1991

EVOLUTION OF FINISHED COMPUTER SYSTEMS: The Dilemma of Enhancement

Gro Bjerknes
gro.bjerknes@avenir.msmail.telemax.no

Tone Bratteteig
University of Oslo, tone@ifi.uio.no

Trym Espeseth
University of Oslo, TrymEspeseth@emailaddressnotknown

Follow this and additional works at: http://aisel.aisnet.org/sjis

Recommended Citation
Available at: http://aisel.aisnet.org/sjis/vol3/iss1/1

This material is brought to you by the Journals at AIS Electronic Library (AISeL). It has been accepted for inclusion in Scandinavian Journal of Information Systems by an authorized administrator of AIS Electronic Library (AISeL). For more information, please contact elibrary@aisnet.org.
EVOLUTION OF FINISHED COMPUTER SYSTEMS
The Dilemma of Enhancement

GRO BJERKNE, TONE BRATTETEIG & TRYM ESPESETH
Dept. of Informatics at University of Oslo
P.O.Box 1080 Blindern, N-0316 Oslo 3, Norway

Abstract
The notion of computer systems as finished products that operate in a stable environment leads to viewing system enhancement as an error correcting activity. However computer systems change due to new requirements stemming from organisational changes and the users’ experience with the system. Because of this, it is more fruitful to view system enhancement as an evolving activity. This is a better basis for balancing the technical and functional quality of a computer system during its working life.

On this basis we suggest that system enhancement should be planned, and that techniques known from system development should be applied also during enhancement. Both development and enhancement aim at adjusting computer systems to changes in usage. By means of a case study we show how our proposals for changes in the organisation of enhancement tasks in the computer department may be realized.
1 Introduction

In this article we investigate the enhancement of computer systems as seen from
the computer department’s point of view. We take a closer look at some of the
problems computer professionals experience in their daily life, and offer a frame
of reference for understanding why they occur. Based on this frame we point to
ways to solve some of the problems encountered.

1.1 The Notion of Maintenance

Maintenance is usually defined to be the activities performed to change a com-
puter system after delivery to the customer or users. Swanson (1976) character-
ises maintenance as three distinct kinds of tasks: corrective, adaptive, and
perfective maintenance. Corrective maintenance means to correct faults stem-
ing from development or previous enhancement; that is construction activities
done in order to fulfill requirements. Adaptive maintenance means to enhance the
computer system to a new or changed technical environment, or to adapt it to
new or changed requirements imposed on the organisation from outside. Perfective
maintenance means to enhance the computer system due to new or changed
requirements from the organisation. According to Lientz & Swanson (1986), per-
fective maintenance accounts for 55% of the system and programming effort used
in maintenance, while adaptive and corrective maintenance accounts for 25% and
20% respectively. Approximately 80% of the resources are used on adaptive and
perfective maintenance. In addition, a large part of corrective maintenance is a
consequence of the other two.

We prefer to use different terms in this article, for several reasons. The most
important reason is that the notion of maintenance is closely connected to the
view that a computer system is a finished product. In addition, the imprecise
and wide definition of the concept hides the basic dilemma of maintenance: the
balance of technical and functional quality of the computer system. Before dis-
cussing the issue in more depth, we need to introduce some concepts:

- Error correction corresponds to corrective maintenance. These are the kind
  of tasks normally considered as maintenance.

- System enhancement corresponds to the terms adaptive and perfective main-
  tenance. These two kinds of maintenance both refer to enhancement of a
  computer system stemming from the environment of the system — even
  if adaptive maintenance particularly addresses technical adaptation of the
  computer equipment.

We have chosen to use terms that give associations to evolutionary system de-
velopment because the perspective that a computer system is a finished product
leads to major problems in system enhancement. Traditionally, the software en-
gineering discipline views computer systems from a product-oriented perspective:
The *product-oriented perspective* regards software as a product standing on its own, consisting of a set of programs and related defining texts. In doing so, the product-oriented perspective abstracts from the characteristics of the given base machine and considers the usage context of the product to be fixed and well understood, thus allowing software requirements to be determined in advance. . . .

Software production is followed by software maintenance, comprising the removal of faults and the adaption to new needs. (Floyd 1987, p.194–201)

Considering a stable and finished computer system implies a (temporal) freezing of the environment of the system. However, the broad computer system environment consists of people using the computer system, of organisational structures, of organisational goals and markets — none of which are stable or behaving in totally predictable ways. ‘Software maintenance’ does not put the computer system in this complex context, thus changes to the system are seen as repairing product failures:

Errors are considered to be an expression of incompetence, competence being equated with the ability for correct use in predefined situations. (Floyd 1987, p. 202)

We elaborate this point further in Section 2.

