the techniques have been applied. The references in parenthesis, e.g. (1M) below, represent relationships between the activities and concerns and they refer directly to cells in Table 1. The digits, 1-7, refer to the 7 activities discussed below and the letters refer to the concerns (**M**anagement, **A**nalysis, **D**esign, **R**ealisation, and **C**omputer supported work).

Project establishment

A main focus of the project establishment activity was on planning of tasks, deliverables and a schedule (1M), in order to provide a means of project management that corresponded to the EU requirements. Analysis of the state of the art within CSCW research and of CSCW markets (1A) also played a role in establishing a project that would focus on relevant problems and developments. Finally, initial visions about the technological innovations (e.g. cooperative design tools and hypermedia) of the project (1D) were formulated. This activity mainly involved the developers' practice and the visions about the project and products.

Initial inquiries and workshops

A condition to get going with the specific cooperative analysis activities was the formation of a design group with users experienced in supervision of construction processes. Such a group was identified during the first meetings (2M).

[3] Explanation of legend: GB stands for Great Belt, NR stands for Norwegian Computing Center (in Norwegian). 'Shell' is the concept used to describe the entire CSCW development environment being developed in EuroCODE.

Then the objective of the initial inquiries was to get an overall picture of the GB organization, its objectives, practices, objects of work (bridge construction), etc. We conducted interviews, studies of materials, studies of work practice in the user organisation, and evaluations of current technology in use (2A). This involved a number of visits to the headquarters in Copenhagen, a site office, and a construction site. To a large degree, the focus in the initial inquiries was determined by GB — they told, showed, and demonstrated what they considered to be of relevance for us. Our understanding of the GB work practice and the overall project assignment led to more specific inquiries focusing on the issue of sharing materials. A part of the more specific inquiries was a modified *future workshop*. It became evident that organization of huge amounts of heterogeneous material (thousands of drawings, text documents, reports from various reporting systems, e-mail, pictures, videos, and many more) was a key issue. In course of the future workshop vision phase several ideas of improved work procedures and improved computer support were formulated (2D).

In this activity, the main concern was analysis, and the primary focus was on the domain of users'-practice. The initial inquiries and workshops improved our understanding of supervision practice and GB people reconceptualized part of their work. However, several times the inquiries made us focus on the domain of visions as well. According to the project assignment we should develop better support mainly for asynchronous collaborative *editing* of design diagrams and reports. However, the primary problems for GB supervision appeared to be handling of huge amounts of heterogeneous supervision materials. Hence, we turned the focus towards experimentation on the construction of hypermedia support for *handling* heterogeneous materials. As a result, the initial inquiries informed the design concern as well.

Experiments with HyperCard prototype

We planned prototyping sessions (3M) and a first experiment was conducted based on a prototype with basic hypertext features. This prototype was developed on top of HyperCard over a couple weeks, and documents from supervision were scanned (with OCR) and entered in the prototype (3R). Having prepared the prototype with example material, we conducted a series of sessions where 10-15 supervisors and secretaries from GB had the opportunity to experience hypermedia in relation to their work. Several aspects of the prototype were challenged. Both link structures and user interface were modified during the sessions.

The two major concerns in this period was analysis and design: to assess and concretise the envisioned future use of hypermedia. The general conclusion was that a company wide system with hypermedia linking capabilities would help overcome many serious bottlenecks in working with huge amounts of heterogeneous materials. However, for this possibility to become reality we had to deal with two issues regarding GB practice: a critical mass of supervisors, secretaries, and area managers should commit themselves to establish links when they saw relations between parts of materials; and we had to find out how to incorporate the existing material. During this activity it became more and more clear to us that we had to develop an open hypermedia service which could integrate the existing materials in their current form (3A). Furthermore, experiences from using the prototype in work like settings triggered a range of suggestions for redesigning the prototype including suggestions for span-to-span links (part of a document to part of a document–the first prototype only supported links from parts to the wholes), one-to-many and many-to-many relations (the prototype only supported one-to-one), different types of link markings indicating e.g. priority or type of linked document, and many more (3D). Finally, it became clear that in order to provide a prototype which could be integrated in the users work environment, we had to change our technological platform. Thus the experiments resulted in several adjustments of goals for the hypermedia development (3M). To sum up, the focus was on the domains of users' practice and visions, the primary concerns was analysis and design, and the activities brought to bear were coding and prototyping sessions.

