Rapid Prototyping with Fourth Generation Systems - an Empirical Study[†]

By

Kaj Grønbæk

Department of Computer Science

University of Aarhus, Denmark

Email: kgronbak@daimi.au.dk

Abstract

The focus of this paper is on design and evaluation techniques supporting active end-user involvement in Information System (IS) development based on rapid prototyping with fourth generation systems. The paper discusses experiences on the development and use of mainly two sorts of prototypes denoted horizontal and vertical prototypes. The experiences result from an interview study, carried out by the author and two colleagues, in nine Danish development projects.

A central result from the study is that horizontal prototypes, which canbe developed with little effort, have shown insufficient in order to get end-users involved in the system development process. One of the problems is that the end-users have not felt motivation to evaluate prototypes with very limited functionality.Moreover, unexpected iterations became necessary in most of the projects studied although horizontal prototypes had been accepted by the users. In contrast vertical prototypes, which are capable of handling realistic data from the use domain, appeared to stimulate extensive and constructive response from end-users before the final system tests. These observations lead to the claim that the developers should be aware of the tacit knowledge which plays an important role in users work practices. To utilise the users tacit knowledge, the design techniques based on prototyping should involve the end-users more actively, and the evaluation techniques should support testing in a work-like setting early in the development process.

Three proposals on techniques to meet these requirements will be given. The first proposal is aimed at having end-user representatives participating in certain design activities where fourth generation systems are being used. The second proposal is aimed at utilising the potential of simulating functionality behind horizontal prototypes. The final proposal is aimed at performing ongoing evaluation activities in conjunction with design activities.

[†] Grønbæk, K. Rapid Prototyping with Fourth Generation Systems - an Empirical Study. *OFFICE:Technology and People, 5(2):105-125,* September 1989. An earlier version of the paper was presented in a forum on Rapid Prototyping at HICSS-22. The paper is also available as *DAIMI-PB 270,* Computer Science Dept., Aarhus University, Århus, November 1988.

1. Introduction

During the 80's a lot of programming environments for development of administrative information systems have been marketed under names like fourth generation systems and Application Generators (Horowitz et al. 1985). A number of authors e.g. (Canning 1984, Canning 1985, Martin 1983, Martland et al. 1986) and of course the vendors of such environments promise that they will solve a lot of the traditional problems related to system development[1]. The main claim is that fourth generation systems will increase programming productivity and thus give the basis for reducing the long queues of user requests on new computer systems. A secondary claim is that they give better conditions for end-user involvement in system development, thus leading to more acceptable computer systems. But how are these expectations met by the experiences?

The author of this paper was a member of a group that investigated aspects of this question by undertaking an empirical study involving nine Danish development projects during the Fall of 1986. Our main focuses were on how prototyping techniques were applied and how users were involved in the projects. The empirical study was carried out as a series of qualitative interviews with system developers from nine IS development projects[2]. We performed two interviews with each developer. The first interview was an informal interview lasting two to three hours and the second interview was a structured interview lasting three to four hours. We wrote summaries of the interviews, and these summaries were commented and accepted by the system developers. These summaries constituted the basis of our analysis which are documented in (Christensen et al. 1987), a report written in Danish. Several of the participating developers found the report useful, and they used it afterwards as a basis for discussing new working practices.

Prior to the empirical study we had gained some experience, beside theoretical knowledge, in the field of rapid prototyping and system development with fourth generation systems. This practical experience came from two minor development projects. One case involved a project of approximately six man-months, where we developed a rooms registration system together with two end-users from the University administration office, using a fourth generation system called MIMER. The second case involved the development of a prototype patient record system to be used by nurses. In this case we used OMNIS-3[3] on a Macintosh, and we worked together with three nurses over a few weeks.

Before going into the concrete experiences from the projects a short introduction to the possibilities of using rapid prototyping techniques based on fourth generation systems will be given. The following sections presents results from the empirical studies and proposals on how to improve end-user involvement utilising rapid prototyping based on fourth generation systems.

---

[1]Refer to e.g. (Lyytinen 1987) for a discussion of Information System development problems

[2]According to (Patton 1984): ``Informal conversational interview" and ``General Interview Guide Approach"

[3]OMNIS-3 is marketed as a fourth generation system but it is not a fourth generation system in the sense that we describe in section 2

2. Fourth Generation Systems and Rapid Prototyping

The fourth generation systems covered by this study fall into the categories of tools described in (Martin 1984, Martland et al. 1986). The tools are provided on mainframes or minicomputers with terminal access. They are aimed at developing data-intensive administrative applications, i.e. applications for updating, storing, retrieving and presenting large sets of data. In general such fourth generation systems cover the following facilities (cf. Figure 1):

