Chapter 10: Design in Action: From Prototyping by Demonstration to Cooperative Prototyping

> "This also leads us to the recognition that the development of any computer-based system will have to proceed in a cycle from design to experience and back again. It is impossible to anticipate all of the relevant breakdowns and their domains. They emerge gradually in practice."
>
> (Winograd & Flores, 1986, p.171)

Some time ago, we worked with a group of dental assistants to design a prototype case record system to explore issues of using com-

puter support in public dental clinics. The application was not primarily intended to be used directly in the treatment sessions, but to help administer the patients' visits. The dental assistants would be the primary users of such an application, and they took part in a series of prototyping sessions to formulate requirements to how they would like to use computers in their work. They knew that some kind of computer application would soon be introduced at their workplace, and it was important for them to be able to influence the choice of system for their work. The following scene is taken from one of the prototyping sessions:

At some point two dental assistants were sitting together in front of the screen and I, the designer, stood just behind. They switched between screens containing pictures representing patients' teeth . Suddenly one of them said: "*The lower jaw is turned upside down - that's quite confusing according to how we number the individual teeth !*" -- I had turned both of the upper and lower jaws the same way with the front teeth pointing upwards. The dental assistants did not like this. One of the dental assistants explained: "*I want to think of the pictures of the teeth as I'm looking into the mouth of the patient when I look at the screen!*" -- I got hold of the mouse and used a few menu-operations to flip the picture. I dragged the transparent buttons linking the teeth to their history card to the right positions on the master card and all new patients added to the register had the new and improved teeth representation. The whole modification activity lasted only a few minutes and I did not even touch the keyboard. The dental assistants were amazed by the ease of changing the prototype. They were satisfied with the change and one of them took over the mouse again and continued using the prototype.

System design and user involvement

We saw that by actually using the prototype, the dental assistants were able to make demands for changes that may otherwise not have surfaced until the users got the final system in their hands. The designer, of course thought that he had turned the picture right. He had not realized that there was a "mapping" between the drawing and the position of the jaws in the mouth.

In this and similar examples, we see heavy arguments for a more direct and active involvement of users in the design of computer systems: To find out how the computer application functions in the use situation the users must be able to somehow experience this. We call this *envisionment*. To experience is neither to read a description of the computer application or of its use, nor is it to watch a demonstration. We have found *prototyping* very useful.in uncovering unarticulated aspects of users' work and having them contribute to the design of improved tools for their work. In envisionment breakdowns may lead to a change of the prototype, and eventually to a change of the future computer application. What we find useful in prototyping, compared to the use of mock-ups as described in Chapter 9, is that a prototype better shows dynamic aspects of the future application than a mock-up.

In this chapter we will discuss how to get going with prototyping that involve users actively and creatively. We will give examples illustrating how to obtain a close coupling between design activities and experimental evaluation of prototypes in work-like situations. Before going into the examples we give a brief overview of current prototyping approaches and point out how the approaches we propose are slightly different.

Different approaches to prototyping

A rich variety of approaches to prototyping have been proposed in recent years. They all provide possibilities for users to gain hands-on experiences before the final application is built, and yet, the way they are used today, they are not often applied this way: In Grønbæk (1989b) a critique of three categories of prototyping approaches is given. They are called "*prototype becomes the system*" approaches, *executable specification* approaches, and *exploratory* approaches.

The "*prototype becomes the system*" approaches dominate current systems development practice and literature. They aim to supplement a traditional requirements specification with a prototype of the user interface aspects before the implementation begins, primarily for the users to adjust details of the system. From empirical studies (Grønbæk 1989a) we see that such prototypes are used primarily for demonstrations of features, and not to let users

try them out actively. The main purpose of the *executable specification* approaches is to obtain a full, formal specification of *what* (parts of) the future system should do. The specifications are made in a formal, executable specification language, meaning that a program, and thus a prototype, can be generated automatically from the specification. In theory, users and designers can evaluate the specification by evaluating this prototype, in practice this is hardly ever done. The specification languages that serve as basis for generating the prototypes are usually not suitable as a means for communication with users and empirically it has turned out that a full-scale formal specification of a system requires an effort similar to traditional programming of the system. Offsprings of these approaches are seen in the so-called CASE-tools. Code generating CASE-tools have, however, not been used enough yet to assess their ability to support prototyping. In *exploratory* approaches mock-ups, simulations, and "throw-away" prototypes are developed employing various tools. The aim of the approaches is to make quick-and-dirty sketches of the computer application in order to clarify requirements for a new computer system. A number of cases describing user involvement in evaluation of prototypes exists, but there are not many examples, where users have been actively involved in the design and modification of prototypes and thus creatively influenced the future system. However, the advantage is that the rapid development of simulated applications can facilitate early hands-on evaluation. The prototypes in these examples serve mainly as substitute specifications for the application and to propagate ideas to detailed design activities (see also Bødker (1987b)).

Cooperative prototyping to stimulate user participation and creativity

The "traditional" prototyping approaches mainly take the perspective of the designers and software engineers, and pay little attention to user involvement in the design process. We will now introduce a slightly different approach that we call *cooperative prototyping*. The approach has its roots in the exploratory approaches described above, but we will demonstrate that prototyping can be a cooperative activity between users and

designers rather than an activity of designers utilizing users' more or less articulated requirements.