Experiments with Dexter-based prototype on Macintosh

Shortly after, we began developing a generic hypermedia package (Devise Hypermedia, DHM) based on the Dexter Hypertext Reference Model [27], programmed in the Mjølner Beta System [31], on Macintosh (4R). The second prototype supported a range of different node types: Text, Draw, Movie and File. File nodes supported linking to arbitrary files, and following a link to a File node implied launching the proper application with the attached file, i.e. simple integration of third party applications. The prototype supported various composite node types and bi-directional links with multiple endpoints [25]. This development had its main focus in the domain visions and technology.

With the DHM prototype, we began preparations for a second round of workshops (4M). Through a series of meetings between people from GB and ourselves, we addressed issues of how to organize GB material into a hypermedia structure, who should establish links where and when (4A), and we discussed a set of use scenarios for DHM. Furthermore, we organised a body of GB material into a hypermedia structure (4D), thus focusing on change of the users' practice.

In the second round of workshops, first, about 20 people were introduced to the general idea of hypermedia, how it might support work tasks, and what it would require for it to do so. Secondly, in smaller groups, DHM was demonstrated by one of the supervisors and used in work-like settings by different potential user groups (4C). In effect, the prototype–which was developed primarily to support supervision–was now confronted with the work tasks of many other parts of GB. This led to new insight about technology and design visions as well as highlighted constraints and potentials regarding hypermedia-possibilities at GB. Ideas for improving the general design of DHM including features like awareness notifications (e.g. notifications to supervisors when parts of the procedure handbook were updated), user-defined link types, and providing links in read-only documents (4D). The workshops generated both experiences regarding current prototype and assessments regarding envisioned use of hypermedia at GB (4A). As a result we developed elaborate ideas on how GB material should be structured, the work with use scenarios indicated who would establish and use links where and when (4D). Finally we discussed a number of constraints and potentials for introducing hypermedia in organizations like GB.

Although, in general, DHM was seen as promising, there were two obstacles to DHM being introduced at GB: hardware platform (GB used mainly PCs), and lack of integration with the users favourite applications—the ability to make links, say, from paragraphs in their usual text processor documents to objects in their CAD drawings.

Development of Dexter-based framework and OODB

To overcome the limitations of the initial design and implementations a major effort on generalizing the first Dexter-based prototype into a generic, platform independent, multi-user hypermedia framework was planned (5M). Detailed design of an object-oriented Dexter-based framework for hypermedia and an object-oriented database (OODB) with cooperation support was conducted (5D). Implementation of the generic Dexter-based object-oriented hypermedia development framework (class libraries etc.) and OODB was undertaken (5R), and the process was monitored (5M) in order to ensure that the overall project deadlines were met. The main focus in this activity was on visions, technology, and developers' practice.

Experiments with open hypermedia on Windows/NT

The platform independent framework that was developed made it easy to port the DHM prototype to Windows/NT which was the user's hardware platform (6R). Subsequently, this version of DHM was improved with inter-application communication protocols (6D and 6R) to support integration with third-party applications on Windows. As a balance between the wishes

at GB, available resources, and technical possibilities we integrated with Microsoft Word, Microsoft Excel, and Bentley's Microstation. These systems were extended to support communication with DHM, facilitating creation, following, and inspecting links between, say, parts of a Word document to a range of cells in an Excel document and/or an object in a Microstation drawing.

The pilot test was planned (6M), and the third prototype was installed in a network consisting of five PCs supporting six users at GB (a secretary, the person responsible for quality information systems, three supervisors, and one manager) (6R). It was used for the handling of non-conformance and change request reports (NCRs and CRs) at GB in a three month period (6C). The use context was an elaborated and extended version of the use scenarios from the second prototype and was primarily focused on interlinking materials relevant for assessment of Non-Conformance Reports, NCRs, and Change Requests, CRs.

The two major concerns in this period were realisation and computer supported work: the assessment of the running prototype. The affected domains were users' practice, technology, and visions. This work provided substantial input. First of all, it has provided a new body of suggestions for improving DHM (6D). Secondly, it highlighted organizational issues regarding hypermedia introduction and use, for example the question of flexibility: one of the virtues of DHM as a general tool is its flexibility; however, in order to be used in GB work is needed on structuring and restricting the use in *this* practice (6A). Last, but not least, DHM has been confronted with several challenges sparking new visions and design ideas: how to support interlinking of hundred thousands of documents if used for maintenance, how to support a generic way to integrate third-party applications avoiding the need to tailor each new one, and finally the complex issue of handling integration with third-party applications in a multi user environment.

GB use of open hypermedia prototype in a new area

After the three months pilot use, GB wanted to continue using the DHM prototype for Windows/NT in a new area. First on hardware borrowed from the project, later they acquired new hardware themselves suitable for running Windows/NT, DHM, and all the third-party applications.

