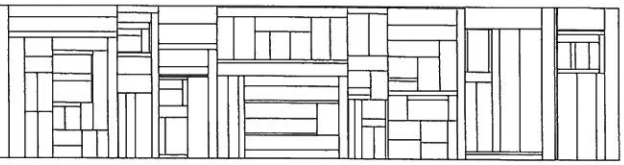
ISSN 0105-8517

PROTOTYPING REVISITED
– design with users in a cooperative setting

Susanne Bødker

DAIMI PB – 233
September 1987

AARHUS UNIVERSITY
COMPUTER SCIENCE DEPARTMENT
Ny Munkegade 116 – DK 8000 Aarhus C – DENMARK
Telephone: + 45 6 12 83 55 *Telex: 64767 aausci dk*



ABSTRACT:

In this paper, prototyping will be discussed from the point-of-view of user/designer cooperation in design. The starting point is that active user participation in systems design is a way of improving the quality of the design process as well as the product – a computer application in use. But, to participate actively in design, users must be allowed to experience the future use situation in the design process – to gain hands-on experience. The ideas of various prototyping methods seem to offer valuable help in this process.

Prototyping is, however, not one thing: we can not from an epistemological point of view claim that all prototyping approaches have the same aims. This advocates the necessity of distinguishing between different types of approaches. Cooperation between users and designers, the mutual learning process, and hands-on experiences to reveal the triggering of proper operations is only one side of the epistemological interests behind prototyping.

At the same time, the practical solutions of different approaches are to some extent applicable: we can learn from e.g. 4th generation tools, etc., but it is perhaps a good idea to reconsider the concept from the perspective of cooperation between users and professional designers.

CONTENTS:

1. INTRODUCTION

Recent research has provided both empirical and theoretical evidence to two types of statements which will be my points of departure in this paper. The first statement is that active user participation in systems design is a way of improving the quality of the design process as well as the product – a computer application in use. The second statement is that to participate actively in design, users must be allowed to experience the future use situation in the design process – to gain hands-on experience. This means that the best way for users to get acquainted with e.g. a requirement specification is not through reading but through use and discussion of prototypes or test examples of the future, running computer application.

To establish a situation where this is possible, professional designers need means of design. And not only this; perhaps the designers' traditional way of thinking of the whole design process need to be changed: If design is to be based on a dialogue with the users in which prototypes are build and tried out, design should perhaps no longer be seen as a series of steps in which more and more refined descriptions of the computer application are produced. One tradition of systems development/software engineering has a different, and perhaps helpful view of the process, the prototyping tradition. In this paper I shall examine the possibilities of applying prototyping, the way this is thought of within the software engineering tradition, as part of the above types of design processes. We can say that the purpose of the paper is to discuss the following question: *has the software engineering tradition already got the answer, when we, from a human activity point-of-view, ask for new design means?* Are we philosophically or practically trying to answer questions which have already been answered by the prototyping tradition? I do not expect a yes or a no, but something to get wiser from, conceptually as well as practically.

First, I will discuss in more details the type of design processes that are of my concern and present my approach to design. Secondly, I will discuss the relation between my approach and prototyping. This is followed by a discussion of three different styles of prototyping, their applicability for my purpose, and some problems that we must face.

2. DESIGN PROCESSES OF MY CONCERN

This paper takes its starting point in an understanding of design as a special human activity that shares a number of properties with other

human activities. The approach can be characterized as a human activity approach (Bødker[1]) because it is based on the human activity theory of Leontjew and others (Leontjew[2], [3]). It is focusing on the use of computer based artifacts in human work activity. Specific for human life, the way it is viewed here, is that human beings, as opposed to animals or things, create artifacts to be applied in a *future* use activity. By this choice of approach I deal with computer use as going on in purposeful work, with a specific organization and division of work, and based on a specific practice of the users.

2.1. Human activity and computer applications

Individual human work activity can be seen as part of a collective work activity of one or more groups. In the collective activity, language is used to coordinate work. Each individual activity consists of communication with others human beings to organize, coordinate, and control the activity, and of actions directed towards things which serve as objects or artifacts in the material production. Use as well as design is determined in a societal process, characterized by power relations and groups of conflicting interests.

Actions are consciously conducted by human beings. They are carried out through a series of *operations*. Each operation is corresponding to the concrete physical or social conditions for conduction of the actions, and it is triggered by the meeting with the specific concrete material conditions. Operations are senso-motor units which a human being performs in a specific situation, without consciously thinking of it, to perform the action which she is consciously aware of.

The individual human being possesses a certain *repertoire* of operations. This repertoire is part of the conditions of a specific activity because they form the basis from which operations are triggered by the meeting with concrete material conditions. For each specific action, the human being is dependent on the triggering of a sequence of operations. If these do not exist she must conduct different actions consciously. Most likely, she must conduct more detailed actions due to the lack of operations. We can consider a simple example, writing a letter with a new word processor – we need not only be conscious about writing the letter, but also about turning on the computer, opening the editor, etc.

Actions can be turned into operations, i.e. they are no longer conducted as a conscious effort but as a result of the meeting with the physical or social conditions. This is a typical result of learning. Operations can be brought back to the human consciousness when we

articulate for our self what is otherwise self-evident. When we use a new word processor we know, or become conscious of, that we cannot just operate it the way we did with the old one, and we conduct what were formerly operations as actions. Although the operations, when applied in a specific action are not conscious to us they can be made conscious to us as the actions they once were, in unfamiliar situations; we can name a specific sequence of operations, and understand and explain reasons for their application at the level which was the level of the former actions. In specific situations, or after conduction of a specific activity, we can ask a person how and why she did what she did.