The cooperative prototyping approach aims to establish a design process where both users and designers are participating *actively* and *creatively* with their different qualifications. To facilitate such a process, the designers must somehow let the users experience a fluent work-like situation with a future computer application, i.e. users current skills must be confronted with new technological possibilities. This can be done in a simulated future work situation or, even better, in a real use situation. When breakdowns occur in the (simulated) use situation users and designers can analyze the situation and discuss whether the breakdown occurred because of the need for training, a bad or incomplete design solution or for some other reason. Breakdowns caused by bad or incomplete design solutions should be rapidly turned into improved designs, in order to re-establish the fluent work-like evaluation of the prototype. To fully experience the prototype, the users need to be in control of its use for some period of time, and thus to try it out in a use like setting. The roles of the professional designers include anticipation of the use situation and building/cleaning up the prototypes in between the prototyping sessions. Ideally, cooperative prototyping should be performed by a small group of designers and users with access to flexible computer-based tools for rapid development and modification of prototypes providing some (simulated) functionality that makes it possible to envision future work tasks.

Examples of Cooperative Prototyping

In this chapter we will give two examples. One is of a cooperative prototyping process where users participate directly in making certain changes to the prototype, using direct manipulation design facilities. This process is primarily a laboratory experiment. The second example of a cooperative prototyping process takes place in a real organizational setting, but with a less close cooperation on making changes to prototypes. In the first example, HyperCard on a Macintosh is used, and the focus is on *exploring a new graphical user interface* for a dentist case record system providing direct representation of teeth on the screen. In the second example, the

4th Generation System, ORACLE, is used, and the focus is on *exploring organizational aspects of a new computer system* in an office that manages registration of incoming and outgoing mail for a large Trade School. The two examples illustrate quite different cooperative prototyping processes, one focussing on technical issues and one on organizational issues, both issues that need attention in an Information System development project.

Prototyping of graphical user interfaces

This example was mentioned in the introduction. It deals with the development of a prototype of a patient record system for municipal dentist clinics (for more details see Bødker & Grønbæk 1989). The end-users are mainly dental assistants working in clinics placed in public schools in Denmark. The purpose of these clinics is to provide regular dental check-ups and treatment for school children. These clinics as well as the work of the individual dental assistants vary a lot depending on the size of the school, and the organization of work at the clinic. All the dental assistants in these clinics, and thus in the prototyping sessions, were women, most of them with no previous experience using computers.

This prototyping process was carried out by a designer together with a number of dental assistants in an educational setting where the main purpose was to learn about computer technology, and about systems design in general and for dental clinics in particular. The topic was to explore problems and prospects of developing a decentralized patient record system, combining administrative information with treatment-oriented information. The application was not primarily intended to be used directly in the treatment session, but enough medical information is kept to fulfil the demands for statistical information from the Ministry of Health, and to help administer the patient's next visit.

The prototyping process

In order to prepare the prototyping sessions we designed an initial prototype with some reuse and modification of a prototype case record system for general practitioners (Bajlum & Nielsen 1988). Pictures of human teeth were integrated in the prototype, and a set of fictitious test data was entered into the prototype. Later on, we

augmented the prototype with report facilities provided by Reports[1]. A few illustrating example reports were prepared. We had some knowledge of the application domain in advance, and the preparation activities required only 2-3 days of work from us as designers in order to provide reasonably stable prototypes for the sessions[2].

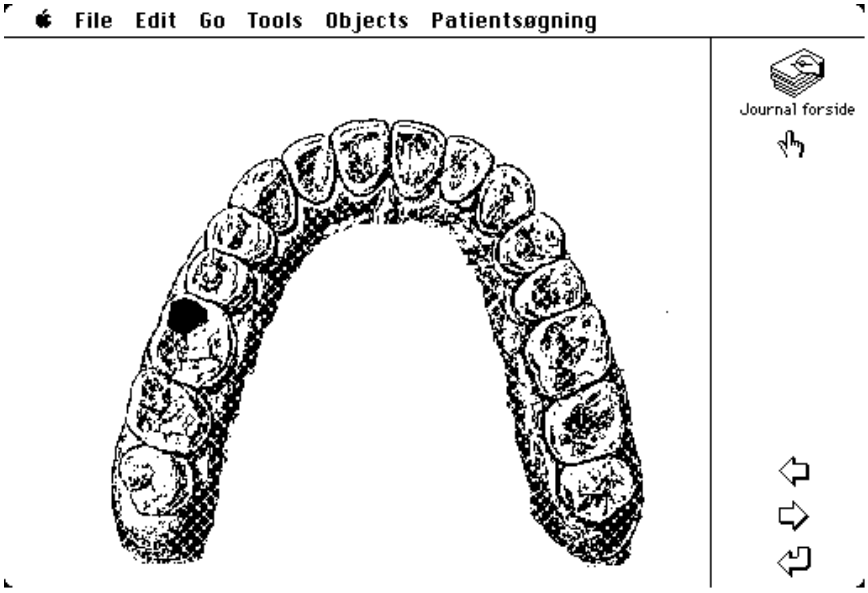

Figure 1: Direct representation of teeth in
a prototype dental patient case record(Lower mouth part)

The dental assistants were freed from their daily work to participate in the educational activities as described above. The ideas of the prototyping process were introduced and discussed with the whole group of dental assistants. The sessions also started with a short demonstration of the prototype to the whole group. Then the dental assistants worked with the prototype in groups of 2-4 to get a better understanding of the prototype. The designer, who was one

1 Reports is a program for retrieving information from HyperCard stacks and formatting it as reports for paper printout. Laying out reports is done by direct manipulation in an editor which is separate from HyperCard.