The primary use is the establishment of a considerable hypermedia structure supporting the organization of material relevant for maintenance of the bridge, which is expected to last for at least 100 years. Basically, the hypermedia has two objectives: 1) Enable supervisors to track down all information relevant to a particular problem, e.g. crumbling concrete on pylon 17, i.e. quality documents, non-conformance reports and letters regarding the agreed solution, pictures of the problem, etc. 2) To function as an inspection tool supporting planning of spot checks of the bridge. These are entered into the system with links to "historical" material, and the maintenance staff in turn update the structure with new information after performing the checks.

Clearly, the all important concern here is computer supported work (7C), the attempt to build up a use practice around DHM; and the domain involved is primarily users' practice (GB). The activities to accomplish this are almost entirely a cooperative endeavour for GB personnel. However, there is still a modest dialog with the Quality Manager about experiences (7A), and relevant (Re-)Design of enhancements based on feedback from user are considered for the general development of the DHM framework (7D). Finally, bugfixes and minor enhancements are implemented (7R).

Concern / Activity	**M**anagement	**A**nalysis	**D**esign	**R**ealisation	**C**omputer Supported Work
(1) Project establishment	●	•	●		
(2) Initial inquiries and workshops	●	●	•		
(3) Experiments with HyperCard prototype	•	●	●	•	•
(4) Experiments with Dexter-based prototype on Macintosh	•	●	●	●	●
(5) Development of Dexter-based framework and OODB	•		●	●	
(6) Experiments with open hypermedia on Windows/NT	●	●	•	●	●
(7) GB use of open hypermedia prototype in a new area		•	•	•	●

Table 1: Examples of relationships between activities and concerns

Table 1 summarizes the relationship between concerns and the activities in EuroCoOp/-EuroCODE project. It shows schematically how most of the activities listed contribute to most of the concerns but with various weight. At the same time it shows that there is a main progression of the project from dominance on the management concern over analysis, design and realisation to the computer supported work concern.

CONCLUSION

This chapter has taken the Scandinavian tradition in Cooperative Design as the starting point for discussing a more comprehensive development approach, Cooperative Experimental System Development (CESD). CESD features cooperative and experimental techniques throughout the entire life cycle of a computer system from initial ideas to tailoring of the system for specific use contexts. The *cooperative* aspect of CESD covers, e.g. workshop techniques similar to those of PD and CD which enable actors such as end-users, analysts, designers, and programmers with quite different competence to actively contribute to the development process. The *experimental* aspect of CESD covers the iterative approach where alternative futures are explored and compared through experiments with embodied design visions such as mock-ups and prototypes in work like situations. Compared to previous PD and CD approaches CESD differs in at least two respects: (1) it applies the cooperative and experimental techniques also in the parts of the project where the main concern is technical design and implementation, i.e. developers with CD competencies and technical skilled developers cooperate throughout a project; (2) it pays explicit attention to transformation of loosely specified design artefacts such as mock-ups and prototypes into properly engineered and documented computer systems.

In the presentation (and management) of a system development approach, in particular CESD, we have argued that it is necessary to analytically separate the abstract concerns, e.g. analysis, design, and realisation from concrete activities and techniques. A CESD model providing a framework for handling this separation and the rich variety of relationships among concrete

activities and the main concerns was introduced. Given this framework, an excerpt of the basic tenets for CESD has been presented and examples from our use of the approach in the Esprit projects EuroCoOp and EuroCODE were given. These projects represent research aiming at developing industrial prototypes of general tailorable computer systems. The CESD approach is, however, not limited to this development context but can be applied for contract development and in-house projects as well. We expect the CESD approach to evolve in the years to come, in particular through work at the DEVISE centre at University of Aarhus, where we develop both CESD techniques and supporting tools. An important source of experience to improve CESD is a continuous cooperation with European and in particular Danish companies engaged in software development.

Acknowledgements

We thank our colleagues in DEVISE, EuroCoOp and EuroCODE for cooperation in much of the work on which this chapter is based, we thank Angelika Paysen for helping in proof reading, and we acknowledge Esprit II, Esprit III and The Danish Natural Science Research Council for funding the work.