‘Maintenance’ also makes it difficult to address the dilemma of balancing the functional and the technical quality in a computer system. *Functional quality* refers to how well the system performs in its working area, while *technical quality* refers to how well the system is designed with respect to performance and structure. The two terms denote different, but interrelated, aspects of the quality of a computer system.

The technical quality of a computer system is the basis for the functionality of the system: the computer as a technical device must perform the expected operations. On the other hand, there is no use having a computer system with a high technical quality if the functionality does not fit the work tasks the computer system is supposed to support.

As the context of use of the system changes, the users experience a decreasing functional quality, resulting in new functional requirements of the system. Enhancing the system may, however, affect the technical quality of the system. Enhancements require time for the enhancers to comprehend the system (or system parts) which they are making changes to. Insufficient time may lead to the isolation of changes, to ‘quick and dirty code,’ and to unexpected errors. Different requests for changes may oppose each other. It is difficult to change the functionality of a computer system if the technical quality is so poor that the effect of even minor changes in the code cannot be predicted.

In order for enhancement to extend the lifespan of a computer system, both functional and technical quality must be considered. Focus on functional qual-
ity ensures a longer working life of the system, whereas if the technical quality deteriorates, the lifespan may be reduced.

The rest of this section describes a fictive case based on real life: the enhancement of the Customer Service System in a bank. In Section 2 we discuss the problems caused by conceiving computer systems as finished products in a stable environment. Section 3 discusses different ways of approaching the problems. We argue that neither reverse engineering nor quality control can solve the problems encountered. We conclude that an evolutionary system development model seems to be useful. In Section 4, evolutionary system development is discussed in more detail, and we apply this way of thinking to the bank case.

1.2 The Customer Service System — An Example

We will illustrate our views by looking at a bank with approximately 600 employees and approx. 300,000 customers. The bank is a fictional case inspired by Espeseth (1991). The bank has a central administration and 50 branches in different cities. The central administration is divided into several departments which provide the bank with services such as product development and computer services. The departments are responsible to the board of executive directors in the bank. The computer department manager is not a member of the board of directors of the bank, hence he does not participate in the weekly board meetings. Usually he attends the meetings concerning matters of the computer department.

The Customer Service System (CSS) is an in-house developed on-line computer system, which has been in service for 3 years. CSS contains all the customer account information. It is written in COBOL, using a hierarchical database system. The system operates in an IBM mainframe environment. The CSS consists of approximately 300 different programs.

The computer department consists of several groups, which develop and maintain systems used in various application areas. The groups are administrated by the manager of the computer department. Each group in the computer department consists of a supervisor and several analysts and programmers. The supervisor has both technical and administrative responsibility. Both the supervisors and the manager have been recruited from within the computer department. Their work experiences are technical, and they do not have a formal training.

The users of the CSS are located in different departments and cities, and all executives officers in the bank use the system in their daily work. In addition, there is an Expert User Group located at the central administration offices. The Expert User Group gives courses on how to use the system, helps users who have problems or questions, and receives error reports from the users. The group analyses errors. If an error is caused by the computer system and not by a misconception of the functionality, the users are asked to phone or write to the computer department about the problem. The Expert User Group is also involved when the computer department wants to test enhancements or error corrections.
2 The Notion of Computer Systems as Finished Products and the Consequences of This for Enhancement

The assumption of computer systems as finished products leads to two problems as far as system enhancement is concerned: planning and knowledge transfer. Planning is a problem since errors are not supposed to occur, and because the context of usage is assumed to be constant. The problems associated with knowledge transfer between development and enhancement stems from viewing these as separate tasks, thus different people are responsible. In addition, knowledge about the context of usage is not included in the documentation of the computer system.

2.1 Computer Systems Change

After about two years of working with the CSS, the product development group in the bank decided to offer a new service to its customers. If a customer has an account for everyday use, a saving account, and saves minimum 10,000 NOK a year, the customer will receive a discount of 1% on the interest rate of all her/his loans. The purpose of this service was to attract new customers to the bank and to keep the existing ones. The sales people in the bank were in favour of the new product, and the marketing department started an advertising campaign on request from the product development department. The new service was a success, and a lot of customers signed on to it.

To provide this service it was necessary to make major changes in the CSS system. Moreover, the people in the product development group and in the marketing department thought that there should be no problem implementing the product in the CSS system. In fact they did not consider the technical difficulties nor the consequences for other enhancement tasks at all. Unfortunately, the Maintenance Group in the computer department already had a 2 year backlog of requests for enhancements. The computer department got to know about the service at the same time as the customers, and they were furious that they had not been informed earlier. Nevertheless, they were forced to implement the new service. The consequence was delays to other enhancement projects, and they had to use a lot of ‘quick and dirty code’ in order to implement it in time for the date promised to the customers. The ‘quick and dirty’ solution produced errors that troubled the users. The computer department had to use more resources on error correction, which in turn caused further delay to the other enhancements.