2 The main benefit we got from reusing the system for general practitioners was a piece of code that helped us generate a set of threaded cards to consist the case record for a single patient. The rest of the prototype was developed from scratch.

of us, had told the dental assistants that the prototype had been built in a flexible environment which allowed for changes to the design, thus the dental assistants were encouraged to come up with their suggestions for improvements. The designer tried to stay at a distance in order to let the dental assistants themselves explore the prototype and imagine that they were performing their daily tasks. There was no intervention by the designer in the process except in breakdown situations where the dental assistants had problems or suggestions for improvements. Some groups worked quite enthusiastically with the prototype and came up with constructive suggestions which we built into the prototype while they were in the midst of using the prototype, or after the sessions.

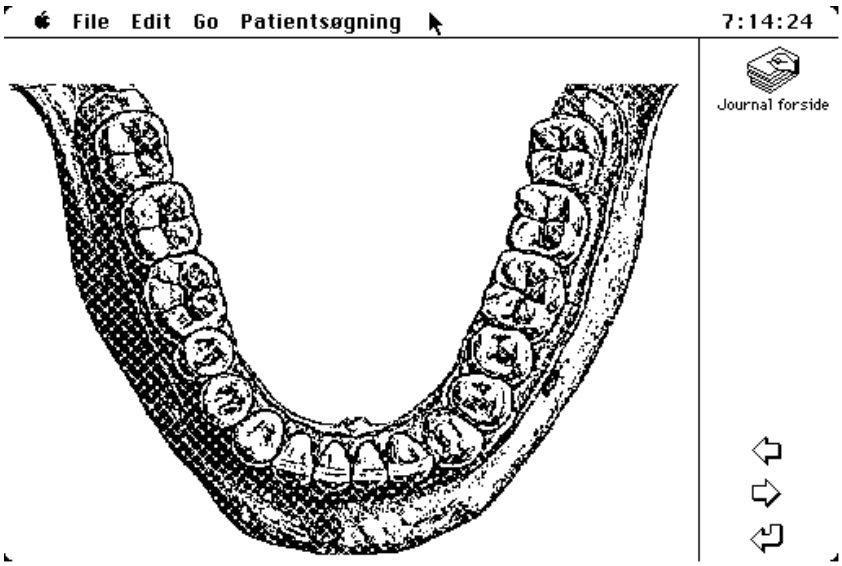

Figure 2: Direct representation of teeth in
a prototype dental patient case record (Lower mouth part)

Here are some examples of improvements to the prototype that were made in cooperation with the dental assistants in the prototyping session:

Change of teeth representation: The mouth pictures, which we used to indicate where treatment had been given, were initially turned upside-down, as we have described in the introduction, but

HyperCard provided point-and-select tools to turn the pictures around.

Exploring alternative representations of tooth treatment: The dental assistants asked for direct marking of fillings on the mouth pictures as an alternative to our solution which had each tooth as a button link to a separate card with a verbal treatment description. The visual aspects of this alternative representation could immediately be explored using the freehand painting tool of HyperCard, and thus be compared to the initial proposal.

Report lay-out design: The dental assistants participated in laying out simple reports from scratch and changing existing ones. This was possible because objects such as fields, text and graphics could be instantiated from available menus.

All of the above modifications were done through *direct manipulation* and we made many such changes. The general experience was that the dental assistants became quite enthusiastic and creative when they discovered the potentials of making such changes. We also made modifications that required modest programming:

Copying buttons and modifying scripts: The dental assistants asked for functionality that was quite similar to functionality already available in an existing button. A few times we copied such a button and made some minor modifications. When this kind of programming lasted longer than a couple of minutes, the dental assistants lost their patience, because they could not follow what went on.

"Simple" query formulation and modification: When the dental assistants were participating in the design of reports they also had to participate in the formulation of queries. This soon became a little too hard. Not surprisingly they had to rely totally on the designers suggestions regarding more complex queries constructed from ANDs and ORs.

A number of suggestions and ideas that came up during the sessions could *not* be integrated in the prototypes directly. The following are examples of this:

- The implementation of a menu with items to search for patients was made after the sessions, because it is rather time-consuming to program global menus. Moreover, the change could not be done via direct manipulation.

• In the sessions we only made reports with data from one type of cards. One group realized that they wanted to print out a survey of the treatment of all the teeth of a patient. We had to tell them that this ought to be done, but that it was quite a cumbersome process. Programming reports was possible, but again the dental assistants would loose their patience if we started doing that in the sessions. For this reason many good suggestions for reports were never realized in the prototype.