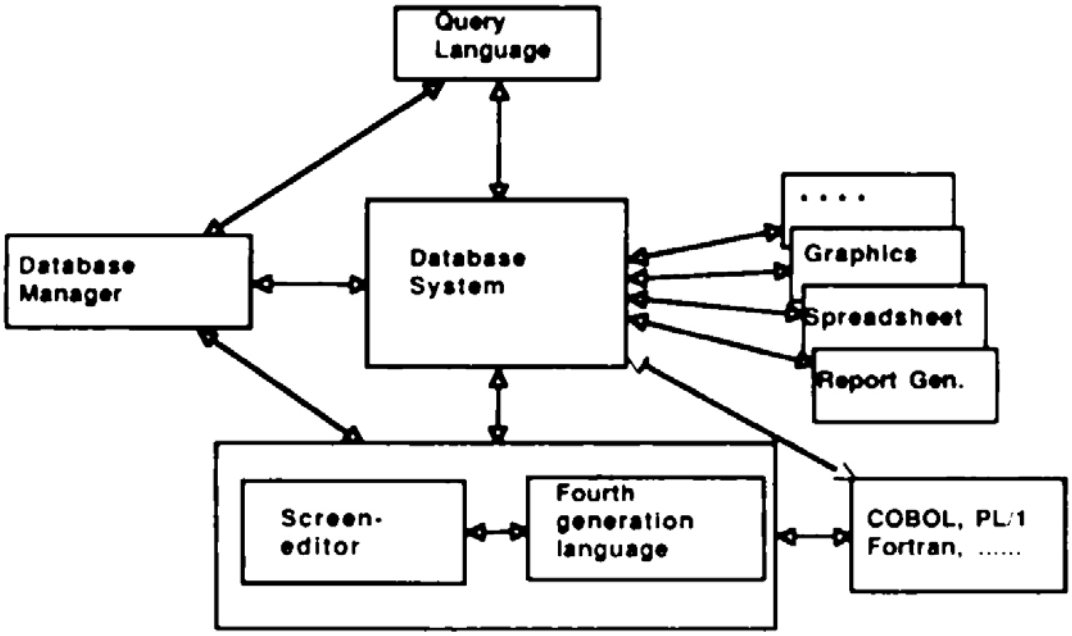

Figure 1: Relations between facilities of fourth generation systems

• A *flexible database management system* (DBMS) including a *Query Language*

• An *interactive screen editor*, which supports design of screen dialogues by specialised drawing and writing facilities. The format of fields for updating or presenting data items is specified interactively by calling a procedure with the cursor in the position of the new field on the screen.

• A *very high level programming language* including specification oriented constructs for database access, dialogue control and report definition[4]. The language is usually both interpretable and compilable. In well integrated fourth generation systems a subset of this language is available as the *Query Language* for making ad-hoc requests on the database.

[4]These concepts refer to fourth generation languages as described in (Martland et al. 1983) or very high level languages as described in (Horowitz et al. 1985)

• A *data-dictionary* for handling information on screen dialogues, program modules and database organisation (metadata). The data dictionary facility has, in a well-integrated system, relations to most of the other facilities[5].

• A few *auxiliary facilities* like business graphics, spreadsheets, statistics, and report generators. These facilities are often aimed at being used by end-users with some knowledge on computers.

• An interface to access program modules written in Cobol, PL1, Fortran, Pascal, etc.

There are, of course, many variations on the quality of the facilities being available with different systems. But the main and common intention is to keep the facilities integrated to constitute a full development environment. The facilities support rapid prototyping mainly by allowing screen dialogues to be designed interactively without traditional programming and the fact that the programming language is interpretable and contains specification oriented constructs. The interpretable language provides a short turn-around time for making tests of the changes to programs.

A final big advantage concerning fourth generation systems, which was stressed by the system developers in the projects studied, is the fact that fourth generation systems support both rapid development of prototypes and implementation of complete computer systems. This capability implies that (parts of) the code from prototypes can be reused for the final implementation.

In this paper the focus is on the capabilities for rapid prototyping, thus we complete the section with a short description of the possibilities of developing prototypes with fourth generation systems.

2.1 Characteristics of prototypes

In the discussion of prototyping with fourth generation systems we find the concepts *horizontal* and *vertical* prototypes (cf. (Floyd 1984)) useful. A horizontal prototype is a prototype, where all the ``visual'' parts of the user interface of a new computer system is implemented, i.e. screen dialogues and their interconnections can be demonstrated, but no data can be processed. In contrast a vertical prototype is a prototype where a few selected functions are implemented in such detail that realistic data can be processed, i.e. a realistic work task can be performed with a vertical prototype.

With fourth generation systems the development of horizontal prototypes requires relatively little programming effort because of the existence of an interactive screen editor where screen dialogues can be ``painted'' and specified interactively on the screen. Vertical prototypes require more effort, but the very high level language, the data-dictionary, and the flexible DBMS supports reasonable rapid development of vertical prototypes, too.

Horizontal and vertical prototypes can also be developed and used in combination where underlying functionality of selected parts of a horizontal prototype is implemented in full-scale as a vertical prototype. The possibility of combining prototypes this way relies on the

[5]This is the reason why no attempt is made to position the data-dictionary in figure 1

modular design of programs developed with fourth generation systems. A general experience from both our own development projects and the projects studied was that one or a few screen dialogues together with the underlying functionality constitute a natural program module when fourth generation systems are being used.

Finally it is possible to simulate the functionality behind a horizontal prototype either by using the fourth generation language and the DBMS or by using special facilities dedicated to performing simulation. Actually MANTIS, one of the systems used in the projects studied, includes such facilities as a record data structure that directly mirrors the fields of the screen dialogues designed by the screen editor. This facility makes it relatively easy to simulate simple storing and retrieval of data items, but the code written for simulation can normally not be reused in the final implementation of the computer system. The issue of simulating functionality will be discussed in details in section 4.2.

3. Experiences from the nine projects studied

In the previous section a brief overview of the potential in fourth generation systems for doing rapid prototyping was given. Now the most important experiences on the approaches to rapid prototyping in the projects covered by our research are presented.

3.1 The projects

A short description of the projects is given, and it is supplemented with a schematical overview in table 1.

The projects were all carried out in development departments which had a close organisational relationship to their user organisations. The computer systems which were developed in the projects had end-user groups in the range from 15 to 10000 persons. In general the project groups consisted of a few selected user representatives, typically a few managers, a single end-user and one to five system developers. In all cases the system developers had the main responsibility of managing the projects.

In seven of the nine projects the developers based their project management on a traditional life cycle model (cf., e.g. (Lyytinen 1987)). In the last two projects an iterative approach to the development was chosen. In seven of the nine projects the fourth generation system MANTIS, was used. MANTIS was here used upon 3 different types of DBMS[6]. In the last two projects NATURAL/ADABAS was used. It happened to be the two projects using NATURAL/ADABAS that followed an iterative approach to the development process. However, we claim that the differences in approaches do not rely on the tools used, but rather on the current ideas of project management in the development departments.