One important kind of unfamiliar situations is *breakdown situations*, situations in which some unarticulated conflict occurs between the assumed conditions for the operations on the one side, and the actual conditions on the other. Let's assume that we can use the new word processor like the old one, and we start to write a letter. We might succeed with this for a while, but sooner or later something will probably be different and we are forced to see letter writing as something which requires other operations, and thus, other actions. A similar effect as breakdowns can however be achieved in learning situations, through a deliberate pedagogical effort; and in teaching situations where one is often forced to exactly articulate for one self what was otherwise self-evident.

Human beings apply *artifacts* in the intercourse with the world. Artifacts, some of which are computer applications, are not usually applied by their user through conscious actions. The effect of this is that the artifact is transparent in regular use situations – it is between the user and e.g. her material, but she operates on the material. When I apply a text editor, or even a pen, to write a letter, I do not think of pressing the buttons on the keyboard or moving the pen on the paper, but on the letter. Only in breakdown situations where it prevents the execution of certain operations does the artifact become the object for conscious actions. When I use a text editor to write this paper, I think primarily about the content of the text that I am writing, and eventually on the format as well. I do not think about pressing the keys on the keyboard, neither on the commands I use or the menus I open to get the editing done that I need. If, however, such things happen as some keys on the keyboard not working, or I can't find the ' key; if I do not find the command that I was expecting in the menu; a breakdown occurs, and I start to be conscious about which keys I press or which menus I open, and how.

2.2. Artifacts and the challenge for design

For the purpose of design it is interesting to focus on the character of the operations and their material conditions: In design we are going to change operations and their conditions for a specific activity, and for that reason we will like to focus on both actual operations and conditions, and future changed ones.

We face the problem that we cannot ask the person to predict her future operations in a future action. She will not know these until they are done; they are triggered by the material conditions, by the meeting with the actual nature or culture, not by any quantifiable set of conditions. We say that the operations are usually *non-articulated*. The material conditions are often *non-articulable*, which gives an action a certain character of unpredictability. Even though it is possible to get to know something about which repertoire of operations is possessed by the human being for some purpose, neither the person herself nor any observer can predict which operations come into play in the specific activity of use.

A group of human beings who conduct a collective activity with a specific object or goal shares a *practice*. The practice of a group arises from, and is carried by some common goal or object, as well as by the conditions of the collective activity, e.g. materials and organizational surroundings, the artifacts, languages, and ways of organizing the collective and the individual work applied in the activity. Practice is reflected in the repertoire of operations of the individual member of the group, at the same time as the individual member is part of constituting and producing the practice of the group through her actions and operations.

Some aspects of practice can be made *explicit*. They can be formulated in guidelines and theories. Cookbook recipes, text books about food and nutrition, dictionaries for chefs, and books about organizing work in big or smaller kitchens present examples of the explicit practice of cooking. They represent the articulable aspects of practice.

We can, however, only through practical experience learn the difference between a hand mixed cake or a machine mixed, the difference of using four large eggs or four smaller ones, of an oven which is warmer or an oven which is colder. Likewise can we learn the exact result of asking the kitchen assistant for three ripe tomatos, or how to know when a certain steak cooked a certain way is 'medium' as

ordered by the customer. Only through practical experience do we turn actions into operations, so that later the "right" operations will be triggered by the right conditions: e.g. to choose a fork or a whisk to beat the eggs for a certain omelet. We call these aspects *personal* or *tacit.*

When a human being learns to take part in a collective activity - gets socialized into a practice - she learns to use the means of work of the group, and she takes over the techniques, language, and ways of organizing work of the group. She starts out from an unfamiliar situation where most of the activity must be carried out as conscious or purposeful actions. With a growing experience some of these actions become operations, which means that they are no longer conducted consciously, but only as part of other conscious actions. An important part of learning is to build up a repertoire of operations.

H. and S. Dreyfus[4] give an example of how this learning takes place. They deal with five stages of competence from novice, over advanced beginner, competent, proficient, to expert. The novice, they say, knows the explicit rules of the activity, but has not yet gained any experience with the actual physical or social conditions, and thus, she has no operations. The advanced beginner has formed some operations which are applied in the activity. For the competent, the repertoire of operations for conducting the activity is rather big, but there are still many physical and social conditions which have not been met, for which reason breakdowns occur frequently. In all of these stages and the proficient stage, it is mainly the conduction of the activity which is done as operations, whereas the activity is still consciously planned – the situation is assessed on beforehand and a series of steps to be taken are consciously planned whereas the conduction of these steps at the proficient stage is done as operations, except in breakdowns where the whole process will perhaps have to be re-assessed and planned. The expert conducts even the planning as operations, she is acting in the situation.

A fairly familiar example for most computer people is how novices get socialized into the language of the computer world: from the novice stage where all the words and phrases are unfamiliar, they pass a stage where they use computer words all the time, even when talking about totally different matters, and where they are constantly corrected by the experts (breakdowns). Only after this, they get to a stage where they are able to use that language as the rest of the people around them, without constantly thinking of how to say this or that.

Human activity can, for this reason be characterized by situated action, rather than by planning and reflection. Human beings do, of course, plan and reflect - however, this takes place as a secondary kind of activity in situations where we are inexperienced in different ways; in breakdown situations. In familiar activities where the human beings are experts, our activity is conducted unreflected as a result of the meeting with the physical world or other human beings.

When a new artifact is brought into a practice this practice will change. Even the most competent expert will probably have to change her repertoire of operations, and for a while she is returning to a lower level of competence.

2.3. An example

We can look at make-up of newspaper pages. As the basic component, page make-up is conducted by one or more make-up persons, each carrying out his or her own individual page make-up activity.

The individual page make-up activity, is part of the collective activity of various groups, e.g. the make-up persons at the specific newspaper, or the persons handling the front page of a specific newspaper. The individual make-up activity is directed towards the newspaper page.