Lessons learned

Together with the dental assistants we came up with a number of conclusions about the prototype, e.g.:

• The idea with cards and direct representation of the teeth seemed quite promising.

• The idea of having each tooth on a separate card did not work for the dental assistants. They needed to be able to get a quick overview of all the treatment that had been given to a patient. This kind of overview could be provided with direct marking on the teeth combined with links to more detailed information. Similar combination of verbal and direct marking was needed to deal with treatments affecting several teeth. A major restructuring of the prototype seemed to be necessary to fulfil these demands.

Before the sessions we expected that the major challenge would be to keep the unreflected action of the users going in a prototyping process when, at the same time, we often had to stop and make changes to the artifact they were using. In our situation we could not go out into the real setting and have dental assistants use the prototypes there, and also the environment was not suitable for simulating a real dental clinic. It was the fact that the dental assistants were together that made it possible for them to 'be in the situation' by having to demonstrate to each other how they did things 'at home'. It was possible for the dental assistants to step into the illusion that they could perform realistic work tasks with the prototype. The real challenge was to interrupt the process and modify the prototype, when they were criticizing some aspect of it, and then restart the evaluation process again. Using a prototype providing familiar cards and pictures of teeth made it possible for

them to formulate quite specific needs and requirements to a future computer application by using their own language and pointing at the prototype.

We expected that using HyperCard and Reports would make it possible to design solutions to the problems with the prototype right away in a number of situations. As expected it caused some problems when major HyperTalk programming was needed. Moreover, with the present tools, it was clear that a certain clean-up of the prototype versions were needed. There is still quite a lot of work for the computer persons in between meetings.

We see *direct manipulation facilities* as provided with HyperCard/Reports as quite important for a cooperative prototyping process with in-session modifications. The direct representation of the data structure onto the screen (the cards, fields, etc.) was valuable in this case. We can easily see situations where the card structure would be a limitation, but in this case the direct mapping of 'good old-fashioned' cards on the screen made it relatively easy for the dental assistants to understand what the prototype could do for them and how the direct manipulation changes affected it. From our experiences we see mainly two sources triggering undesirable breakdowns in the design situation: reaching the limits of direct manipulation possibilities, and making changes that requires major restructuring of the prototype.

These experiences show that it is possible for a group of workers to come up with constructive and creative contributions to the design of their computer applications, when given the chance. The discussions quite easily get focussed on the current prototype and rather technical issues, though, as can actually be seen from the conclusions. But regarding work-oriented issues the users in this case found out by experiencing a prototype that treatment-oriented work could be supported much better than with current kinds of systems they could get in their workplace.

Furthermore, the study illustrates that it is not necessary to aim at a prototype which simulates full functionality. It is possible for users to abstract from a certain knowledge about these matters when they are provided with a prototype with some essential functionality. The initial prototype was stable enough to be evaluated in a work-like situation, and flexible enough to allow in-session modification. We conducted a cooperative prototyping

process where users used the initial prototype for work-like evaluation, and in cooperation with the designers the prototype were modified whenever breakdowns in the work-like situation occurred. This was a case where we made work-like evaluation and in-session changes of prototypes together with users in a laboratory setting. The following section describes a case where prototypes are tried out in the real organizational setting.

Prototyping in an organizational setting

In our next example, experiments were made with hands-on prototyping utilizing a 4th generation system, ORACLE. This example takes prototyping out of a laboratory setting and into the actual work setting. It also deals not only with a small group of volunteers, but with a large organization. In the example we worked with a large Danish Trade School administration, distributed over a number of different locations. This administration takes care of budgets and other financial issues, management of buildings and other facilities, including construction work, registration of students, salaries and other staff administration, supplies, secretarial work, etc. Many of these functions are partly located centrally and partly locally. In this administration a large project was carried out with one of us as a consultant. The purpose of the project was to create an integrated office automation system, which allowed for a more efficient administration of the school. The office administration system should be financed not by laying off employees, but by allowing for more efficient use of such resources as classrooms and heating.

The project was initiated by management of the school. According to the local technology agreement[3], it was managed by a technology committee with representation of management and employees. It was the general idea of the project that the employees should, in project groups, take part in designing the computer applications that they are to use themselves. The school hired a number of consultants to work together with the employees in the design work. The actual realization of the computer

3Technology agreements are agreements made between unions and employers in order to regulate the development and use of new technology within certain use domains. Refer to Mathiassen et al. (1983) for details on laws and technology agreements.

applications was to be carried out by a computer manufacturer on the basis of the specifications and prototypes created by the users and consultants in cooperation. The case described here deals with one of these project groups, working with information storage and retrieval. The purpose of the group was to find out how the files of the school could be reorganized to be more efficient, eventually by means of a computer application[4].

The file of all incoming and out-going documents represents the history or memory of the organization. The information retrieval was, in the then-existing structure rather cumbersome. The office, where the file is located served the case workers in the administration, who acquired documents on specific issues from the file. The project group consisted of the women working in the filing office, representatives of the case workers who were the users of the file, and consultants with competence concerning organizational issues as well as computers. There was a general agreement among the users that the way documents were filed and retrieved was not working well. Every group of workers had their problems with this, which caused different and sometimes conflicting ways of understanding the problems.