In general the tools in use in the projects were chosen by the departments and *not* by the developers performing the projects. A general statement from all of the interviewed system developers was that their department had bought the new tool mainly to get higher productivity in the development activities. The potential for rapid prototyping were of secondary interest only. Related to the programming productivity issue all of the developers

[6]SUPRA (Relational DBMS), DL-1 (Hierarchical DBMS), VSAM-files

from the projects claimed that their productivity were approximately doubled compared to the use of, e.g. COBOL, and PL/I, when they started using fourth generation systems. According to the developers this productivity gain was mainly caused by the interactive screen editor, which saved a lot of programming effort compared to, e.g. the use of COBOL for implementing screen dialogues. This impact of using the screen editor implied that all of the developers started using the fourth generation systems in the early phases of the projects. However, there were large variations on the use of the tools as we shall see from the following sections.

	Type of project:	Organi-sation:	Tools in use:	Project model:	Number of end-users:
P1	Customer information	Banks	MANTIS/SUPRA	Traditional	3-4000
P2	Authentication system	Banks	MANTIS/VSAM-files and PL-1	Traditional	3-4000
P3	Creditor administration	Electronic prod. comp.	MANTIS/DL-1	Traditional	15-20
P4	Purchase administration	Electronic prod. comp.	MANTIS/DL-1	Traditional	15-25
P5	Case handling	Social secu-rity depts.	MANTIS/DL-1	Traditional	500
P6	Excise duty administration	A local go-vernment	MANTIS/VSAM-files and PL-1	Traditional	50-100
P7	Staff admini-stration	Local go-vernments	MANTIS/VSAM-files and PL-1	Traditional	10000
P8	Health service	National Health offs.	NATURAL/ ADABAS	Iterative	A number of offs. with 10-20 users
P9	Blood banks administration	Hospitals	NATURAL/ ADABAS	Iterative	A number of hosps. with 5-10 users

Table 1: Overview of the projects

3.2 Development and use of horizontal prototypes

In two of the projects, where MANTIS was used, the developers only designed a single screen dialogue with the screen editor during the early phases. The single screen dialogue was discussed internally in the project group. In these cases managers were the only user organisation representatives. This discussion gave the basis for setting up a standard for the rest of the screen dialogues to be used for the new computer system. And the new system design proposal was described purely in a paper-based requirements specification. We do not denote these single screen dialogues as real horizontal prototypes, because they only illustrate a very limited part of the user interface.

In the last seven projects extensive horizontal prototypes were developed during the early project phases, which in most of the projects were denoted by ``analysis'' and ``design''

phases. The horizontal prototypes were prepared with the screen editors by the developers on basis of discussions at project meetings and interviews performed with selected end-users. Consequently no users were directly involved in design activities where a screen editor was used. In each project two to three versions of the horizontal prototypes were developed and presented to the user representatives at project meetings, and in six of the seven projects the prototypes were provided on terminals in the user organisations before the meetings. In two of the projects a single end-user was asked informally to try out the horizontal prototypes and to be responsible of giving his/her comments at the next meeting in the project group.

This approach, however, resulted in very little response from the user representatives. Only proposals on change of details of individual screen dialogues were reported. None of the end-users reactions were directed at new design proposals like e.g. addition of functionality or fundamental changes in the user interface. The developers expressed disappointment on the low response from the user representatives. Despite the disappointment many of the developers interpreted this low response as a silent acceptance of their design proposals.

In five of the seven projects a traditional requirements specification was written simultaneously with the development of the horizontal prototype. The written specification supplemented with the horizontal prototype constituted the final requirements specification. The requirements specifications were in these projects frozen and signed off by the managers to constitute the only basis for the implementation of the new computer system. This acceptance and sign-off on the requirements specification did, however, not guarantee that the new system met the expectations of the user organisation. This aspect is described in more detail in section 3.4.

3.3 Development and use of vertical prototypes

In the last two of the seven projects[7], in which horizontal prototypes were developed, the specification was not frozen on the basis of the horizontal prototype. Throughout these projects an iterative design approach was used. The horizontal prototypes were also accepted by the user representatives with very few comments. But then a few of the most important functions of the new computer system were chosen to be implemented in detail to constitute a vertical prototype. The selected functions were implemented, combined with the horizontal prototype, and provided to a few end-users on their terminals together with some realistic data from their daily work tasks. The prototypes were provided to the users in a test environment on the machine, while the developers were concurrently developing a new version with extended functionalityin a development environment. These new versions of the prototypes with extended functionality were tried out more heavily and enthusiastically by the users than were the simpler horizontal prototype.

In these two projects 20-25 versions of combined horizontal and vertical prototypes were made available in the new end-users usual work place and they were evaluated here, too. Each time the developers had made a new version of the prototype it was made available on the terminals of the selected end-users. Electronic mail became in one of the projects a useful

[7]Projects P8 and P9

medium for communicating evaluation comments and announcements of new versions of the prototype. In both of these projects the selected end-users came up with many and important new proposals for the design of the computer system. Many of these proposals were incorporated in the computer system before it was completed and implemented in the organisation. The final functionality of these computer systems was frozen just a short time before the system was implemented in the user organisation. The last version of the prototype only needed completion of some minor details after the freeze. However, it was difficult for the project group to decide whether the prototypes were eversufficient for the users, because the end-users became capable of making further demands on the continually evolving system. Several of these new ideas that evolved from the iterative design approach were outside the scope of the original project. Thus it was necessary to finish the project, without implementing all of the new ideas, but it was promised that some development could continue on top of the running system soon after it had been implemented in the user organisation.

3.4 Successes and failures of the projects studied

In order to evaluate the approaches to prototyping used in the nine projects, we give a general view of the successes and failures in the projects seen from our point of view.