The neglect of the interplay between computer systems and organisation has implications for the ability of the computer department to plan the work. When computer systems are regarded as finished products, they are regarded as independent of the organisation. Changes in organisational policy can be done without reference to the computer system. When the computer department is
informed about modifications which have to be made, there will usually be little opportunity to plan the modification due to hard time constraints.

Unforeseen changes in organisational policy make planning enhancements difficult. Moreover, computer systems are often changed because the users want improved functionality. Most of the time these changes consist of adjusting the computer system to the working tasks. Nevertheless, the coupling between work tasks and computer system is seldom recognised. According to Brooks:

all successful software gets changed. Two processes are at work. First, as a software product is found to be useful, people try it in new cases at the edge of or beyond the original domain. ... Second, successful software survives beyond the normal life of the machine vehicle for which it is first written. (Brooks 1987, p. 11)

Computer systems do not operate in stable environments. Organisational changes inevitably lead to revised requirements to the computer system. Hence, the assumption that computer systems are finished products does not hold.

### 2.2 Planning

We have argued that computer systems are not finished products that are used in a stable environment. Nevertheless, enhancement activities are often organised according to a notion of a finished product. Either enhancement activities are not organised at all, i.e. all changes are carried out in-between the “real” work, which is development activities, or enhancement activities are organised as separate from development activities, that is, one group develops and another group enhances the system, and there is no connection between the two. In the bank, enhancement is organised according to the last mode:

*The CSS development project consisted of six members from the computer department and four users from each of the four main business areas of the bank. A consultant with many years of experience from computer projects in other banks was engaged as the project leader. Two of the six members from the computer department had experience from the old Customer Service Batch System. The other participants had experience from carrying out enhancements to other systems in the bank. All the selected project members were to work full time with this project. As soon as the system was finished they would all return to their previous tasks.*

The computer system is regarded as finished and error free at delivery date. Development and enhancement are seen as separate activities with loose coupling. However, the turnover of the system from development to usage did not proceed according to the plan:

*After three years the system was finished. The project was forced to postpone several requests from the users due to time constraints set up by management. These requests were planned to be implemented as soon as the system was finished.*
The development group did not deliver a finished product, and the enhancers had to complete tasks left over from the development group. These left-overs were an unstructured collection of requests — which created a situation that corresponds to organising enhancement activities as in-between tasks. In other words, the turnover of CSS was in conflict with the way in which work is organised, which assumes the system to be finished and error-free when the enhancement group receives responsibility. The backlog started before the system was turned over to production. In addition, new requests for changes emerge. In total, this gives the enhancement group a bad start for planning the enhancement activities of CSS.

The typical flow of requests for changes is informal. The dialogue between users and people in the computer department is carried out in informal meetings or by telephone. The requests concern improved functionality, and the impact these changes may have on the technical quality of the computer system is thrown into the shade. No one has a plan for the changes to be implemented, due to the lack of priority of the changes. The existence of a requirement specification for enhancements is very rare.

Modifications are given priority arbitrarily or according to power, as in the case of changing organisational policy. This confirms Kling (1987) that for some people — those whose requirements are implemented — the computer system may appear as a tool, and for others it appears as an institution.

The way the backlog is handled gives the enhancement group no possibility to consider the overall quality of the system. Their work is reduced to an endless sequence of more or less isolated tasks, and the possibilities for the enhancers to plan and control their own work is also reduced.

In the bank, there are two factors that make planning a problem: the first is that the organisation regards the computer system as a technical device that is independent of the organisation as such. For the computer department, this means that they usually have too little time to plan changes that are a result of changes in organisational policy. The second factor is that the enhancement group treat all requests for changes as in-between tasks, i.e. there are no priorities given, and no evaluation of the requests.

Planning will also be a problem in organisations where enhancement activities are treated as in-between tasks and where enhancement is basically invisible.

### 2.3 Knowledge Transfer

The assumption that the context of use of the system is stable and well understood underlies the notion of a finished product. It is possible to document the system in such a way that a person without previous knowledge can understand the system by reading the documentation. When development and enhancement are seen as separate activities, carried out by different people, knowledge of the system conveyed in the documentation becomes crucial.

Mathis describes the role of documentation like this:
Documentation has always been important for the follow-on users and maintainers, but it is not always realised that it is critical to conveying human understanding of the program. The documentation will have to be coupled more closely with the system itself so that not only current status can be described, but also past history if necessary. (Mathis 1986, p. 710)

Much knowledge is gained through participation in a system development process. Such knowledge is only partially documented, partly due to the notion of computer system as finished products in a stable environment, and partly due to the fact that it is difficult to document even the technical aspects as argued by Parnas & Clements (1986).