REFERENCES

1. Andersen, N.E., *et al.*, *Professional Systems Development: Experience, Ideas and Action*. Business Information Technology series. 1990, New York: Prentice Hall.
2. Bjerknes, G., P. Ehn, and M. Kyng, eds. *Computers and Democracy: A Scandinavian Challenge*. . 1987, Avebury: England.
3. Bødker, S., *et al.*, *The AT-Project: practical research in cooperative design*. 1993: Computer Science Dept, Aarhus University. Daimi PB-454.
4. Bødker, S., *et al.*, *A UTOPIAN Experience: On Design of Powerful Computer-Based Tools for Skilled Graphic Workers*, in *Computers and Democracy*, G. Bjerknes, P. Ehn, and M. Kyng, Editors. 1987, Avebury: Aldershot. p. 251-278.
5. Bødker, S. and K. Grønbæk, *Design in Action: From Prototyping by Demonstration to Cooperative Prototyping*, in *Design at Work: Cooperative Design of Computer Systems*, J. Greenbaum and M. Kyng, Editors. 1991, Lawrence Erlbaum Associates: Hillsdale, NJ. p. 197-218.
6. Bødker, S., K. Grønbæk, and M. Kyng, *Cooperative Design: Techniques and Experiences from the Scandinavian Scene*, in *Participatory Design: Principles and Practices*, D. Schuler and A. Namioka, Editors. 1993, Lawrence Erlbaum Associates: Hillsdale, New Jersey. p. 157-175.
7. Bødker, S., K. Grønbæk, and M. Kyng, *Cooperative Design: Techniques and Experiences from the Scandinavian Scene*, in *Readings in Human-Computer Interaction: Toward the Year 2000*, R.M. Baecker, J. Grudin, and W.A.S. Buxton, Editors. 1995, Morgan Kaufmann Publishers, Inc.: San Francisco. p. 215-224.
8. Boehm, B.W., *A Spiral Model of Software Development and Enhancement*. Computer, 1988. 21: p. 61-72.
9. Budde, R., *et al.*, *Prototyping - an Approach to Evolutionary System Development*. 1992, Berlin: Springer Verlag.
10. Coad, P. and E. Yourdon, *Object-Oriented Analysis*. 1991, Englewood Cliffs, New Jersey: Yourdon Press.
11. Dreyfus, H. and S. Dreyfus, *Mind over Machine: the power of human intuition and expertise in the era of the computer*. 1986, Oxford: Basil Blackwell Ltd.
12. Ehn, P., *Work--Oriented Design of Computer Artifacts*. 1988, Stockholm, Sweden: Arbetslivscentrum.

13. Ehn, P. and M. Kyng. *A Tool Perspective on Design of Interactive Computer Support for Skilled Workers*. in *Seventh Scandinavian Research Seminar on Systemeering*. 1984. Helsinki: Helsinki School of Economics.
14. Ehn, P. and M. Kyng, *The Collective Resource Approach to Systems Design*, in *Computers and Democracy*. 1987, Avebury: Aldershot. p. 17-57.
15. Ehn, P. and M. Kyng, *Cardboard computers: mocking-it-up or hands-on the future*, in *Design at Work - Cooperative Design of Computer Systems*, J. Greenbaum and M. Kyng, Editors. 1991, Lawrence Erlbaum: Hillsdale, New Jersey. p. 169-195.
16. Ehn, P., B. Mölleryd, and D. Sjögren, *Playing in Reality: A Paradigm Case.* Scandinavian Journal of Information Systems, 1990. 2: p. 101-120.
17. Floyd, C., *A Systematic Look at Prototyping*, in *Approaches to Prototyping*. 1984, Springer-Verlag: Berlin. p. 1-18.
18. Floyd, C., F.-M. Reisin, and G. Schmidt, *STEPS to Software Development with Users*, in *proceedings from ESEC '89. Lecture Notes in Computer Science 387*, C. Ghezzi and J.M. McDermid, Editors. 1989, Springer Verlag: Berlin. p. 48-64.
19. Greenbaum, J. and M. Kyng, *Design at Work: Cooperative Design of Computer Systems*. 1991, Hillsdale, NJ: Lawrence Erlbaum Associates.
20. Grønbæk, K., *Prototyping and Active User Involvement in System Development: Towards a Cooperative Prototyping Approach*, . 1991, Computer Science Dept., University of Aarhus.
21. Grønbæk, K., A. Hviid, and R.H. Trigg. *ApplBuilder – an Object-Oriented Application Generator Supporting Rapid Prototyping*. in *Fourth international conference on software engineering and its applications*. 1991. Toulouse, December 9-13.
22. Grønbæk, K., M. Kyng, and P. Mogensen, *CSCW Challenges: Cooperative Design in Engineering Projects.* Communications of the ACM, 1993. 36(6): p. 67-77.
23. Grønbæk, K. and J. Malhotra. *Building Tailorable Hypermedia Systems: the embedded-interpreter approach*. in *ACM conference on Object Oriented Programming Systems, Languages and Applications (OOPSLA '94)*. 1994. Portland, Oregon, US, 23-27 October, 1994: ACM.
24. Grønbæk, K. and P. Mogensen. *Specific Cooperative Analysis and Design in General Hypermedia Development*. in *Participatory Design Conference (PDC)*. 1994. Chapel Hill, North Carolina.
25. Grønbæk, K. and R.H. Trigg, *Design issues for a Dexter-based hypermedia system.* Communications of teh ACM, 1994. 37(2): p. 40-49.
26. Grudin, J., *Interactive Systems: Bridging the Gaps between Developers and Users.* IEEE Computer, 1991. April: p. 59-69.
27. Halasz, F. and M. Schwartz, *The Dexter Hypertext Reference Model.* Communications of the ACM, 1994. 37(2): p. 30-39.
28. Heath, C. and P. Luff, *Collaboration and Control. Crisis Management and Mutimedia Technology in London Underground Line Control Rooms.* Computer Supported Cooperative Work, 1992. 1(1): p. 69-94.
29. Hughes, J., D. Randall, and D. Shapiro, *From Ethnografic Record to System Design - Some experiencies from the field.* Computer Supported Cooperative Work (CSCW). An International Journal, 1993. 1(3): p. 123-141.
30. Kjær, A. and K.H. Madsen. *Participatory analysis of flexibility: Some experiences*. in *Participatory Design Conference*. 1994. Chapel Hill, North Carolina: Computer Professionals for Social Responsibility.
31. Knudsen, J.L., *et al.*, *Object-Oriented Software Development Environments - The Mjølner Approach*. 1993, Englewood Cliffs, NJ: Prentice Hall.