A general observation in the projects was that the planned deadlines for completion of the computer systems were considerably exceeded in all of the projects. However, the overrun was not worse in these projects than the overrun seen in general in development projects cf., e.g. (Andersen et al. 1986). The overruns were accepted by the user organisations, mainly because of the close connection to the development department, and completed computer systems were delivered in all of the projects except one. The failed project[8] was a very large project running over several years and aimed at a complex user organisation. And it happened to be one of the two projects, where only a single screen dialogue was developed with the fourth generation system in the early phases. In a state where large efforts had been spend on implementing the new system, several end-users were asked to run the final external tests. During these tests it was realised that the new computer system would never be able to fit the current requirements of the user organisation. Consequently the project was stopped without finishing the system, and only a few of the new program modules became integrated in the old batch system that was originally supposed to be completely substituted.

In all of the projects that froze a requirements specification on basis of a horizontal prototype only, unexpected and unintentional iterations became necessary during the late project phases. In two of these projects a new iteration over a one year period became necessary from the results of the final external test, where end-users evaluated the system for the first time since the acceptance of the horizontal prototype and the written requirements specification. One of the problems in these projects was also bad run-time performance of the system. However, the main problem in the projects in general appeared as an instance of a classic problem (cf., e.g. (Lyytinen 1987)) of system development: The developed computer system do not meet the expectations of the users in the user organisations.

[8]Project P7

In the two projects that developed a number of combined vertical and horizontal prototypes, the installation was done with only few problems, because several of the end-users had been working with a prototype much like the completed system a few weeks before the installation of the system. Furthermore the delay of the installation in these projects was not at all worse than the average of the projects that were based on an early frozen requirements specification.

Lessons learned from the projects

One of the most central observations from the studies is that development and use of horizontal prototypes as a basis for freezing a traditional requirements specification after the introductory analysis and design phases does not utilise all of the potentials for rapid prototyping in fourth generation systems. Horizontal prototypes as they have been developed and used in the projects studied do not seem to provide sufficient coupling between the users understanding of their current work and their visions, i.e. ideas and understanding of the future work with a computer system. The users did not adequately evaluate the horizontal prototypes, because of the limited functionality, lack of motivation, and bad conditions. Consequently the horizontal prototypes were reduced to be just substitutes for parts of the traditional written requirements specification. The fact that unexpected iterations became necessary in the projects using this elaborated specification driven approach leads to the lesson that the development of horizontal prototypes does not by itself guarantee the end-users to be satisfied with the implemented system.

In contrast the use of vertical prototypes in two of the projects seemed to stimulate quite useful user response, which could be utilised in making stepwise development and test of a new computer system with ongoing end-user involvement. The vertical prototypes were provided with test data and they were capable of being used in a work-like situation. The users trying out vertical prototypes were then able to reflect on far more aspects of their work with a new computer based system than they were on basis of the more demonstration like use of the horizontal prototypes. The lesson we can learn from that is that the users should be provided with prototypes in a way that they can get a close coupling between the prototype and their work situation.

Although vertical prototypes seem to give the best basis for end-user involvement, the disadvantage is that they still require a large amount of resources to implement. The developers that relied on horizontal prototypes only, said they were not likely to develop vertical prototypes before the requirements specification was agreed on, because there was a great risk that they had to throw them out later on. The possible lessons we can learn from this is that we have to find more efficient approaches to utilise horizontal prototypes, and the concept of simulating functionality behind horizontal prototypes would be worth studying.

Moreover, the user representatives from the projects were not freed from any of their usual work to participate in the evaluation activities with the prototypes. On that background we claim, it is important that the end-users are carefully motivated and provided with better conditions for performing the evaluation of prototypes, e.g. they should be freed from parts of their daily work to perform the evaluation of prototypes.

A final and more general lesson learned from the projects studied is that the introduction of a new system development tool does not by itself solve the main problems of system development like, e.g. mismatch between expectations and the implemented computer system, insufficient user involvement, and delay of installation. In order to reduce the effect of such problems and make new computer systems better tailored for the users needs, it is very important that also working practices in the development department are discussed and adjusted in conjunction with the introduction of a new tool. These issues will be discussed in details in the next section.

4. Rapid prototyping and end-user involvement

In this section we give a theoretical interpretation of the most important lesson from the projects, and we discuss proposals aimed at stimulating active end-user involvement in system development based on the potential for rapid prototyping in fourth generation systems.

The lesson from above that we find most important in relation to rapid prototyping is the obvious need of getting a closer coupling between the development, and use of prototypes and the users ideas, and understanding of their future work with the computer system.

We find it useful to use the concept of tacit knowledge (cf. (Polanyi 1966)) to explain this lesson. The basic assumption expressed with this concept is that people in general base much of their reactions and thereby their daily work on tacit knowledge. Tacit knowledge is knowledge which one uses without any reflection and consciousness. Tacit knowledge appears as intuitive reactions rather than reactions according to certain rules and procedures. Consequently this kind of knowledge involved in users work is very difficult to capture and to communicate through, e.g. traditional interviews and descriptions. A way to let this tacit knowledge contribute to the development process is to stimulate the users hands-on experience with prototypes in work-like settings. The hands-on experience is crucial to the development of computer systems with high quality support for the work of the end-users, because the only efficient way to bring tacit knowledge to the surface is to provoke the use of it in a work-like situation. The concreteness of a prototype will appeal to the users imagination of a future work situation, if the prototype is coupled so closely to his/her current work situation that the tacit, and intuition based knowledge have to be used. Moreover, the hands-on experiences will stimulate the end-users reflections on a future work situation, because a prototype is palpable and the users have the opportunity to point at things which they have not been able to verbalise yet.

