

## **Part II – Five published research papers**

## **Paper A:**

### **Challenges in web development**

Carstensen, P.H., and Vogelsang, L. "Design of web-based information systems - new challenges for systems development?" *Proceedings of the 9th European Conference on Information Systems*, Bled, Slovenia, 2001, pp. 536-547.

# DESIGN OF WEB-BASED INFORMATION SYSTEMS - NEW CHALLENGES FOR SYSTEMS DEVELOPMENT ?

**Peter H. Carstensen and Lasse Vogelsang**

The IT University of Copenhagen, Denmark  
e-mail: carstensen@itu.dk, vogelsang@itu.dk

## ABSTRACT

*The web-technology is going through major changes these years, both with respect to types of systems based on web-technology, organization of the development work, required approaches and competencies, etc. We must rethink the organization of the development work. This requires a deeper and coherent understanding of the nature of web-development. This paper presents findings from a field study undertaken in a web-development company. From these findings we characterize web-application development and discuss some major challenges to cope with and to find proper support for in the future. We found that web-development is characterized by involvement of many expertise groups with little training and experience in information systems design, that some of the existing modeling and communication tools introduce severe problems, and that the pace of the introduction of new tools and features causes development and management problems. A further complication factor is that web-applications of today often are business critical systems.*

## 1. INTRODUCTION

As a basis for designing complex information systems the Web-technology has matured a lot over the last few years. The technology is still fairly simple with a number of unsolved problems, but the advantages and potentials are so significant that most of today's design of information systems to some extent is based upon web-technology. Organizations increase their investment in and usage of web-based technology. The scope of web-based application has grown enormously and has moved to become a platform that can support all facets of organizational work (Isakowitz et al., 1998). Furthermore, the web-technology differs from the traditional information technology in that "it might be labeled as a new type of information system, but [...] it is fundamentally a new medium of human communication" (Turoff and Hiltz, 1998, p. 116).

In parallel with this trend the activities to be IT supported become increasingly knowledge demanding, and the actors are confronted by increasing demands for improved quality products or services, improved complexity of the products and services, higher flexibility, shorter lead-times, etc. To cope with these demands work is often undertaken by large groups including people with different background and perspective. More actors become involved and an abundance of decisions have to be made by mutually interdependent actors. Actors involved in complex cooperative activities need support for communicating, coordinating their activities, keeping track of state of affairs in the field of work, sharing information, etc. To complicate it further an increasing part of the work that needs IT support is conducted in virtual organizations, meaning that the control structure and the spatial and functional arrangements differ from situation to situation (Mowshowitz, 1997). We need a re-orientation of the philosophy of management, IT usage, etc.

The trends mentioned above have serious implications for the development of information systems: New groups of expertise must be involved, the technology is rapidly changing and well established products and standards are rare (Burdman, 1999), the roles of and interplay between the developers and users are dramatically changing. These challenges become even greater due to the fact that most web-application development has been started as ad-hoc based 'quick and dirty' development of small sets of web-pages mainly used for 'toy purposes', information publishing or advertising. To phrase it differently: Many of the at last adapted traditions within traditional software development on careful analysis and modeling, punctual establishment of conform architectures, attempts to estimate, etc. are absent in most web-application development.

In order to come up with ideas, recommendations, tools, techniques, concepts and methodologies supporting the processes of developing web-based information systems we need a better understanding of what characterizes development of large scale web-applications today, who are involved, what expertise is needed, what are the essential problems, etc. This paper presents a first set of findings from an ongoing project that follows design, development and use of interactive web-applications (we stress the interactive aspect of the application to put explicit focus on web-applications that mediate interactions among multiple distributed collaborating actors, cf., DIWA, 2000). We present findings from a field study of web-design undertaken in a web-development company. From these findings we characterize web-application development and discuss some major challenges to cope with and to find proper support for in the future.

Much of the literature on web-design concerns the new potentials, possibilities and challenges for business and user organizations. Examples are articles on the potentials of the new technology in terms of 'perfect communication and information transmission' (e.g., Turoff and Hiltz, 1998), challenges when setting up the e-businesses or changes in the supply-chain relationships as a result of web-based e-business (e.g., Baron et al., 2000). Of course are these aspects essential when discussing web-based information systems. They are however considered out of scope.