To organize, coordinate and control the collective activity, communication plays a role. This means that the make-up persons direct some parts of their activity towards other human beings, when for instance the make-up person discusses with the editor how to have the articles fit into the page. Other parts are directed towards objects: the make-up person handles paper galleys, pictures, etc., to actually create the front page.

The artifacts used in the activity are knife or a pair of scissors to mediate his relation with the physical world, the forming of the newspaper page. Lay-out sketches, production plans, etc. to mediate the communication and coordination of the production.

The individual human activity is conducted through actions, which are consciously conducted in a unity of time and space. Making-up a specific newspaper page may consist of placing an ad in the rightmost bottom corner, fit in some text between the ad and a picture, etc. An action is conducted through one or more operations, which are bound to specific material conditions. To place an ad the make-up person picks up the photo typesetter paper, picks the knife, cuts off some of the white area around the ad if there is too much white, without being conscious about this. The right operation to be used in a specific

situation is triggered by the material conditions; it is not chosen consciously by the typographer. When the typographer places an ad, we can say that the actions are what he is consciously doing, e.g. placing an ad in the right most bottom corner, whereas the operations are e.g. hold knife, cut paper, try position on page ground.

What in some situations are actions can in other situations be conducted as operations as part of other actions: e.g. placing a picture can in some situations be something which has its own specific purpose, whereas in other situations it has not, it is conducted as part of placing an article on the page ground.

Through learning, in special learning activities or in daily work activities, the person obtains a repertoire of operations to be used in a specific activity. He gets to share the practice of typographers, at the same time as he is part of constituting this practice: by actually doing the make-up work the techniques, the use of tools, the language, and the organization of work is reproduced. At the same time, if a make-up person finds out about doing something differently, this can result in a change of practice.

The make-up person can reflect upon what were formerly operations, and try to perform former operations as actions, e.g. if the editor tells him that he is not pleased with the product of the work of the typographer. Changing the level of action means changing the object (or subject) of the actions – instead of working on the article the make-up person starts to think about headlines, pictures, and the like. Unforeseen changes in the material conditions in the specific page make-up activity may cause breakdowns.

Situations where the make-up person's knife causes a breakdown could be:

•while learning to use a new kind of knife,

•if the knife breaks,

•if the knife is badly suited for the kind of cutting its user wants to achieve, a switch of knife can be necessary, or

•a special handling of the knife to achieve the intended result.

In such a situation the knife is no longer something which is handled only through operations. Rather the knife becomes the object for the actions, removing the focus from the real object, the newspaper page.

2.4. Design

The challenge for design of the artifact is to build on existing use practice to avoid that all experienced users are turned into novices, both in their more general practice of the activity and in the specific use of the artifact. Design changes practice by introducing the new artifact, but practice can also be influenced through learning taking place in the design process: if expert users take part in design, their general practice changes as does their specific practice in relation to the future artifact. This is because in the process they encounter many breakdowns, and teaching situations in which they are forced to articulate for themselves their otherwise self-evident practice.

Design of artifacts is a process in which we determine and create the conditions which turn an object into an artifact of use. The future use situation is the origin for design, and we design with this in mind.

Use, as a process of learning, is a prerequisite to design. Through use, new needs arise, either as a result of changing conditions of work or as a recognition of problems with the present artifacts through recurring breakdowns. These breakdowns are situations where the artifact, time after time, proves to be insufficient in the activity; each time causing breakdowns. The power relations and the division of labor are important factors for what kind of needs eventually leads to design activities and implemented artifacts.

Design of computer based artifacts is a meeting place for many different practices, and sharing experiences is something which requires a deliberate effort. *Design is a process of learning*, both when viewed as a collective process and as an individual process for the participants. The different groups involved learn about practice of the other participating groups. For the computer experts this involves learning about the work and prerequisites of the application domain. For the users who participate, learning about computers is involved together with learning about design of computer applications. For all groups the confrontation with practices of other groups contributes to learning about their own practice. This brings to design an innovative character: the confrontation with different practices, and thus, with one's own, is opening possibilities for new ways of doing things, and transcending the traditional practice of the users. This is the reason why design should be done with the users, not for, nor by them. Design done by the users alone, or design by the computer experts alone may well be very conservative, because they lack the possibility of confronting their own practice with other ideas.

Design is based on, and may change, all aspects of the practices of the users. The process of bringing into our consciousness the nature of our practice takes place in different situations triggered by different means. In design, *we need means to trigger awareness of all aspects of practice,* the language, the use of artifacts, and the ways of organizing work. At the same time, some of the personal or tacit aspects of practice is dealt with better without explicitation, because a dealing only with what can be made explicit reduces the physical and social conditions to a set of quantifiable conditions. Such a set is, however, not sufficient for knowing if and how the triggering of the operations will occur.

In this process two potentially conflicting goals of design come into play: that the future users must be able to assess the artifact-to-be, and that the programmers need a formal and detailed basis for their programming. This potential conflict can be dealt with in two ways: either to let the users be as detailed and explicit about their requirements as possible, or to provide the programmers with the needed competence to take part in design and help interpret breakdowns into actual programs. The human activity framework tells us that the first way is hardly feasible alone. The second way emphasizes the need for a collective learning process among the groups involved in design. Furthermore, to be a good designer means to be able to facilitate the reflection and interpretation in breakdown situations. In this conflict the interest for prototyping methods can be found, because many prototyping methods, as I will get back to, seem to allow us to consider both of the above ways.

To design an artifact means not only to design the artifacts for a specific kind of activity. As the use of artifacts is part of social activity, we *design new conditions for collective activity*, e.g. new division of labor, and other new ways of coordination, control and communication.