The prototyping process

The consultants prepared by interviewing the participants and observing the work in the filing office. Then the group gathered to start its part of the work. After initial discussions, three alternative ways of filing emerged. Scenarios were made of how the future would look with the three alternatives. Also different ways of organizing the file, and of doing the filing and retrieval were envisioned by the group. This was done through visits to trade shows and other workplaces where filing technology had been applied. Furthermore, the filing office borrowed a couple of filing systems from vendors, and had them in the office for a couple of weeks' time. This allowed the group to focus on what it wanted and didn't want from these different applications and the organization of work around them.

4 For a more detailed discussion of the project and the specific case see Kristensen et al. (1986).

After this, an initial paper based sequence of screen image simulations, was discussed in the group. This mock-up illustrated one of the three alternatives, a paper based file with lists of incoming and outgoing mail kept and distributed via the computer (see Chapter 9 for a discussion on Mock-ups).

On the basis of these discussions, the consultants built a first prototype by means of ORACLE. This was used in trying out, for instance: What does it mean to have the lists of incoming and outgoing mail computerized? Which information needs to be entered by whom? Which lists should be available to which case workers? How should the lists be structured? Who 'owns' the lists? At this stage there was a strong emphasis on the cooperation between the filing office and the case workers, on the different case workers different needs for information, and on the qualifications of the filing office workers.

As the versions of the prototype got more stable they were also used in the real use setting: the women in the filing office used the prototype to create mail lists and make these available to the case workers[5].

Programming the prototype required much programming experience because much of what the group wanted, concerning the cursor movements and the like, could not be programmed directly in the available version of ORACLE, but had to be programmed in Pascal. The prototype, as it stood, illustrated only a limited part of a future new application, a part which was running rather well. In the next step, where we wanted to expand this prototype, it turned out that the database had to be completely restructured, and the previous prototype, including test data thrown away. In this case we found that in particular management of the office was reluctant to take this step. Although the prototype was unable to handle large data sets needed for daily work, management experienced the prototype as something running perfectly well in the setting. Thus they tried to use it for work purposes as it stood, which of course failed. This issue is discussed in more detail later in a subsection on unrealistic expectations.

---

5 The workers denote various collections of task as "cases" and they label the person who take the main responsibility to treat the case as case-workers or case- handlers. Thus we selected the term case-workers for talking about these employees in general.

ORACLE allowed the group to run the prototype on the computer that was also used for other purposes in the office. For some weeks, the prototype was used daily, under supervision of some of the consultants. They helped each worker getting started, they were there when things broke down, and they made observations of the use processes. The aspects which were illustrated had to do with how mail lists worked. This prototype allowed for experiments with how a rather limited, but essential part of a future filing system would work in the real setting; what could be experimented with was the communication between the case workers and the filing office, which was something new as compared to the traditional way of making mail lists. Furthermore there came to be a strong focus on the qualifications of the filing office in relation to the work of the case workers (how much do the filing office workers have to know about the work that the case workers are doing in order to fill in the proper information in the mail lists?).

Lessons learned

We learned from this process that using prototypes as well as existing systems as alternative suggestions for the future, allows the participating users to formulate their suggestions better. It was not necessary to get to a consensus about the understanding of the problem as long as some solutions could be found that made everybody comfortable with the future use. By integrating the prototypes in the organizational setting, it became, as opposed to the dental clinics example, possible to focus not only on individual use, but also on cooperation among people. And still, to avoid a too technical focus, it was necessary to shift between techniques, and also to bring in other kinds of "prototypes", e.g. filing systems from other domains. The actual use of prototypes required a robust prototype, running on equipment which could be made available in the use setting. Because the prototype ran on the same computers as the various other programs that the workers used, it was easier for them to get started, and also easier to integrate it into the daily work tasks, than what it would have been with a prototype running on a separate computer. In this case the price paid was that it was difficult to involve the users actively in the actual construction of the prototype, because the programming effort was too big for

them to spend time on. The 4th generation system used was too limited in the facilities provided, the concepts used were too hard for the users to understand, and a structural change of the prototype was very hard. Furthermore, in this case, a major restructuring of the prototype was prevented by the inflexibility of the tool.

It was important for all the participants to keep in mind the status of a prototype: what purpose it is intended to serve, and which aspects of the future application it is illustrating, as to avoid problems of unrealistic expectations like those of management in this case.

We have seen a case where cooperation issues were important and where prototypes running in the organization, as well as borrowed computer applications, became important means of a cooperative design process.

How to get going with Cooperative Prototyping

We have now given some examples of how we did cooperative prototyping using existing tools. To get a cooperative prototyping process going it is crucial for the working group to establish a common understanding of the aims of the process, the status of the intermediate products developed in the process, as well as of the role of prototyping in the overall design process. Moreover, some organizational problems must necessarily be handled to establish a basis for performing cooperative prototyping in a specific project. These problems cannot necessarily be handled within the project, some of them needs to be handled before the project is established, or parallel to its work. We find it important, though, that the systems designers realize how they want the questions treated, in order to create better products. In this section we summarize some of our experiences as to suggest possible steps for systems designers to take in order to get going with cooperative prototyping. We present these suggestions by pointing at a number of issues or rather tensions between issues that we find it relevant to consider.