32. Kyng, M. *Scandinavian Design: Users in Product Development*. in *Human Factors in Computing Systems, CHI '94, Celebrating Interdependence*. 1994. Boston, MA: Association for Computing Machinery.
33. Kyng, M. and P. Mogensen, *Workpackage WP1 Task T1.3: Evaluation of EuroCoOp prototypes re work at A/S Storebæltsforbindelsen (Great Belt Link Ltd.)*, in *Report from ESPRIT Project 5303 EuroCoOp: IT Support for Distributed Cooperative Work*. 1992, Aarhus University.
34. Mathiassen, L., *Systemudvikling og Systemudviklingsmetode*, . 1984, Computer Science Dept., Aarhus University.
35. Mogensen, P., *Towards a Provotyping Approach in Systems Development.* Scandinavian Journal of Information Systems, 1992. 4: p. 31-53.
36. Mogensen, P., *Challenging Practice: an Approach to Cooperative Analysis*, . 1994, Ph.D thesis. Aarhus University, Daimi PB-465: Aarhus, Denmark.
37. Mogensen, P. and R. Trigg. *Artifacts as triggers for participatory analysis*. in *Participatory Design Conference (PDC)*. 1992. Boston, MA.
38. Naur, P., *Programming as theory building.* Microprocessing and Microprogramming, 1985. 15: p. 253-261.
39. Pape, T.C. and K. Thoresen, *Evolutionary prototyping in a change perspective: A tale of three municipalities.* Information Technology & People, 1990. 6(2-3): p. 145-170.
40. Royse, W.W. *Managing the Development of Large Software Systems: Concepts and Techniques*. in *Wescon*. 1970.
41. Simonsen, J. and F. Kensing. *Take users seriously, but take a deeper look: Organizational and technical effects from designing with an ethnographically inspired approach*. in *Participatory Design Conference*. 1994. Chapel Hill, North Carolina: Computer Professionals for Social Responsibility.
42. Suchman, L., *Plans and situated actions. The problem of human-machine communication*. 1987, Cambridge: Cambridge University Press.
43. Thomsen, K.S., *The Mentor Project Model: A model for experimental development of contract software.* Scandinavian Journal of Information Systems, 1993. 5(August): p. 113-131.
44. Trigg, R.H. *Participatory Design meets the MOP: Acountability in the Design of Tailobale Computer Systems*. in *15th IRIS*. 1992. Larkollen, Norway: Department of Informatics, University of Oslo.
45. Trigg, R.H., T.P. Moran, and F.G. Halasz, *Adaptability and Tailorability in Notecards*, in *Human Computer Interaction – INTERACT'87*. 1987, North-Holland: Amsterdam. p. 723-728.
46. Yourdon, E., *Modern Structured Analysis.* 1989, Englewood Cliffs, NJ: Prentice Hall.