In order to be able to realise and change a computer system, one needs knowledge about the complexity and the technical design principles of the system. In short, we can call this technical knowledge. In addition to the complexity of the system itself, system development also requires knowledge of the application area, which is the basis for the functional qualities of the system. This knowledge can be found among the users, and it is seldom documented in any written form.

In addition to these knowledge domains, knowledge of the interplay between the computer system and the application area is needed. This knowledge is related to the use, or the anticipated use, of the computer system, and is normally developed in the project group during a system development process. It is seldom explicitly stated, although the design of the human-computer interface is (or should be) an expression of knowledge of the context of usage.

The complexity of computer systems “creates the tremendous learning and understanding burden that makes personnel turnover a disaster” (Brooks 1987, p. 11). The knowledge gained through participation in system development cannot be fully documented through ‘traditional’ means of documentation. Thus, several people have suggested new kinds of documents and new ways of documenting a system in order to capture the ‘missing’ parts. Naur (1983) proposes using diaries, Oman & Cook 1990 gives guidelines on how to organise the documentation of programs as books, and Soloway et al. (1988) suggest delocalised plans as an example of how to convey the basic structure of programs in a better way.

Parnas & Clements (1986) argue that it is impossible to document a computer system completely, although we should try. Both Oman & Cook (1990) and Soloway et al. (1988) are concerned with the computer system as a product, not the decisions and trade-offs made while developing it. According to Abdel-Hamid & Madnick (1989), this knowledge is what makes the people important in a development process. When development and enhancement are regarded as separate activities, like in the bank, knowledge transfer is turned into a problem. To bridge the knowledge gap between developers and enhancers, documentation about the product and the process is needed. In addition, informal communication is necessary.

In an organisation where enhancement is done in-between the development
tasks, knowledge transfer will be a problem if all the people that once developed a computer system are not available any longer. The alternative is for the developers to carry out enhancement. As the system grows older and more sensitive to changes, and the developer finds herself doing low-status repair work instead of high-status development she may leave the organisation. However, this is a problem beyond the scope of this article.

Knowledge transfer is a problem in enhancement because the understanding needed for doing high-quality functional and technical extensions of a computer system is under-estimated. Documentation which treats this as important can to some extent reduce this problem, but it cannot replace the knowledge gained by long experience in developing and/or enhancing a computer system.

3 Towards a New Conception of Enhancement

The example from the bank indicates that we would benefit from considering the similarities between system development and enhancement. In particular, we should consider all kinds of knowledge gathered and developed in a system development project and how it can be transferred to enhancement. A notion of system development and enhancement in which changes are anticipated will have a better chance of coping with the relation between the technical and functional quality of computer systems. The basis for understanding the need for enhancing a computer system is knowledge in the several knowledge domains involved in system development.

In the following section, we will discuss different remedies that are proposed to mend the mismatch between the current ways of performing enhancements — corresponding to the notion of computer systems as finished products — and the experience that computer systems evolve.

We discuss reverse engineering, quality controlled enhancement and evolutionary system development. We have found that evolutionary system development is the notion that best captures the aspects of computer systems changing. This notion also supports our view that knowledge and understanding are necessary for implementing changes.

3.1 Reverse Engineering Does Not Solve the Problem

There seems to be a growing awareness that enhancements have to be expected. It is also recognised that some of the problems concerning enhancement are different from those which stem from system development. Usually, some people design and implement a system, while others enhance it. Thus, resources for examining and learning about the system must be expanded. In order to address this problem area, a new term has been introduced, namely 'reverse engineering' — as opposed to 'forward engineering.' Reverse engineering "is a process of examination not a process of change and replication" (Chikofsky & Cross, p. 14)
The two most important subareas of reverse engineering are redocumentation and design recovery. As reverse engineering is connected to understanding a computer system, it is a prerequisite to ‘restructuring’, which is defined to be:

the transformation from one representation form to another at the same relative abstraction level, while preserving the subject system’s external behaviour (functionality and semantics) … while restructuring creates new versions that implement or propose change in the subject system, it does not normally involve modifications because of new requirements (ChiKofsky & Cross 1990, p. 15).

Modifications with respect to new requirements not met by the original system, are referred to as ‘reengineering.’

‘Reverse engineering’ is one of many attempts to mend the deficiencies by the notion of computer systems as finished products. Reverse engineering, as presented in IEEE Software (1990), aims at developing better documentation and support systems that may help in grasping the structure and functionality of the systems. This is also outlined in Dhar & Jarke (1988) and in Biggerstaff (1988).