Establishing project groups: Making a workable group vs. involving a large user group

There are a lot of issues involved in selecting a competent group of users to participate in design/prototyping activities. Many of these are not within the control of the designers, but are determined by power relations, technology agreements and the like. The literature proposes a number of criteria for selection of participants:

- middle management can be chosen, because they are supposed to have an overview of the task domain
- participants can be selected to constitute statistical samples
- users can elect their own representatives
- employees with experience in using computers can be chosen
- the most skilled workers among the future users can be chosen
- the most enthusiastic among the future users can be selected
- the design can starts as a pilot project in one department of the organization and later be spread further.

We are not able to point to a single of these criteria as the most important, nor do we believe in setting up a single criteria. The appropriate choice in the actual project should be discussed carefully when establishing the group. However, the first criteria mentioned we consider particularly dangerous in relation to the view of design we present here. Middle management, who are only on an abstract level involved in the tasks to be considered, cannot be expected to make relevant contributions through hands-on evaluation of prototypes as they do not have the necessary familiarity with the daily work processes. This we say, even though we have in a previous project encountered the problem that workers at the shop floor have very little understanding of problems of planning and coordination of their work processes. In relation to establishing a working group of a reasonable size the second criteria is problematic too, because many computer systems have user groups which are far too big and diverse. What we find most important, though is to establish a working group together with *competent user representatives*.

It is also important that the group of designers and users get to know each other quite well, because cooperative prototyping is based on the assumption that contributions from all participants are important. Cooperation is crucial to maintain the *on-going mutual*

learning process between users and system developers. Steps to establish a project working group is discussed elsewhere in the book and in Andersen et al. (1990).

The product of the prototyping process is not only a computer prototype, prototyping is a learning process, and much of the new understanding must be spread to workers and managers who are not participating directly in the prototyping process. One way (Bisgaard et al., 1989) is to use the different prototypes in a process, where all involved personnel is guided through a compacted version of the prototyping process. In general the prototypes are valuable means in the education of future users, because education can start while the final computer application is being implemented. Similarly, the participants from a prototyping process can most likely act as teachers.

Depending on the size of the organization, a process such as the one described by (Pape & Thoresen, 1986) may be appropriate. They describe a prototyping process where first one intermediate prototype is built in one part of the organizational setting. The designers move on to a new part of the organization, bringing this prototype to be used as a starting point here. A new process with a new group of people is conducted, and the designers move on to yet another part of the organization, where, in this case, the final prototype is developed. As with all prototyping processes this requires a mutual learning process in each of the settings.

Setting up prototyping sessions: Designers as conductors vs. users being in charge

To users, designing a new computer application is secondary whereas for designers it is their primary work. This means that the designers are the ones who know how to set up the process, and who need to make sure that the group gets something out of their meetings. This involves considerations of a number of questions such as: What is the purpose of the session? How stable should the prototype be in advance? To what extent should in-session modifications be done? What setting should be chosen? How should the outcome be documented/evaluated? These issues are discussed in more detail elsewhere (Grønbæk, 1989a). Here, a remark on the question of stability and allowing in-session modifications is relevant. As we saw in the Trade School case, cooperative prototyping

does not require that prototypes are modified in the sessions, but in our experience, using direct manipulation for modifications is a good way to obtain engagement from the users. However, only a few types of prototype modifications can be done in-sessions, and it is necessary for designers to be aware of the constraints and to know when to postpone a modification until after a session with users. In particular much homework must be done by the designers to prepare prototyping sessions carefully. Thus the designers still hold the power when many design decisions are taken.

Moreover, designers often like to demonstrate all the wonderful features to the users in prototyping sessions. Our claim, however, is that demonstrations do not necessarily tell the users anything about how the prototype nor the final application fulfils their needs. To fully experience the prototype, the users need to be in control of its use for some period of time, to try it out in a work-like setting. If the prototypes are not sufficiently stable to let the users work on their own with it, the designer should be prepared to give first aid for breakdowns caused by the prototype. At the same time as the designers are, initially, the ones who know how the process should be set-up, it is important that the process is adjusted to the needs and wishes of the users.

Providing prototypes: Showing fantasy vs. being limited by the tools

We have given examples of how we used two quite different tools for prototyping, neither of which was ideal. Getting going so that the users can experience the use of some prototype of the future system is important. We find the experiences of potential problems and possible solutions gained from the early prototyping experiments quite valuable and thus worth the whole process. The prototyping activities can always be supplemented by using more flexible mock-ups and even traditional description to cover aspects of a future application that cannot easily be covered by the prototypes.

If prototypes that are very unstable or poor are presented to the users there is of course a danger of missing the point. Still, if there are no realistic possibilities of making better prototypes, the users' hands-on experiences with some imagined parts of the future system are still valuable. In the Trade School case we saw that

even existing but different computer applications can serve as sources of comparison to developed prototypes. Thus we could say that a poor example is better than no example, if the status of the prototype is made clear. We recommend that shortcomings of a prototype is compensated for by a good explanation about what the problems are, also in relation to the problems of a more thorough implementation.