2.5. Better design

Good design must therefore facilitate the communication between these different groups. This can not be done just by introducing a common formalism or language, because such pre-supposes that the necessary aspects of the use practice, the computer application and its future use, can be made explicit. What human beings can make explicit of an actual activity is, however, what can be explained through a post-factum reflection only. The actual triggering of certain operations, and the physical or social conditions by which they are triggered, belong to a different domain – acting in situations. Why and when a

certain operation is triggered and by which conditions can not be turned into an explicit set of conditions and rules.

Thus, good design means a mutual learning process in which also the non-explicit aspects of use practice can be dealt with. In particular it is important to set up situations which make it possible for the users to somehow experience the future use activity; to gain hands-on experience with the future computer application in use.

As compared to traditional design methods, this view on design suggests that design is an incremental process, which is characterized by the experts' acting in situations, rather than it is a planned process. Furthermore, it is not possible to make the traditional distinction between analysis/investigation on the one hand and design/construction on the other: it is primarily in the early parts of a design process that different ways of facilitating active user participation is needed. On the other hand, breakdowns are encountered all the time in the process, both in relation to the present use situation and practice, and in relation to the new one, especially the new computer application.

3. THEORETICAL AND EMPIRICAL GROUNDS

In the above I have presented theoretical reasons for my interest in design that allow for hands-on experience. These reasons are discussed in detail in Bødker[1], and are inspired by the work of Winograd and Flores [5], Suchman [6], H. and S. Dreyfus [4], Ehn [7], and others. All of this work has in turn been inspired by philosophical work in the phenomenological tradition (Heidegger), the ordinary language tradition (Wittgenstein), and the materialistic, cultural historical tradition (Leontjew).

Seeking these theoretical reasons is, however, a result of the empirical experience of the UTOPIA project, described in [8, 9, 10, 11] and elsewhere. Here it became obvious that many of the traditional description methods, known to us, failed to work in several respects. We did not, however, choose to apply any specific method or tradition. Instead our ad-hoc attempts lead to the different mock-ups and simulation facilities. It is in this that we can find the reason for the title of this paper: the idea is to go back and reexamine some of the design methods or traditions which were once candidates for use in the project; with the new understanding of design achieved practically as well as theoretically.

I see in the "Knowledge and Work" project ([12, 13, 14, 15]) and the Florence project ([16]) arguments and experiences which are also in

favour of a theoretical approach as the one presented here, and I see these projects as important supplements to our research.

A theoretical framework, as the above, is not trying to describe or explain how systems design in general takes place, and why. Instead it presents an understanding of which role computer applications play in human life. The understanding of this role makes us see how systems design ought to be conducted to more certainly create computer applications which fulfill this role. This in turn leads to an interest in better design methods or practices to set up and support such design situations. In short, I seek ideal cases and examples of good methods. I hope that these can influence practical systems design, although I am aware of the many problems involved in this: systems design is most often not a democratic process, but a process governed by power relations and differences in available resources among the involved groups of interests.

4. PROTOTYPING

With the present theoretical and empirical background, I find it important to go back to see what we can learn from more established traditions in the attempt to consolidate the practical sides of the approach. One school in particular comes into mind when looking for hands-on possibilities and evolutionary, learning approaches: *Prototyping*.

In her survey of prototyping, Floyd [17] tells us that "*Prototyping refers to a well-defined phase in the production process, where a model is produced in advance, exhibiting all the essential features of the final product, for use as test specimen and guide for further production*". She also says that "*a prototype should always be considered a learning vehicle*", and that "*the overriding concern in prototyping is a commitment to the quality of the desired product*". The latter two of the above statements show similar concerns as my own. The definition of prototyping, however, suggests that prototyping is perhaps a subclass of the type of methods that I am after.

Friis[18] suggests that a precise definition of the concept prototyping is perhaps not as important as the discussion of the possibilities of a new view of design, based on simple "throw-away" models of the future computer application. I agree with her in that, but I will for the purpose of the following discussion try to give definition of the concept. On the one hand, prototyping is a way of doing systems design which allows the users to experience their future work/use

situation by allowing them to act in this. On the other hand, the goal of the activity is different from that of the use activity: to gain experiences about the computer application for the purpose of further design, as opposed to the actual conduction of the work.

When I use the concept prototyping in the following it is to reveal differences between examples of methods which we tend to cover by the term. I propose a distinction between three different styles of prototypes and I illustrate the styles by examples. The examples as such serve the role of prototypes, archetypes, or paradigm cases of the different styles.

Floyd, as well as others, suggests taxonomies of prototyping techniques. As I am after the more fundamental perspectives behind the techniques too, I shall apply such taxonomies together with Bansler's taxonomy of research traditions [19] (here taken from Pedersen[20]), with the focus on differences.

5. DIFFERENT PROTOTYPING STYLES

Bansler categorizes traditions according to what types of problems they aim at solving (epistemological interests); what are the basic assumptions? which are the fundamental concepts? and how are practical problems solved? I will, in the following, interpret these dimensions in the context of prototyping. Furthermore, I discuss some typical characteristics of prototyping styles along these dimensions.

5.1. Epistemological interests

The epistemological interests resembles what Sol[21] calls "Why prototyping?" Fundamental differences can be seen between seeing prototyping as e.g. a vehicle of learning, for the designers or the users, or as a way to facilitate programming by non-professional programmers. We find these differences in the following examples.

Wasserman[22, 23, 24, 25] presents a method based on a combination of a formal specification method and prototypes based on a computer interpretation of these. In this presentation, he points at six reasons for his method [23]:

1. improved reliability;

2. verifiability, at least in an informal sense;

3. improved maintainability, including portability and adaptability;

4. system comprehensibility, as a result of improved structure;

5. more efficient management control of the development process;

6. higher user satisfaction.

In his newer paper [25] he suggest other points which are overlapping: the suggested notation has to serve as a formal definition of the interface, it has to be self-contained, comprehensible to designers and users, flexible with respect to interaction styles, and executable to support prototyping.