It is an experience from Scandinavian research in system development, e.g. (Bødker et al. 87) that rapid prototyping techniques can be used efficiently to provide hands-on experience to users early in a development process and thereby to bring parts of the users tacit knowledge to the surface. But the approaches to Information System development, as seen in the nine projects studied, have not yet utilised the potential of fourth generation systems to provide users with extensive hands-on experience in work-like settings.

In the following we will give three proposals on approaches to provide such important hands-on experiences based on the potentials in fourth generation systems. The first proposal

is aimed at having a few end-user representatives participating in certain design activities where fourth generation systems are being used. The second proposal is aimed at utilising the potential of simulating functionality behind horizontal prototypes. The final proposal is aimed at performing ongoing evaluation activities in conjunction with design activities. The proposals given are based on the study of the nine projects, and literature that covers experimental and iterative approaches to system development, mainly (Alavi 1984, Andersen et al. 1986, Bødker et al. 1987, Boehm 1988, Davis 1982, Lantz 1986).

4.1 End-users participating actively in design activities

First it is proposed to involve a few end-users actively in the design activities where the user interface is designed and where prototypes are being developed. The fourth generation systems support very quick implementation and changing (few minutes) of horizontal prototypes i.e. screen dialogues and interconnections between screen dialogues. Major parts of the development of screen dialogues are done by direct manipulation of objects like texts and fields on the screen, and only a little conventional programming effort is required. Consequently the actual effort of implementing a horizontal prototype is typically only a few days work. It should be realistic to require resources from few representative end-users to participate in such activities, although they may last longer when end-users are involved. The activities where screen dialogues are ``painted" on the screen with the screen editor could to some extent be carried out by the end-users themselves. Furthermore some fourth generation systems[9] provides facilities to design report lay-outs, too, through direct manipulation of fields and text on the screen. In the case of such facilities being available users can to some extent participate actively in the design and implementation of report lay-outs, too. This kind of participation requires that the users get some introductory education in the use of the available development tools.

We claim to get the following immediate advantages from this approach:

• The end-users have the opportunity to get ideas and give comments on the design of screen dialogues and report lay-outs simultaneously with the design of these.

• The end-users get a better understanding of the potential of the technology and the ease of making changes to a design proposal through hands-on experience with the tool.

• New contributions to data analysis may evolve when users are working on positioning data fields in screen, and report lay-outs.

• The representative users will be more motivated to discuss the prototype(s) with his/her colleagues, because he/she has been involved in the design.

• The developers get a better understanding of the users work situation through a very close personal contact during the design activities.

• The users can help in pointing out relevant test data and in setting up tasks to be performed afterwards when more users are brought into the evaluation activities.Of course some dangers can be seen in this approach, too:

---

[9]Examples are MIMER and ORACLE

• There is a danger that the end-users get more focused on the design tool than the work situation he/she should design computer support for.

• The colleagues of the representative end-users can get the feeling that the representatives have been seduced to agree on a bad design.

Despite the dangers, we claim that such an approach based on fourth generation systems can give extensive end-user enthusiasm and involvement in the design and evaluation of prototypes. Furthermore the approach is realistic, if user resources are available. We have ourselves tried out this approach[10] on involving end-users in design activities, and our experiences are promising. The end-users become very enthusiastic, when they discover the ease of making changes by themselves. They start asking questions of the form ``What if I want to have xxxxx, can we try that?". Our experience is that such sessions with end-users bring far more aspects of the users work to the surface than, e.g. having developers performing interviews. Consequently this approach results in better quality of the computer system being developed.

The developers from the projects studied did in general not deny the possibility of involving end-users in the design activities. The main reason given for not using such an approach was that the development organisation relied on an approach using traditional analysis methods with intentional separation in time of analysis and design activities. Another reason given by a few of the developers was that users in general are unable to decide on different solutions, and that the developers own design proposals were the best anyway. However, we do not accept these arguments for separating the users from the design process, because one of the lessons learned from the projects studied was that the implemented systems did not meet the expectations in the user organisation. This lesson gives evidence to the claim that the users participation is the key to the development of valuable computer systems, because of their tacit knowledge which captures major parts of the work performed as mentioned earlier in section 4. Consequently the developers have to be more patient and give the users the opportunity to participate actively. The user organisations also have to provide better conditions for the users participation through, e.g. education and resources for participation.

4.2 Simulating functionality

Secondly we propose that the developers consider how the potential of simulating functionality can be applied to horizontal prototypes. An experience from the projects studied was that it was much easier to motivate the users to try out prototypes with more functionality than provided by pure horizontal prototypes.

Simulating functionality behind some of the screen dialogues from a horizontal prototype could be a way to get more benefit from horizontal prototypes avoiding to spend large efforts on full-scale implementation of vertical prototypes. It is normally not necessary to implement detailed validation rules for data fields in screen dialogues in order to simulate e.g. sequencing of work tasks. The fourth generation system MANTIS actually provides facilities (cf. section 2.1) to be used for simulation without using the more complex and less

[10] In the two cases mentioned in the Introduction

flexible DBMS, but these facilities were not utilised in the projects where MANTIS was used. A major reason given for not using the simulation facilities was that the developers did not expect to get any benefit from using this facility before going into the real implementation activities. Furthermore they did not want to make code which could only be used for simulation and had to be thrown away when the real implementation should begin. However, we claim that some of the unintentional iterations in the projects could have been avoided if the developers had been experimenting more with the functionality together with the future end-users. In two of the projects, vertical prototypes were used successfully to provide the users with some early hands-on experience with parts of the new system. Similar hands-on experiences might have been provided by simulation in the projects were MANTIS was used.