We claim, however, that reverse engineering focuses — and thus maintains — the difference between development and enhancement. In this way, reverse engineering maintains and supports the perspective that computer systems are finished products. Reverse engineering presents a solution to the problem area of deterioration of the technical quality of computer systems. But only the symptoms are cured; the real problem is that the connections between system development and system enhancement are not recognised. Moreover, this problem cannot be solved by isolated techniques for knowledge transfer; also the organisation of system development and system enhancement has to questioned.

3.2 Quality Controlled Enhancement

To take into account the changing nature of computer systems, quality control of enhancement is necessary, as shown in Boehm (1988), Martin & McClure (1983), Osborne (1985), and Schneiderwind (1987). These assume that changes are planned, and that enhancements are organised correspondingly. Thus we find suggestions for organising work similar to those we find in system development, e.g. a change board is recommended, and users participate in decisions on which changes to implement. The change board will have the same task as a project steering group: to discuss what changes are to be implemented if there are conflicting requirements, and to assure an overall technical quality. Usually techniques such as versioning and configuration management, and tools like CASE tools are proposed for keeping track of the system history and making the computer systems easier to understand.

Quality controlled enhancement is a step in the right direction. This approach emphasizes the similarities between development and enhancement, and points
to the necessity of planning and quality controlling changes. This may increase
the technical quality of the computer system.

The drawback of this approach is that the computer system is still regarded
as a product which operates in stable environments. Knowledge that concerns
the changing conditions for using the system, and knowledge about the fit be-
tween organisation and computer system, is not considered. Thus the chance for
implementing changes in an inappropriate way will increase through the working
life of the system, since the system enhancers do not have sufficient knowledge of
the setting in which the system functions.

3.3 Evolutionary System Development

The view that the functional and technical quality of a computer system should be
based on the usage of the system has been discussed in several articles. Bjerknes
& Bratteteig (1984) address the topic as a part of the Application Perspective,
the argument in Floyd (1988) concerns presentation of a process-oriented per-
spective, and Nurminen (1988) presents a humanistic perspective as the basis for
emphasizing usage of systems. The basic idea in all three works is that computer
systems should evolve corresponding to the evolving organisational context of
usage. The emphasis on evolution has important implications for the perspective
on enhancement of computer systems.

Since the work that the computer system supports is naturally evolving, so
must the computer system. This should be reflected in the organisation of both
development and enhancement work. With an evolutionary perspective on com-
puter systems, there will no longer be a need for the present important distinction
between development and enhancement: they will just be different cycles of the
evolutionary computer system development process. Biskup & Kautz (1990) pro-
pose to organise the work in the computer departments according to the necessity
of systems' change instead of forwarding the notion that computer systems are
finished products.

The emphasis on an evolving computer system advocates what in Floyd (1987)
is called a process-oriented view on computer systems and on system development
and enhancement. The perspective that computer systems are stable may be
expressed as product-oriented. The characteristics of process-orientation and
product-orientation are thus another way of expressing the relation between the
functional and technical quality of a computer system.

A process-oriented approach is best reflected in evolutionary system devel-
opment models, like e.g. STEPS (Floyd et al. 1989). In every cycle, a version
is developed and used, and enhancement activities like error correction are per-
fomed during the use phase in a cycle. Larger enhancement tasks are postponed
to the next cycle in which a new version of the system is developed.

A cyclic project model provides a better way of considering the relation be-
tween functional and technical quality than the present way of regarding en-
hancement work. Within a cycle, work can be planned. The plan should com-
prise means for controlling both technical and functional quality of the computer system. Within a cycle, we allow the technical quality of the system to deteriorate if this is due to necessary error corrections or important ad hoc changes. However, these changes can be thoroughly embedded in the next cycle or version of the system in which technical quality is on the agenda. In this way, the relation between functional and technical quality can become less dominating in enhancement work.

There is, however, an important difference between the first cycle and subsequent cycles, concerning the existence of a computer system. Within the first cycle the system does not exist, and the development process must address a vision of a system within a new organisation. During subsequent cycles, the development activities are concerned with modifying the existing system to the current or planned organisational setting of the system. We therefore propose that the concepts system development and system enhancement should cover two different kinds of activities in the system evolution process. System development is succeeded by system enhancement, while system use and system enhancement are concurrent activities.

An evolutionary system development model comprises the attempts of solving the enhancement problems expressed as reverse engineering and quality control. Reverse engineering techniques can be used for restoring the technical quality of the system, and quality control can be used as a means for planning and evaluating an enhancement cycle.