These points present to us three epistemological interests which are frequent in prototyping: the aim to improve technical reliability and verifiability of the product; the aim to improve efficiency of the design process, and systems design as a learning process for the users. Wasserman[23] says: " *..it lets the eventual user interact with 'the system' at a very early stage of the life cycle, and provides a 'feel' for the character of the eventual system.*"

To Wasserman the technical reliability of the product, and the efficiency of the design process are the reasons for his concern – also for the user acceptance: if the users don't accept the product, the design process has to go into a new cycle, the sooner the users get acquainted with the product, the less risk of extra cycles due to the requirements of the users. Prototyping is taken into use, because the users are not capable of reading the formal specifications without too much help. Prototyping can be seen as a short-cut to make users understand the specifications. It is not a matter of testing whether the user can apply the artifacts through operations in regular use, and how the user meets the physical and social conditions of the activity, but a matter of having the user reflect upon the future activity through concrete actions, and not through reading of abstract specifications We move in a domain of reflection, although a rather concrete reflection, not in a domain of situatedness. This type of thinking is reflected in such titles as " A Technique for Prototyping Directly from a Specification" (Tavendale[26]), and arises from traditional, formal specification methods, where programs are proved or verified.

Whereas Wasserman does not deal directly with the qualifications of the designers, other authors are more directly concerned for this. Martin[27] in his survey of 4th generation methods deals with prototyping by means of 4th generation methods as means for non-professional programmers (users) to program themselves. He also deals with this type of prototyping as means to reduce the needed

qualifications of the professional designers, which places in two groups distinguishing between DP analysts and programmers. Furthermore, prototypes are means to improve the efficiency of design. The latter perspective is also the background for the work of Kuvaja[28], in which he, among other issues, has compared the time for programming of specific functions, applying different application generators.

A different perspective is represented by Floyd[29], as well as by other members of the German software engineering community. Starting out from the traditional view, called product oriented, she argues for a wider interest in the environment in which the design is taking place. She says that with the process oriented view, quality, as an example, must be discussed from the following point of view:

• *Quality is associated with processes of using the product.*

• *Quality characteristics refer to the reliability, efficiency, etc, of program use; they can be influenced by changing the work situation.*

• *Quality is determined by evaluation (argumentation, trial use, critical appraisal).*

• *Quality is defined by looking from the users to the program (e.g. 'relevance', 'suitability, 'adequacy').*

The distinction between the above approaches, thus, is not only whether users' hands-on experiences are involved or not, and whether the learning of users is involved in the considerations. The distinction is reaching far further: Whether the product of design is conceived only as a technically sound computer system, or whether the product is a changed organizational setting, in which the new computer application has its place.

With a human activity approach in mind, the latter is of course what I am after, rather than just efficiency of design, or technical soundness. As most of the practical tools and methods accessible today belong to the first category, we cannot just stop here. Instead we need to go further into a study of details and technical aspects to find out to what extent some of these tools and methods can be applied with the human activity perspective.

5.2. Basic assumptions

The basic assumptions are what the school or method places itself upon without questioning them. They concern the type of product being

designed, e.g. whether this is a system, a dialogue partner, a tool, or a medium (Bødker[1]). Furthermore, such aspects as the assumed qualifications of the designers, and whether we deal with a process or a product oriented approach (Floyd[29]). An important assumption to focus on is the assumption concerning the tacit character of some parts of human competence: does the approach assume that all aspects of practice can and ought to be made explicit as basis for design, or does it attempt to treat some aspects as tacit? In the light of the title of this paper, one must also look at whether the design process is looked upon as a collective or an individual process. Just to mention some important assumptions.

Many of the methods that deal with prototyping from specifications has inherited, from formal program verification, the concern only for the correctness of the program in technical terms. This means that the role of the program or application in use is not as such of concern. At the same time, exactly this view puts the user in a position as input-generator for the programs; a typical systems perspective, where it is important that the user generates the, from the computer program's view, correct input at the right time. In terms of the human activity framework, the human repertoire of operations is reduced to exactly one way of doing things in each well-defined situation. With the according product oriented view, design is not looked upon as a collective process, but at best as a process where the user and the designer exchange points-of-view at certain, well-defined checkpoints.

As already mentioned in the previous section, the reason for hands-on experiences of this type of approach is a matter of a faster way of making explicit the parts of practice necessary in design.

Tanner and Buxton[30] list, in their discussion of the approach of Wasserman and others (the so-called User Interface Management Systems) the common basic assumptions of these approaches:

- that the user interface of an application can be isolated from and designed after the functionality,

- that the ideal method render all dialog styles equally accessible.

- that the method will render complex interfaces more maintainable, and facilitate portability.

- that the user interface design is inevitably intertwined with its implementation, testing, and evaluation.

User Interface Management Systems, thus, build on the assumption that user interface design prosper from a separation from the design of the rest of the application, although it is part of an iterative process where a sequence of prototypes is constructed and evaluated. This means that design of how the computer application, is to be used is something which can be placed in a single phase or activity. Tanner and Buxton point at some critical questions by asking:

• *Is there a point where the separation of the user interface and the semantic functionality breaks down? How can semantic feedback, for example, be adequately dealt with in a methodological way?*

• *Do the systems really push back the complexity barrier and make user interfaces easier to implement, test, and maintain?*

• *The modules provided in a User Interface Management System will affect User Interface style through the bias of the path of least resistance. How can we exploit this bias to encourage a preferred and appropriate style of interaction?*

Due to the difference in epistemological interests between this type of approach and the "least-programmers'-skills" approach of Martin, there is a difference in the assumptions about the qualifications of the programmers in the two approaches. Whereas it takes well-qualified programmers to make specifications from which a prototype can be derived, the ideas of the "least-programmers'-skills" approach is to get to a situation where programs can be made without qualification demanding advanced structuring, specification, and programming. This in turn also forces a certain perspective both on the product of design, and on the future use situation (see below). Martin does deal with design as a collective approach where more groups of users and designers are involved. He sees design as a rather mechanistic approach which could, by better design means, be conducted by people who have no design competence.