Since the simulation facilities have not been used in the projects, we do not have empirical evidence ourselves on the benefits of vertical prototypes versus simulated functionality based on fourth generation systems. But from other empirical studies we have strong evidences showing that developers can gain important experiences from using facilities for rapid prototyping that are not directly aimed at final implementation of the new computer system. Such experiences are described in, e.g. (Bødker et al. 87), where non-computer based tools were used to develop mock-ups as a basis for simulating functionality of an integrated text and picture processing system for graphic workers. The simplicity of the means implied that the graphic workers very easily could act on their own with the tools and simulate different aspects of the work with a virtual computer system consisting of paper, plywood and slides.

A similar experience utilising computerised tools is described in (Thomsen 1987). In this example a Xerox 1108 computer with Interlisp-D has been used to develop a horizontal prototype with simulated functionality for a very complex control system to the Alarm Center of the Copenhagen Police and Fire Brigade. The Xerox 1108 did not have the facilities needed for a final implementation of multi-user databases, and the code of the prototype could by no means be reused in the final system. But it did provide the flexibility needed to simulate a lot of the functionality with small sets of test data. From the developed prototype the users got useful hands-on experience on using windows, menus, and mouse to perform the alarm controls. These hands-on experiences made the users extend their understanding and imagination of the new control system considerably during the design process. This process of simulating the functionality of the control system lead to a common understanding between users and developers, which the developers claim would have been impossible with traditional interviewing and description techniques.

However, the facilities of fourth generation systems are not in general aimed at simulating functionality. And the area of simulating functionality of complex computer systems as a basis for giving users hands-on experience is not deeply explored yet, and there is a need to study simulation techniques and to develop better tools to get benefit from simulating functionality in rapid prototyping. Currently the author is participating in a research project (cf. (Christensen et al. 1987)) where such development tools and techniques for use of graphical workstations in administrative settings are being investigated.

4.3 Organising experiments based on prototypes

The third proposal is aimed at involving several representative users in ongoing evaluation activities based on prototypes. The projects studied give evidence to the claim that it is too late to spend all the efforts on testing in the final phases of a development project, when everybody expect the new computer system to be complete. We propose an approach where rapid prototyping is utilised more consciously throughout projects in a series of experiments. The main reason to propose setting up experiments is that the utilisation of the prototypes developed in the nine projects seemed too arbitrary. The developers used too little effort on motivating and setting up appropriate conditions for the users to evaluate the prototypes. Another reason is that normally only a few end-users can participate in the design activities as proposed above. But we need a setting for involving several end-users in evaluation activities based on hands-on experience closely coupled to their daily work.

The proposed experiments should be carefully organised similar to external test activities performed at the final stages of traditional system development projects. Based on the problems seen in the projects studied we have got an understanding of issues which are important to consider when experiments on prototypes have to be organised in IS development. Another source of inspiration is the ideas of Boehm (Boehm 1988) who proposes to organise system development based on a spiral model. The development process is then seen as a series of cycles where each cycle consists of activities aimed at reducing discovered risks or uncertainty factors as they are denoted by Davis (1982).

We present the proposal by setting up a set of issues to be considered when organising experiments to constitute the individual cycles of the spiral project model. The issues are referring to selected problems seen in the projects studied, and a set of questions to be discussed and clarified prior to each experiment is given[11]. The proposal is stated this way rather than by giving strict guidelines, because each project is an individual setting, and we do not believe that general guidelines to cover all settings in projects, can be given[12].

4.4 Important issues on organising experiments

The issues are grouped under the following headings: *Purposes, Extension of prototypes, Selection of participants, Preparation of participants, Setting of the experiment, and Evaluation criteria.*

Purposes

In the projects there were very little consciousness about the goals of the prototyping. In two of the projects[13], however, the developers had freedom to choose an iterative approach for the projects, and one of the reasons they gave for preferring an iterative approach was that they felt uncertain about understanding the tasks performed in the user organisation. Inspired by these projects and the ideas from (Boehm 1988, Davis 1982), we claim that a fruitful way

[11]Or prior to each cycle in the development spiral according to Boehm (1988)

[12]This belief is argued in detail in (Andersen et al. 1986)

[13]Projects P8 and P9

of facing the purposes of experiments is to analyse uncertainty factors affecting the design activities. Analysis of which factors contributes to high uncertainty can help determining where most efforts on experiments with prototypes should be spend. Davis (Davis 1982) gives some examples on uncertainty factors that it is worth considering when starting a new development project. On that basis we propose that a qualitative evaluation of uncertainty is done in advance to all experiments or cycles of a project. Examples of important factors that can cause high uncertainty are: the users ability to formulate their requirements[14], the developers ability to understand the users requirements, the complexity of the proposed computer system, constraints related to the tools being used, the organisational context of the future system, and the stability of the use domain.

Concerning purposes of an experiment there is an important distinction between *creating visions on desirable features* and *evaluating the adequacy of a proposed solution*[15]. The priority of these contrasting purposes should be clarified, when organising experiments. This distinction on purpose determines by which means and to what level of detail prototypes should be developed. When we consider the purpose of the use of prototypes in the projects studied there has been a strong bias towards evaluation of a fixed solution only. It has not been considered to provide alternatives to the users or to extend the users imagination early in the process before going into the targeted prototyping of a single proposal.

Another secondary distinction is between *experiments with modification* and *experiments for evaluation only*. If the design activities with end-user participation mentioned in section 4.1 are considered as experiments these would be experiments with modification. In experiments for evaluation only the prototype is kept unchanged during the experiment.