If the similarity between system development and system enhancement is fully exploited, we also get a new repertoire of techniques that can be used in enhancement for making use of the knowledge gathered through the use of the system. Examples are prototyping, mock-ups, and task analysis with users. In addition, the problem of knowledge transfer may be addressed:

programs have to be viewed in the context of the total evolving system... This means that design and use history is extremely important. Trade-off decisions for space and time may alternate many times throughout the lifespan of the system. The program can not be viewed as a static result; it is a dynamic, ever changing entity.
(Mathis 1986, p. 710)

The existing means of documentation can be improved, and techniques for documenting the functionality of the system can be investigated. One example is to let the user manual be the actual documentation of the system as described in Gould (1988). In this way knowledge about functionality and the interplay between functionality and use setting can be updated and documented during the different cycles.

According to a study done by Swanson & Beath (1989), "user demands for enhancement and extensions to application systems" is the highest ranked problem in system enhancement. Other problems concerning use and user understanding
are also ranked high. We believe that such problems will be reduced by introduction of techniques that put functionality on the agenda throughout the whole lifespan of a computer system.

4 Realizing a Model of Evolutionary System Enhancement

In this section we discuss how a model of evolutionary system development and enhancement can be realized. We especially address problems concerned with knowledge transfer and planning of system enhancement. It may be possible to change the problematic enhancement situation in the bank by reorganising the work according to an evolutionary model.

After five years in service, the CSS system contained a lot of errors, and the computer department had a growing backlog of requests for enhancements. Due to complaints from the users, the computer department decided to do something about the situation. They founded two working groups, one for error corrections and one for system enhancements. The Error Correction Group is responsible for the running version of the system and the Enhancement Group is responsible for the next version of the system. The Expert User Group is (still) the link between the users and the computer department.

In addition there exists a Change Board (called the ‘Priority Committee’) in the bank. This Change Board should control the changes of the computer system by giving priority to the requests sent to the board. However, the board does not work. Requests seldom follow the formal channel, so the members of the board have lost interest in it.

At the point of establishing the permanent Error Correction Group and Enhancement Group the error situation was worse than ever. Some users had reported directly to the top executive director. This resulted in a meeting between the board of directors and the manager of the computer department. At this meeting the present situation was discussed, and it was decided which steps to take to improve it. The directors’ point of view was that the present organisation — with the two working groups — had to be provisional.

The computer department has already tried to do something about a frustrating situation by changing the organisation of the enhancement work. They have admitted that computer systems are not finished commodities, but have to be constantly changed. They have organised the work in the computer department according to this. We claim, however, that the organisation given supports a product-oriented view:

- The idea of a Change Board is not working. Requests for changes are not given priority, and the work of the Change Board is not contributing to giving the computer department better possibilities for planning the work.
• The Enhancement Group organises its tasks as projects. However, the
decisions about giving priority to the time consuming modifications are still
non-transparent and arbitrary. The Enhancement Group seems to consider
only the technical quality of the CSS, as the functional quality of the system
is not addressed.

• The Error Correction Group is a ‘fire brigade.’ They are not really getting
in touch with the users, and their knowledge about functionality of the CSS
is not increasing. The knowledge acquired through daily contact with the
system is not exploited in the Enhancement Group.

• The Expert User Group is not representing the users. The group has a lot
of relevant and necessary knowledge, but they cannot really imagine how
the ‘real’ users experience the work with the CSS. In addition they act as a
buffer between the computer department and the users, for better or worse.

• The management insists on the current organisation of the computer de-
partment being temporal, i.e. that the need for modifications of the CSS is
exceptional.

Although the present organisation of work supports a product-oriented view
on the use of computer-systems, the situation is promising. We will discuss how
the group structure described could support a process-oriented view if tasks were
divided otherwise.

4.1 A Scenario of the Bank

Let us imagine the computer department with the same structure, but with some
differences of content. The same basic structure is present: a change board, an
enhancement group, an error-correction group and an expert-user group. We
give the groups tasks that are slightly different from the tasks they have today.
The new tasks are organised in accordance with an evolutionary model of system
development and enhancement. Notice the contrast to the existing product-
oriented organisation of work in the following scenario:

The Change Board

The Change Board is responsible for changes in the organisation that also affect
the computer system, i.e. for the long-term development of the computer system.
In addition, it is concerned with changes that cannot easily fit in with functional
and technical quality. The Change Board consists of managers, in addition to
some people from the Enhancement Group and the Expert User Group. More-
over, the board is a decision-making body The agenda will thus be influenced by
top managers, concerning new policy and organisational changes. The members
of the board represent different areas of knowledge and interests related to de-
cisions about the CSS. The Enhancement Group transfers requests that cannot
easily be implemented in the system to the Change Board, and the group assists in estimating resources needed for implementing changes and in discussing consequences for the functional and technical quality of CSS.