Today a number of tools exists that can be utilized for cooperative prototyping (see Grønbæk, 1989b). Such tools are often available on PC's or graphical workstations, and they are not necessarily expensive. Smalltalk and various LISP tools can also be bought for PC's at reasonable prices. Some tools are useful for prototyping in certain application domains, although they are not from the outset application specific. For example, in the domain of patient case records, HyperCard, providing a card metaphor user interface, supports prototyping of some aspects quite well, because case records in the real world do consist of cards in folders. To summarize we find that it is possible to get far by utilizing the potentials of tools already available.

Finally, alternatives are useful for the users to get their imagination going and thereby for the group to discuss different ways of organizing work. Exploring alternatives are, thus, not a waste of time, but a necessity to get fantasy into the development process and to improve work of the users through computer support. A way to stimulate user engagement in proposing alternatives is to illustrate the ease of modifying a prototype as we did in the dental assistant example. Building up a toolbox of different primitives to make exploration of alternatives easy would also seem like a good investment in most cases, e.g. general primitives to support different interaction styles and devices.

Maintaining communication: Describing requirements vs. experiencing work

Users are not there to annoy the designers or to spoil their wonderful design, but because they are the ones who know the relevant work tasks. Users are not necessarily good designers of computer systems, but their more awkward suggestions may be grounded in tacit knowledge related to aspects of the work that the designers do not understand fully. As discussed in Part I, designers will perhaps

have to study the work of the users more carefully and discuss this with them further before suggestions can be understood or turned down. Keep in mind that the users are the key to the design of a useful system and the designers are the key to propagate the user demands into the technical design of the system. The delicate balance may well be to design an application which is both useful to the users and quality design from a technical point of view.

The most fruitful communication regarding the design is obtained when problems can be explained within language games familiar to the users. Examples of this are seen in the case with the dental assistants. We used a direct representation, a picture of the teeth on the screen instead of trying to impose explanations of for instance a form or a record data structure on them. At the Trade School we used concepts from the existing file and mailing lists. Moreover, the principle used for the interaction with the system was similar to the idea of putting information on cards/sheets in a case record folder in the manual case records. In general we have found it useful to have users and designers experience the use of prototypes in work-like situations and to make use of language games which are familiar to the users.

Users perception of the process: Realistic prototypes vs. unrealistic expectations

To be able to obtain a work-like evaluation in prototyping sessions prototypes need to be realistic and stable enough to let the users be in charge of the evaluation. But it is also important from the beginning to create awareness of prototypes as rough drafts which should be thrown away if they are bad. Not every prototype the group comes up with will be a hit, and many times the group wants to choose a minimal solution when making modifications in the prototyping session. The group may know right away that this solution is temporary, but it may still be worth pursuing.

However, prototypes direct the expectations of both designers and users in a way that creates a blindness towards other and maybe better ways of dealing with the issues being considered. One problem of blindness relates to the designers expectation that the prototype can be included in the final product. On the other hand we have, in the Trade School example, seen an example of user blindness towards an early developed prototype: Users in the

organization find the prototype so stable that they think of it as the final product although it is not suitable for handling production data. And a manager demands production data to be entered into the database and treated by the prototype. These observations make us stress the need for setting the scene and creating awareness of the objectives of prototyping approaches.

Maintaining focus: A Technical vs. a Work-oriented focus

Traditionally technical skills have been considered the only important skills in the development of computer applications, thus often working groups have consisted of only technically skilled designers and a representative for the "customer". However, experiences from systems development shows that the main problems are not purely technical, but rather grounded in the coupling between technical solutions *and* work, the organization of work and political issues at the workplace. Thus, early prototyping experiments should act as catalysts for focussing on this coupling, as illustrated by the Trade School example where prototypes were evaluated in the use domain, and e.g. cooperation issues were in focus. To maintain this focus requires more than the pure technical skills needed to build prototypes. Domain knowledge is crucial as is a certain level of knowledge on organizational issues in general. The question is if we can realistically have it all? In our experience, the size of the working group, and the enthusiasm of being actively engaged in design may well be spoiled if the working group grows too big.

Prototyping easily gives quite a technical focus on the computer application as was seen in the case with the dental assistants. There is a danger that the designer gets more focused on making 'a hack' than on discussing problems related to the use situation. The kinds of things that are talked about in front of the prototype often has to do with the interaction with the computer more than with, how work is organized around the computer. These matters are important, (Bødker, 1987a) but they are not all. To avoid a too technical focus all the time it is important, as we saw in the Trade School case, to shift back and forth with other techniques which have different foci. And it is important to use experiments with the prototypes in work-like settings to get to an understanding of such

changes in work as qualifications and organization of work around the computer application. In our experience it is important to combine prototyping with other techniques such as the ones described in this volume, and not be afraid of stepping away from the prototype.