To summarise we claim that at least the following questions concerning purposes should be on the developers mind when organising experiments: *Which uncertainty factors should be reduced by the experiment? To what degree should the attention be on creating visions about the new system? To what degree should the attention be on evaluating the adequacy of an existing design proposal? To what degree should modification of prototypes be allowed during the experiment?*

Extension of prototypes

The prototypes should be developed according to the purposes of the experiment. fourth generation systems give several options on the extension to which prototypes can be developed. But only a very limited set of these options has been considered in the nine projects. The possibilities of providing alternatives and simulating functionality was in general not considered in the projects studied. Although some literature like, e.g. (Lantz 1986) recommends to select between alternatives based on paper analysis only, we believe that early experiments can be useful in choosing between alternatives. Mogensen (Mogensen 1985) describes a case where alternative prototypes are provided with a fourth generation

[14]Cf. the assumption on tacit knowledge earlier in section 4

[15]Cf. (Floyd 1984) the distinction between prototyping for exploration and prototyping for experimentation

system. The two alternatives that were provided helped the user organisation in analysing the needs, and in choosing between the development of a large computer system for a mainframe and the development of a smaller system running on a number of PC's.

Consequently we recommend that all of the potentials of fourth generation systems are considered in relation to organising concrete experiments. The intention of the following questions is to outline the dimensions of options given by fourth generation systems in general: *Which alternatives should be provided? To what degree should the user interface be implemented? To what degree should functionality be simulated? To what degree should functionality be implemented?*

Selection of participants

When a new computer system has many future users it can be quite hard to select the user representatives. The typical solution that was seen in the nine projects was to choose a few relevant managers and maybe a single end-user. But this approach can be dangerous, because potential problems related to concrete use situations, will be hidden until the system is finally tested or implemented in the user organisation. Evidence from the nine projects points to the importance of involving only a limited number of managers and more end-users with different areas of competence in the evaluation of prototypes early in the projects.

Providing resources to the participants is important in relation to end-users. In the examples from the nine projects the end-users were often supposed to do their daily work in addition to the evaluation activities in the projects. We claim that the participating end-users should be freed from parts of their daily work to motivate their participation in experiments with prototypes.

To summarise we claim that at least the following questions concerning selection of participants should be on the developers mind when organising experiments: *Which end-users should participate? Which managers should participate? To what extent should they participate? How can it be ensured that the end-users get resources to participate in the experiment?*

Preparation of participants

If extensive and constructive response is expected from end-users, then it may be necessary to provide some education or training to the participating users in advance. This issue is of greatest importance when users with no experience on using computers are involved in the experiments. There were a number of future end-users in the nine projects that had no experience at all in using computers. These users would have needed both general and more specific education in order to gain the knowledge necessary to participate in the development process and in experiments with prototypes especially.

Another purpose of the preparation is to make the users understand what a prototype is. It is important that the users do not get wrong impressions on the distance between a very simple horizontal prototype and the final system. Some of the developers from the projects studied reported that it could be difficult to explain to the users that there is a big gap in time between design of the screen dialogues and the implementation of the complete system in the user organisation.

We claim that at least the following questions concerning preparation of participants should be on the developers mind when organising experiments: *What kind of knowledge is necessary for the users to participate in experiments? How can the needed education be provided? How can the users understanding of the limitations of prototypes be ensured?*

Setting of the experiment

It was pointed out earlier (cf. section 3.2) during the discussion of the users low response on prototypes in the projects, that it is very important to couple prototypes closely to the users work situation in order to motivate their active participation and evaluation. To utilise the tacit knowledge (cf. section 4) involved in the work performed by the users, the prototype(s) used in an experiment should be set up in a way that help the users imagine that they are in a future work situation. It can be hard to achieve this imagination on preliminary prototypes, but if a few end-users are participating in the design activities they should help in organising the following experiments with other end-users and thereby ensure the coupling of the prototype to the relevant work situations. End-users involved in organising experiments can propose relevant test data and tasks to be tried out with the prototype(s). They can also help pointing out conditions that might make the participants feel more comfortable about the experiment, because they know their colleagues and the tasks to be performed very well.

An important decision concerns the place of the experiment. The main distinction here is between a *laboratory* in the development organisation and *the users own work place*. Of course the users own work place is the best choice in order to achieve a work-like situation, and we would recommend this choice in many cases. But very preliminary prototypes with little robustness can as a start be tried out in a laboratory. Moreover, the laboratory in some cases can have the advantage that the users feel free from their normal duties and thus in better conditions to concentrate on the evaluation.

Another important point is the developers role in the experiment. On the one hand it is very important that the users feel they are in control of the situation and do not feel themselves being examined by the developers (or managers) during the experiment. On the other hand it is also important that the developers can observe the users working with the prototype, motivate the evaluation and supply assistance if unexpected situations are caused by the prototype. In the projects studied the users were mostly expected to do evaluation on their own and that did not give the expected response to the developers. The only motivation activities performed by the developers were done through presentations on meetings. We claim that the developer must be available and catalyse the experiment from the beginning, but reducing his/her participation as much as possible to an anonymous observer/consultant later on.

To summarise we claim that at least the following questions concerning the setting of the experiment should be on the developers mind when organising experiments: *How can a realistic future work situation be simulated? Which test data can be used? Which working tasks should be performed using the prototypes? Where shall the experiment take place? How should the developer participate in the experiment?*

Evaluation criteria

As it was seen from the two projects where an iterative approach was used, it could be quite hard to determine when the prototyping had given sufficient outcome to continue new development steps or to complete the system. Inspired by (Andersen et al. 1986) our proposal on this problem is to describe a set of criteria for the evaluation of an experiment in advance. By criteria we mean statements expressing expectations and minimum requirements to a certain outcome. The advantage of specified criteria is that they give a basis for measuring the evolvement of products (prototypes) and process (experiment activities) during an experiment. Examples of evaluation criteria for the process could be: *1) ``At least two end-users from different departments should have tried out and commented on all of the screen dialogues.'' 2) ``At least the work tasks A, B and C should be tried out using the prototype''*

The evaluation criteria of course have to be closely determined by the purposes of the experiment. And it is important to let the criteria express contents of the evaluation instead of just setting up a time limit, because uncertainty often makes it hard to predict when a sufficient outcome will be achieved.