The Enhancement Group

The Enhancement Group is responsible for the next version of the CSS. The requirements and goals for the next version are set by the Change Board, but the Enhancement Group is also involved in the negotiations of the resources needed. The Enhancement Group plans and coordinates the necessary activities that lead to a new version of the computer system. The process is planned as a development project, utilising techniques and tools from the system development practice in the bank, such as the PSO-model (a model for system development projects), the CASE tool Teamwork, the system description technique SA/SD, and the debugger Expediter, and techniques and tools of quality assurance/control, like baselines and checkpoints with walk-throughs and inspections. In this way, the knowledge of the connection between the computer system and its application area is preserved and renewed. Historical knowledge (about why the system is as it is), however, needs new kinds of documentation, aiming at documenting decisions made during the work processes. Diaries, log-books or books of minutes may provide this kind of documentation.

The Enhancement Group estimates the resources needed for realising the proposed enhancements. The group has to participate in giving priority to suggested changes, especially if conflicting requirements are requested or if the requirements are in conflict with the existing architecture of the system. In this case, the request should be forwarded to the Change Board.

The leader of the Enhancement Group has knowledge about the development of the existing CSS, and he also knows quite a lot about enhancements in the system. The other members are selected according to experience and knowledge. The users involved in the group are ‘ordinary end users,’ rather than people from the Expert User Group. Cooperation between the computer professionals and the users in the Enhancement Group is regarded as just as important as user participation in system development.

The Enhancement Group will live through a number of enhancement projects, some even divided into subprojects. All members will after some time personally know the history of the CSS, being educated through experience. The system history will be recorded in the group, by the group members, as a transition from one version to another.

The Error Correction Group

The Error Correction Group is responsible for the functional quality of the existing version of the system. Their main responsibility is therefore to correct errors that are critical to the users. The Error Correction Group consists of system
developers, and at least one of them should have experience from the development of the system or know it well by participation in the Enhancement Group. In this way, the knowledge available among the staff is utilised. This is especially important during the first busy weeks after a new version has been put into action. Knowledge about the history of the system speeds up the error handling.

The Expert User Group and the Error Correction Group should jointly decide whether an error is a misconception of the use situation or a technical defect. The two groups are therefore responsible for creating the queue of change requests which are not attended at once. Some of the requests thus become requirements to the next version of the system.

The group has daily contact with the users who report errors. Therefore the members of the Error Correction Group will have a unique chance to learn about the functionality of the system.

Since the Error Correction Group is a frequent user of documentation of the CSS, the group is responsible for defining a standard for documentation that fits the needs of the enhancers. In addition, they log the reported errors and make statistics of the performance of the CSS in order to find modules/programmes that are obvious candidates for redesign.

The Expert User Group

The Expert User Group does not consist of user representatives any longer. They are consulted by the Enhancement Group concerning both design and acceptance tests, due to their knowledge of CSS and the application area. They are also responsible for appointing 'ordinary end users' as participants in the Enhancement Group. In addition, their teaching load has increased. Due to their specific knowledge, they teach the users about the CSS, both formally and informally.

4.2 What Is the Difference?

The main difference between the existing organisation of work and the scenario is that the structure of the work tasks reflects a clearer distribution of responsibility. However, four different groups with different tasks and responsibilities do not necessarily ensure evolutionary development and enhancement.

We have suggested dividing the work according to responsibility, and giving the groups authority for practising the work tasks belonging to their responsibility domain. In order to adjust the new work tasks to existing individual work styles in the computer department, the groups should be goal-driven. The working conditions within every group are improved by giving the group information that supports planning their own work. However, planning in one group is dependent on decisions and knowledge gained in the other groups. Thus, some structure has to be imposed on the work to ensure the necessary information flow between the work tasks. We have tried to establish different kinds of knowledge transfer to support the cooperation between the groups.
Planning of Enhancement Work

In the bank there should be discussions about the overall evolution of the CSS. The decisions made will be the basis for the planning in the groups which are involved in the enhancement work. We believe that a Change Board may be an arena for such discussions. The Change Board decides, and thus gives priority to the changes to be implemented in the next cycle. Cycles should be limited according to negotiated milestones, as a way of making the evolutionary path explicit and transparent. In this way, it may also be predictable and controllable for all parties involved: both computer staff, managers, and users need to have a certain general understanding of the computer system in order to control their own work situation.

The Enhancement Group plan their work according to the decisions made by the Change Board. Thus the group is responsible for a specific set of activities that typically occur within a cycle: 1) development activities like design, implementation and test, 2) a period of error correction and day-to-day support when the next version is put into action, and 3) activities concerned with evaluating this cycle and planning the next cycle, i.e. the next version of the computer system.