Floyd is viewing the product, the computer application, as something which is under the complete control of the user: " *The use of programs should be intelligent; 'intelligence' refers to competent human dealing with open situations, complex needs and changing goals. Programs should support human decision-making according to actual needs and unforeseen events.*" This entails viewing the computer application as an artifact in many respects: a tool or a media, depending on the purpose of the use of the application. The aspect which is lacking, from the human activity point of view, is that she does not explicitly deal with the transparency of the computer application. She does not

explicitly use a framework by which she can say that the important thing about an artifact in use is that it is not noticed by the user, but operated through operations.

Floyd sees the design process as a collective process, where prototypes as well as different descriptions are means of communication, and where there is a continuous learning process going on, both on the side of the users, and on the side of the professional designers.

Compared to human activity view, this again comes close. Floyd is inspired by the ordinary language school (Wittgenstein[31]), which has also been an inspiration for my own framework. According to this tradition, we can understand human ways of life as language games. Language games are the way in which we interact with the world. They are called language games because just like other games they are played according to certain rules. To take part in a language game we must know these rules. To know does not mean to be able to state or make explicit, but that the other "experts" in the use of the rules accept us as equals. In our intercourse we can change these rules too, as we move along. We get to know the rules of a certain language game, because it has family resemblances with rules of other games that we already know how to play. The major difference relates to differences in the view of practice. In my, material, conception of practice, this is deeply rooted in human beings intercourse with the physical world, as well as with each other. In Floyd's world the thinking of practice is much more idealistic/individualistic (see discussions in Bødker[1]).

5.3. Fundamental concepts

The fundamental concepts deal with the basic 'unit' of thinking of the school or method, e.g. information, information flow, documents, etc.

As the ideas of the "least-programmers'-skills" approach is to get to a situation where programs can be made without advanced programming, this type of approach aims to provide a structure of the prototype as well as simple mechanisms, by means of which the prototype can be build. Martin[27] mentions the different components of 4th generation languages (query language, report generators, graphics language, data base, screen editors, etc.) which enforce a certain structure on the prototype/computer application. It is, for instance, fundamental that what can be fields to be edited or typed in on the screen are the fields that relate to the fields in the database records, etc. Furthermore, 4th generation languages do not deal with general programming language concepts, but with more application directed concepts, such as records of a file, documents, etc. This is

certainly a help when one keeps well within these concepts, but it also restrains the application domain, and reduces flexibility of the design process as well as the types of product.

The formalistic approaches apply traditional programming language-like concepts, perhaps combined with graphical means, such as transition diagrams (Wasserman[22, 23, 24, 25]). As in the case of Wasserman, these graphical means are used to check consistency, at the same time as they are the basis for deriving the actual prototypes. Again this type of framework puts a certain structure to the product, and the way we can deal with it in design. As opposed to the 4th generation languages, these are not application specific, i.e. they do perhaps not enforce a similar type of perspective on the application domain. On the other hand, it does enforce a thinking of the application domain and the application in terms of transitions, data base operations, and the like.

It is of course valuable to base a prototype on verifiable concepts, and it is also valuable to have the application specific concepts of the 4th generation languages at hand. To get to a situation where prototyping can be done together with the users, we must, however, go further. We must get to a situation where the computer environment, in the hands of a skilled computer expert, can be a flexible help to build prototypes rooted in the practice of the future users. This means that we need support for application specific components such as documents. At the same time these components need to be changeable to fit the exact needs of the situation. In the hands of a skilled computer expert flexibility, and a wide repertoire of components, is more valuable than structuring support and easily handled components. The idea of the formalistic approach to be able to technically verify or test the prototype, seems to be valuable as long as one only verifies the technical prototype. Attempts to verify on the use situation as a totality is not possible in the same way, because it is not possible to focus on breakdowns, triggered in the meeting between the future user and the computer application, the materials, and human beings around the user. And in the next step the extent to which she is capable of forming a repertoire of operations to be applied in the use activity.

In the above approaches there has been made no systematic attempts to experimenting or assessment. The ways of setting up use situations for users to get hands-on experiences are rather ad-hoc. The literature on prototyping give very little support for this type of considerations. Card, Moran and Newell [32] deal with different models for evaluation of user performance including one based on prototyping. Regrettably, they do not go into further discussions about how the

prototypes are to be set up. Neither do they give any details about how they suggest that one evaluates user performance based on prototypes.

5.4. Solving practical problems

The solving of practical problems can be a matter of the sequence of prototypes, and the extent to which they cover the same or different aspects of the computer application or the use situation: horizontal or vertical prototyping is one distinction often made – the distinction between whether prototypes are made to examine a major part of the future use, although not in depth, or a small part of use in depth – or; versioning, experimental or exploratory prototyping are other words used for this (Floyd[17], Kammersgaard[33]), to stress different strategies for an iterative process. Also the type of computer support applied in the process, and other means of design fall into this category.

The literature about prototyping falls into two categories when we get to this level of detail about how to do prototyping. As a first group, many case studies of how prototyping was done in specific situations has been reported (See several contributions in Budde[34]). Most of these do not have a theoretical background, and they are very ad-hoc in the way the process has been structured, which tools have been applied, etc. The second group are papers about theoretical /philosophical reflections about the prototyping concepts (e.g. Sol[21], Kammersgaard[33], Floyd[17, 29]). These often contain frameworks by which prototyping can be conceived, and perhaps also recommendations for the process at a more overall level. What they do not contain are detailed discussions and recommendations about when to do horizontal and when to do vertical prototyping, etc.