The criteria can also be related to collection of certain data during the experiments. The users could be interviewed or be asked to answer questionaries. Furthermore certain operations of special interest on the prototype(s) can be logged automatically.

Finally we claim that it is important to give the participants special areas of responsibility for the evaluation activities in order to ensure a sufficient evaluation. The informal approach to the evaluation that we discovered in the projects studied made it hard to ensure that the prototypes had ever been tested.

Finally at least the following questions concerning evaluation criteria should be on the developers mind when organising experiments: *When is testing and evaluation expected to be sufficient to let new development activities continue? What are the criteria to be used for the evaluation of the process? What are the criteria to be used for the evaluation of the product? Which data should be collected during the evaluation? Who has the responsibility of performing the evaluation tasks?*

5. Conclusion

fourth generation systems give a potential for system developers to develop Information Systems well tailored to the end-users needs based on rapid prototyping techniques. The empirical study, however, shows that this potential is not satisfactory utilised, mainly because of the lack of active end-user involvement. We have argued that active end-user involvement is crucial in order to develop computer systems that meets the needs of the users more successfully than the results often seen from traditional specification driven system development projects. The main reason given is that specification driven system development, even though horizontal prototypes have been developed and used, is unable to capture the tacit knowledge involved in the users dayly work. The implemented computer systems still do not meet the expectations in the user organisation.

To ensure the active end-user involvement the developers must be careful to couple prototypes closely to the users understanding of their work. Evaluation activities with prototypes where end-users get hands-on experience in work-like situations are important in order to provide this close coupling and to capture parts of the tacit knowledge involved.

A more general conclusion is that the potential of fourth generation systems can be utilised more extensively. The system developers have to be aware not only to focus on the tools, but also to develop their own work practice with techniques to involve end-users more actively. Three proposals on possibilities for developing work practices in system development with the goal of stimulating end-user involvement based on rapid prototyping with fourth generation systems have been given.

The proposals on approaches to rapid prototyping with fourth generation systems given in this paper are mainly based on exploratory empirical research. To state more precise and elaborated techniques, the next step is to set up field experiments to investigate the proposals. Currently the author is participating in a partly empirical research project (described in (Christensen et al. 1987)) where some of the proposals especially on involving end-users in design activities and simulating functionality will be investigated through field experiments.

Acknowledgements

I would like to acknowledge the system developers from the nine projects studied for their participation in the interview study. I would also like to thank Søren Christensen, Tove S. Rolskov, and Lars Mathiassen for their participation in the empirical parts of this work. Finally I would like to thank Morten Kyng, Liam Bannon, Susanne Bødker, and Søren Christensen for their useful critique on earlier versions of this paper.

References

Maryam Alavi. An assessment of the prototyping approach to information systems development. *Communications of the ACM*, 27(6):556--563, jun 1984.

Niels Erik Andersen, Finn Kensing, Monika Lassen, Jette Lundin, Lars Mathiassen, Andreas Munk-Madsen, and Pål Sørgaard. *Professionel systemudvikling*. Teknisk Forlag, København, 1986.

Peter Bøgh Andersen(editor). Research Programme on Computer Support in Cooperative Design and Communication. *IR 70, Computer Science Department, Aarhus University*, Århus, 1987.

Susanne Bødker, Pelle Ehn, John Kammersgaard, Morten Kyng, and Yngve Sundblad. A utopian experience: On design of powerful computer-based tools for skilled graphic workers. In Gro Bjerknes, Pelle Ehn, and Morten Kyng, editors, *Computers and Democracy - A Scandinavian Challenge*, pages 251--278. Avebury, Aldershot, England, 1987.

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

Richard G. Canning. Attacking the backlog problem. *Edp analyzer*, 22(12):1 ff., dec 1984.

Richard G. Canning. Speeding up application development. *Edp analyzer*, 23(4):1 ff., apr 1985.

Søren Christensen, Kaj Grønbæk, and Tove Rolskov. Arbejdsformer under anvendelse af 4. generationsværktøjer. *IR 69, Computer Science Department, Aarhus University*, Århus, May 1987. Masterthesis. (Title in English: Use of 4th Generation Systems in System Development).

G. B. Davis. Strategies for information requirements determination. *IBM Systems Journal*, 22(1):1--32, 1982.

Christiane Floyd. A systematic look at prototyping. In Budde, Kuhlenkamp, Mathiassen, and Züllighoven, editors, *Approaches to Prototyping*, pages 1--18. Springer-Verlag, Berlin-Heidelberg, 1984.

Ellis Horowitz, Alfons Kemper, and Balaji Narasimhan. A survey of application generators. *IEEE Software*, pages 40--54, jan 1985.

Kenneth E. Lantz. *The Prototyping Methodology*. Prentice Hall, Englewood Cliffs, 1986.

Kalle Lyytinen. Different perspectives on information systems: Problems and solutions. *ACM Computing Surveys*, 19(1), March 1987.

James Martin.*Fourth generation languages Vol. I*. SAVANT, 1983.

James Martin and Joe Leben. *Fourth generation languages Vol. II*: Representative 4GLS. Prentice-Hall, Englewood Cliffs, N.J., 1986.

David Martland, Simon Holloway, and L. Bhabuta. Fourth generation Languages and Application Generators. *The Technical Press - Unicom Applied Information Technology Report Series*, 1986.

Klaus Viby Mogensen. Afprøvning af systemudvikling med prototyper. *DIKU-report 85 10*, Institute of Computer Science, University of Copenhagen, Copenhagen, Denmark, 1985. Masterthesis.

Michael Quinn Patton.*Qualitative Evaluation Methods*. SAGE Publications, Beverly Hills, 1984.

M. Polanyi. *The Tacit Dimension*. Rutledge & Kegan Paul ltd., 1966.

Steen Thomsen. Experiences with system specification by prototyping. Presented at the *EFISS-87 conference, Roskilde*, Denmark, nov 1987.