The Error Correction Group cannot plan their work in the same manner as the Enhancement Group. But at least they should have the opportunity to evaluate the requests, pipe some of them to the Enhancement Group, and decide the order of priority of the requests they choose to carry out.

The Expert User Group is really not relevant here as we are addressing the organisation of the computer department. However, the new responsibility makes the role of the Expert User Group clearer: they no longer find themselves at the same time among and between the users and the enhancers.

Transfer of Knowledge About the Computer-based System

We have proposed two ways of improving the knowledge transfer concerning the CSS, (1) by people and (2) by documentation.

- We want to distribute knowledge about the development and use of CSS in all the groups that are involved in enhancing the system. We have done this by proposing the Change Board as an arena for overall discussions in which all interest groups participate, by introducing knowledge exchange between the Enhancement and the Error Correction groups, and to ensure that use is considered as the basis for all enhancement activities, implying a closer contact between the computer department and the users.

- We have proposed reconsidering the documentation standards. The purpose is to secure that the Error Correction Group gets documentation in a form that eases the work. We therefore suggest that they should have influence on the documentation standards, especially the part that concerns
the technical quality. At the same time, the Enhancement Group needs
documentation concerning the history of the CSS. As the group will last
through many cycles, we expect that they are able to find a satisfactory
form of documentation on their own.

4.3 Is It Realistic?

In general, the most important obstacles for realising the proposed enhancement
organisation in the bank are concerned with motivation. One problem is that
error correction is regarded as a tedious and boring task that nobody wants to
do. Chapin(1986) argues that:

both motivation and and personnel qualifications are factors over
which supervisors actually have substantial influence. ... If we seek
to improve application software maintenance in our organizations, the
attitudes held by supervisors demand attention. (Chapin 1986, p. 56)

Swanson & Beath (1989) agree with Chapin, and they propose the following
improvement for IT-management as far as enhancement is concerned:

1. Give it managerial attention.
2. Provide a vision for maintenance.
3. Use information systems for maintenance.
4. Throw some money at maintenance.
5. Insist on a clean turnover.
6. Design a maintenance career. (Swanson & Beath 1989, p. 227-
230)

Our proposals are in line with these suggestions. The establishment of groups
with distinct responsibilities and the power to exercise these responsibilities is a
way of giving enhancement managerial attention (1), and the organisation pro-
vides a vision for enhancement (2). Moreover, this organisation of work con-
tributes to a “clean turnover” — from development into use — (5). Furthermore,
we propose the use of technical equipment like debugging tools, documentation
tools, report generators, and configuration management to support the practical
work situation (3). The introduction of tools for enhancement and the efforts
spent on establishing new work practices imply that money is spent (4). The
only point we have not discussed is the design of a maintenance career (6), but
this point is obviously important in order to increase the status of enhancement
work.

We have proposed an organisation of the enhancement of CSS that corre-
sponds to an evolutionary model for development and enhancement, based on
the use of CSS. Moreover, we think it is possible for the bank to realize this
organisation of work with some managerial effort.
5 Concluding Remarks

We started this article by discussing system enhancement on the basis of the relation between technical and functional quality. At present, the problem in enhancement is that the functionality of the computer system is given priority at the expense of the technical quality. We have argued that the two aspects may be better balanced if we take into account that computer systems are not finished products when planning and carrying out system enhancement work. Thus we see the relation between technical and functional quality as an expression of the relation between a product-oriented and a process-oriented view of the use of computer systems.

Many problems in system enhancement can be understood through the perspective that computer systems are finished products. We have drawn attention to two sets of problems: the first consists of problems concerned with knowledge transfer between development and enhancement activities, and between the enhancement activities themselves. The second consists of problems related to planning the enhancement work, both within the computer department and in the organisation as a whole.

We have suggested an organisation of enhancement that corresponds to an evolutionary model for development and enhancement. Development and enhancement are seen as different cycles in an on-going evolution of a computer system in a continuously changing organisational setting. The evolutionary model explains current problems in enhancement. In addition, the model may be a basis for solving some of these problems. We have used it as a framework for describing how knowledge transfer and planning in system enhancement work may be carried out.

We believe that the changes we have described are possible to realize in the bank as we build upon the existing organisational structure and only suggest changes that do not exert the organisational culture too much. The bank would benefit from a CSS that is better tailored to the work in the bank. This requires that the computer department is able to plan and carry out enhancement work without the technical or the functional quality of the system deteriorating.

Acknowledgements

We would like to thank the members of the editorial board for their constructive comments on earlier versions of this article, and the participants of the 13th IRIS for instructive comments and engaged discussions.

References