Martin[27] recommends that 4th generation tools are used as basis for requirement specification (the prototype is the requirement specification), or that they are used to create versions of the computer application. In such situations he finds no need for professional programmers, as opposed to the requirement specification situation. But even Martin does not give any recommendations to when to choose one type of prototyping for another, and how to organize the work.

With the theoretical framework in mind, it seems that exploratory prototyping goes well with the ideas of the design process as a learning process, because it allows for an early exploration of alternatives. There seems to be some need, too, for experimental prototyping and versioning to help the users get as much experience as possible with the

future use situation. How to, closely, simulate the future use situation, and how far we can get are still research issues.

The organization of the project is important. We need to achieve not only a learning process on behalf of the users, but also a mutual learning process, where the computer experts learn about the application domain. Again there is not much help to get from the prototyping literature. Experiences from the UTOPIA project (Ehn and Kyng[8], Bødker[1, 11], Ehn[7]) and the Florence project ([16]), show that a valuable approach is for a group of users and specialists to work together over a long period of time. Furthermore, not only prototyping, but also visits to work places, trying out other computer applications, etc. are useful ways to get the work going.

As for the computer support needed, we have seen above the two types which are today dominating the thinking. One originating from the application domain but being inflexible, limiting both how the product can appear to the users, and the technical possibilities of realizing certain visions. Another applying a totally technical view of the prototype, not taking into account the view of the users, leaving this issue eventually to the computer experts. As there seems to be no good computer support for our purpose, an obvious thing to do to is to elaborate on the ideas of the 4th generation tools, but having in mind that they are to be applied by computer experts, i.e. also to look into the ideas of programming environments. Also, along the lines of the UTOPIA project, there are reasons to ask if all prototypes need to be computer based. The final thing that there are reasons to explore further, as a research issue, is the relation in sequence of prototypes as sequences of technically, or structurally alike computer programs: it is clear that what prototypes need to have in common (with each other and with the final application) is **how** they are used in the (future) use situation, not whether one is programmed in Pascal and another in Lisp. On the other hand, the structure may limit the aspects that can be experienced by means of the prototype, for which reason several differently structured prototypes may be needed. .Are there other ways of making computer-based prototypes, and which computer support is needed for this?

At present, many commercially available products are being released, which seem to be able to support this process, but the market and their actual usefulness is difficult to overview. Furthermore, there is a need to explore the issue of a new view of design and its practical implications. With there two aims a research project has been set up. (See Bøgh Andersen[35]):

The purpose of this project is to do research on computer support for the early phases of systems design, where it is important to rapidly create and experiment with illustrations of how a future computer application will appear to the users, with respect to the interface and functionality, as well as the organizational aspects of its use.

This project on cooperative design is based on the development and use of an Application Simulator. The Application Simulator is intended to improve the communication among designers and users about future use situations, based on practical hands-on and organizational experience using the Application Simulator.

6. CONCLUSIONS

The main conclusion of the above discussion is, that although we tend to talk about prototyping as one thing, we can not from an epistemological point of view claim that all prototyping approaches have the same aims. This advocates the necessity of distinguishing between different types of approaches. *Cooperation between users and designers, the mutual learning process, and hands-on experiences to reveal the triggering of proper operations is only one side of the epistemological interests behind prototyping.*

At the same time, the practical solutions of different approaches are to some extent applicable despite the very different epistemological interests. It is a open question, however, whether to include rationalization of programming work, and technical verifiability in an approach based on mutual learning and on an understanding of the future computer application in use. This means that although we can learn from e.g. 4th generation tools it is perhaps a good idea to reconsider the concept from the perspective of cooperation between users and computer *experts*, who themselves possess a practice which includes a practical and theoretical understanding of the computer support to be applied in the design process, ways of organizing the design work, and of making the users active in design. Furthermore, these experts possess knowledge about the possibilities and limitations of computer applications in general, which is also important in the process. One of the problems is that most means, e.g. 4. generation tools can be used to get close to the use situation, but they are too inflexible in the hands of professional designers.

Being a trained computer scientist, I will of course argue that technical soundness is important; that it is important for computer experts to reason about correctness and efficiency, also in technical terms. What

we must not forget is that *technical soundness does not equal applicability*.

It is not of the user's primary concern whether the programs underlying the application are correct or efficient, but that the application works for them in the use situation. This includes that they can form and use the needed actions and operations towards the materials that they are dealing with and the human beings with whom they are communicating, and that the computer application is transparent, i.e. that it can be used through operations in regular use situations: What is of concern of users is the extent to which they can handle the material they are working with or communicate with other human beings through the computer application, without this causing breakdowns. Furthermore, the other conditions of the use situation, such as the organization of work.

This leaves the problem for the computer expert, with the types of approaches known today, that she must be able to assess the implementability of the prototype. The reason for this is that if we go for prototyping environments that are primarily meant for supporting the users' demands in the use situation, we move away from a step by step realization of the final product. Structurally, and in many other respects, the prototype can be very different from the final application. The prototype serves as a requirement specification, and to serve as a requirement specification it must of course have some realism to it – it is important to go for something in design, which on the one hand is as ideal for the purpose of the users as possible; on the other hand, to be a useful requirement specification it must have some amount of realism to it, concerning the possibilities and efforts of implementing it.

Instead of the discussed practical approaches to prototyping it seems that we must go for solutions which require more technical, as well as social competence from the professional designers. And for computer support which is flexible in the hands of these designers, not for computer support which is primarily meant to structure the design process and the product. These ought to be support which provides good facilities for rapid prototyping in a specific setting, at the same time as more time-consuming prototypes can be build when needed, and the components to be applied in a specific setting can be set up

7. REFERENCES

1. Bødker, S.: **Through the Interface – a Human Activity Approach to User Interface Design**, DAIMI PB-224, University of Aarhus, 1987.