Getting resources: Adding more resources to early activities vs. lowering development costs

By pointing at the benefits of cooperative prototyping early in a development project we also point at the necessity for development projects to allocate more resources for the early activities. What might require most resources, compared to traditional projects, is the participation of more users. To ensure a real, active involvement from users they have to be freed from parts of their daily work such that they are not required to do double work in periods. Yet, it is also important that the users keep their relation to the everyday work tasks so that they do not become managers or professional user representatives.

A number of authors (e.g. Boehm, 1988) have pointed out that the most expensive mistakes or shortcomings in system development are those made in the "early phases" where the focus is on analysis and design. At the same time, it is pointed out that the amount of resources spent is traditionally smallest in these phases. We agree with these authors that there is an imbalance between importance and resources. We believe that cooperative prototyping can help anticipate some of the expensive "mistakes" and most likely reduce development costs similar to what is claimed by Lantz (1986).

Design as an on-going process

We have been arguing that cooperative prototyping approaches can attack a number of problems traditionally related to lack of user involvement in Information Systems development. Of course they cannot solve all problems related to user involvement, such as conflicts in organizations or making decisions on when prototyping has produced sufficient clarification to implement a system.

But cooperative prototyping can be applied to improve the quality of computer systems seen from the point of view of the

users, i.e. the users can get a system that is tailored to their needs and thus improve the quality of their work. However, organizations are not static, and technological innovations also increase the number of different interaction styles that can be provided for essentially the same functionality. Neither designers nor users can gain "full" knowledge of the possibilities for providing computer support for an application domain; design is an on-going process. Thus cooperative prototyping should not be viewed as an approach to produce the ultimate computer system for an application domain. It should rather be viewed as one of the initial steps in the on-going development process where a computer application is augmented and tailored in conjunction with changing needs. Actually the possibility for users to create individual interaction styles and augmenting existing applications based on their own needs may not be that far out in the future. Such aspects of computer systems are referred to as adaptability or tailorability and these concepts are discussed further in Chapter 11.

References

Andersen, N. E., Kensing, F., Lassen, M., Lundin, J., Mathiassen, L., Munk-Madsen, A., Sørgaard, P. (1990). *Professional system development - experience, ideas and action*. Englewood Cliffs, NJ: Prentice Hall.

Bajlum, T. & Nielsen, T. (1988). *Edb-støtte til lægepraksis - et eksperiment med udvikling af en brugsmodel og en prototype* [Computer support for medical practice - an experiment with the development of a use model and a prototype]. Unpublished master's thesis, Århus: Aarhus University.

Bisgaard, O., Mogensen, P., Nørby, M., & Thomsen, M. (1989). *Systemudvikling som lærevirksomhed, konflikter som basis for organisationel udvikling* [Systems development as a learning process, conflicts as the origin of organizational development] (DAIMI IR-88). Århus: Aarhus University.

Boehm, B. W. (1988). A spiral model of software development and enhancement. *Computer, 21*(5), pp. 61-72, May 1988.

Bødker, S. (1987a). *Through the Interface – a Human Activity Approach to User Interface Design* (DAIMI PB-224) Aarhus: University of Aarhus (To be published in book form by Lawrence Erlbaum Associates).

Bødker, S. (1987b). Prototyping revisited - design with users in a cooperative setting. In P. Järvinen (Ed.), *Report of the 10'th IRIS Seminar, Vaskievesi, Finland*, (pp. 71-92), Tampere: University of Tampere.

Bødker, S. & Grønbæk, K. (1989). Cooperative prototyping experiments- users and designers envision a dentist case record system. *Proceedings of the First European Conference on Computer-Supported Cooperative Work, EC-CSCW,* (pp. 343-357).

Grønbæk, K. (1989a). Rapid Prototyping with Fourth Generation Systems - an Empirical Study. *OFFICE:Technology and People, 5*(2), pp. 105-125, September.

Grønbæk, K. (1989b). Extending the Boundaries of Prototyping - Towards Cooperative Prototyping. In Bødker (editor) *Proceedings of the 12th IRIS conference*, October, (pp. 219-239) Århus: Aarhus University.

Kristensen, B.H., Bollesen, N., & Sørensen, O.L. (1986). *Retningslinier for valg af faglige strategier på kontorområdet - et case studie over Århus tekniske Skoles kontorautomatiseringsprojekt* [Guidelines for trade union strategies in the office area - a case study of the office automation project of the Aarhus School of Polytechnics]. Unpublished master's thesis, Department of Computer Science, Aarhus University.

Lantz, K.E. (1986). *The Prototyping Methodology*, Englewood Cliffs, NJ: Prentice Hall.

Mathiassen, L., Rolskov, B., & Vedel, E. (1983). Regulating the Use of Edp by Law and Agreements, In U. Briefs, C. Ciborra & L. Schneider (Eds.) *Systems Design For, With and By Users*, (pp. 251-264), North Holland.

Pape, T., & Thoresen, K.(1987). Development of common systems by prototyping. In G. Bjerknes, P. Ehn, & M. Kyng (Eds.),

Computers and democracy – a Scandinavian challenge, (pp. 297-311). Aldershot, UK: Avebury.

Winograd, T., & Flores C. F. (1986). *Understanding computers and cognition: A new foundation for design*. Norwood, NJ: Ablex.