2. Leontjew, A. N.: **Activity, Consciousness, and Personality,** Prentice-Hall 1978 (translated from Russian).

3. Leontjew, A. N.: **Problems of the Development of the Mind,** Progress Publishers 1981 (translated from Russian).

4. Dreyfus, H. and S.: **Mind over Machine**, The Free Press 1986.

5. Winograd, T. and C. F. Flores: **Understanding Computers and Cognition: A New Foundation for Design**, Ablex Publishing Comp. 1986.

6. Suchman, L.: **Plans and Situated Actions: The problem of human-machine communication**, Xerox ISL-6, 1985.

7. Ehn, P.: **Human Centered Design and Computer Artifacts**, Aarhus forthcoming.

8. Ehn, P. and M. Kyng: **A tool perspective on design of interactive computer for skilled workers**, in M. Sääksjärvi, ed.: **Proceedings from the Seventh Scandinavian Research Seminar on Systemeering**, Helsinki 1984.

9. Bødker, S. and K. H. Madsen: **More or Less Systems Description**, in Lassen, M. and L. Mathiassen, ed.: **Report of the Eighth Scandinavian Research Seminar on Systemeering**, University of Aarhus 1986.

10. Bødker, S. et al.: **A Utopian Experience**, in Bjerknes, G. et al., ed.: **Computers and Democracy – a Scandinavian Challenge**, Gower 1987.

11. Bødker, S: **UTOPIA and the Design of User Interfaces**, in Precedings of the Working Conference on Development and Use of Computer-based Systems and Tools, University of Aarhus, 1985.

12. Eriksson, I.: **Learning Processes in the Context of Using ISs**, in P. Järvinen, ed.: **The Report of the 10th IRIS (Information systems Research seminar In Scandinavia) seminar**, University of Tampere, 1987.

13. Hellman, R.: **A Fictitious HIS-Reconstruct of an Information System (A Case Study)**, in P. Järvinen, ed.: **The Report of the 10th IRIS (Information systems Research seminar In Scandinavia) seminar**, University of Tampere, 1987.

14. Nurminen, M.: **How to work with Paradigms?**, in P. Järvinen, ed.: **The Report of the 10th IRIS (Information systems Research seminar In Scandinavia) seminar**, University of Tampere, 1987.

15. Niemelä, J.: **Informal Organization, Coordination of Work and the Use Situation of ISs**, in P. Järvinen, ed.: **The Report of the 10th IRIS (Information systems Research seminar In Scandinavia) seminar**, University of Tampere, 1987.

16. Bjerknes, G. and T. Bratteteig: **Florence in Wonderland. Systems Development with Nurses**, in Bjerknes, G. et al., ed.: **Computers and Democracy – a Scandinavian Challenge**, Gower 1987.

17. Floyd, C.: **A Systematic Look of Prototyping** in R. Budde et al. [34].

18. Friis, S.: **Prototyping – En ny metod eller ett uttryck för ett nytt synsätt?**, in H.-E. Nissen, ed.: **Systemutveckling – av Vem, för Vem och Hur?**, Lund University, 1984 (In Swedish. Prototyping – A new method or an expression of a new perspective).

19. Bansler, J.: **Systemarbejdets teorihistorie – i skandinavisk perspektiv**, Studentlitteratur, 1987 (In Danish. The History of the Theory of Systems Work – a Scandinavian Perspective).

20. Pedersen, J.: **Karakteristik af Human Factors traditionen,** DIKU 1987 (In Danish. A Characteristics of the Human Factors Tradition).

21. Sol, H. G.: **Prototyping: A Methodological Assessment**, in Budde et al. [34].

22. Wasserman, A. I.: **Software Tools and the User Software Engineering Project**, in Riddle, W. E. and R. E. Fairley, ed.: **Software Development Tools**, Springer Verlag 1980 , pp. 93-113.

23. Wasserman, A. I.: **Software Tools in the User Engineering Environment,** IEEE 1981.

24. Wasserman, A. I.: **USE: a Methodology for the Design and Development of Interactive Information Systems**, in Schneider, H.-J., ed.: **Formal Models and Practical Tools for Information System Design**, North-Holland 1979, pp. 31-50.

25. Wasserman, A. I.: **Extending State Transition Diagrams for the Specification of Human-Computer Interaction**, IEEE Transactions on Software Engineering, vol. SE-11, no. 8, 1985.

26. Tavendale, T. D.: **A Technique for Prototyping Directly from a Specification**, IEEE 8th International Conference on Software Engineering, 1985.

27. Martin, J.: **Fourth-generation Languages. Vol. I. Principles**, Prentice-Hall, 1985.

28. Kuvaja, P.: **Experimental Research on Application Generators in Finland**, overheads from talk at DAIMI, January 1987.

29. Floyd, C.: **Outline of a Paradigm Change in Software Engineering**, in Bjerknes, G. et al., ed.: **Computers and Democracy – a Scandinavian Challenge**, Gower 1987.

30. Tanner, P. and W. Buxton: **Some Issues in Future User Interface Management System (UIMS) Development** in Pfaff, G., ed.: **User Interface Management Systems**, Springer Verlag 1985.

31. Wittgenstein, L.: **Philosophical Investigations**, Oxford University Press 1953.

32. Card, S. K. et al.: **The Psychology of Human Computer Interaction**, Lawrence Erlbaum 1983.

33. Kammersgaard, J.: **A Discussion of Prototyping within a Conceptual Framework**, in Budde et al. [34].

34. Budde, R. et al., ed.: **Approaches to Prototyping**, Springer Verlag 1984.

35. Bøgh Andersen, P. et al.: **Research Programme on Computer Support in Cooperative Design and Communication**, University of Aarhus 1987.