The literature on design and development of web-based information systems seems to agree that development of web-based systems is different from development of 'traditional' IT-based systems. Kristin Braa et al. argue that the technology mainly is an interaction medium and that in a www-environment new applications will be developed and assembled by cloning existing components (Braa et al., 2000). Thus the notion of tinkering is more important in web-application development than 'rational' design decisions. From this they argue that the key-words of the information systems discipline will be "prototyping, object orientation, reuse and bricolage, 'quick and dirty ethnography', networking, redundancy, plug-ins, innovations, customer focus and time to market." (ibid., p. 27). Others argue that development of web-sites (a set of web-pages) differs a lot from development of web-based information systems. Tomás Isakowitz et al. state that the latter "supports work, and is usually tightly integrated with other non-WISs such as databases and transaction processing systems" (Isakowitz et al., 1998, p. 79). Web-based information systems are, however different from traditional information systems in the sense that they require new approaches and often are results of grass-root efforts. Some authors also stress the differences in terms of the speed of change in the technological basis. The pace at which the continuous evolvement of tools and features are running at the web-technology is extreme even compared to the rest of the IT-area, and that these tools "have lured people away from recognizing the need for a systematic design approach" (Balasubramanian and Bashian, 1998, p. 113).

The need for new competencies, new roles, new approaches and methodologies, new ways of organizing the development work, etc. has been widely recognized. The aim of this paper is to

contribute to our understanding of the nature of development of web-based information systems. We do this by reporting from a 'real-life study'.

## **2. THE APPROACH TAKEN**

In order to obtain a coherent understanding of complex engineering and development work settings and the work conducted, field studies are essential means (cf. e.g., Yin, 1989; Orlikowski, 1993). This paper is based on data collected in an empirical study of web-application development efforts. The study focused on roles, competencies, communication structures, use of tools and techniques, etc. in a large project designing and implementing a large web-site. Besides addressing activities in the one project in focus we also discussed more general themes related to the introduction of new tools and methodologies in the company. The work setting is described in further detail in the next section.

First phase of the field study and the data analysis was conducted over a period of four months and was mainly based on qualitative interviews (Patton, 1980), and observation and active participation in a number of project meetings. Two months after version 1 of the product was launched, we conducted a second series of interviews with the project participants. These were retrospective reflections on the experiences gained from the project work. Furthermore we dug into documents, models and specifications produced by the project. 10 semi-structured interviews have been conducted.

Our approach had a structured and targeted orientation. Although we did not start out with a strict set of hypotheses, we did bring an articulated perspective. Explicitly we addressed roles, competencies and the interaction between the involved designers. The data was analyzed and structured according to a number of topics that were identified from a first rough analysis. This resulted in a first very descriptive portraiture of the work. Then this was discussed and validated with some of the project participants, and it was presented to and discussed with a large group of researchers who have made similar studies in other web-development companies (see DIWA, 2000 for further details). In the light of the lessons learned from the first phase the data from both series of studies were analyzed. This has been done by going through the interview data, structure it according to the topics identified and a synthesis of the core problems or lessons that can be identified. Then these were written up in a second version of the working and discussed and validated with some of the project participants and the group of research colleagues.

The research approach taken can be characterized as qualitative research heavily inspired by theories and conceptualizations within Work Analysis (Schmidt and Carstensen, 1990) and from other comparable studies of software engineering work (e.g., Carstensen et al., 1995; Kraut and Streeter, 1995; Grinter, 1997).

The purpose of research must be to provide both the richness of detail and relevance of the problems studied, as well as a certain tightness of control or rigor. We do not believe that one empirical effort necessarily needs to encompass both aspects. Of course we do recognize that since the results reported in this paper are drawn from a single field study, we can only make weak claims as to the generality of the findings. We are, however, confident that our results have some validity since there seems to be a large overlap with results found in similar studies by our colleagues in the DIWA project. DIWA is an acronym for Design of Interactive Web-Applications (DIWA, 2000).

### 3. THE WORK SETTING

Our studies took place in an IT-development company (hereafter called the Company). Most of the activities in the Company have to do with development of software for a large Danish pharmaceutical company. The Company develops business systems, production systems, web-based systems, and supports the IT-systems. The Company has developed these systems for the last 30 years. Of course development of web-based systems is of a more recent date. We will return to this issue and some of the problems related to this activity. Our studies have taken place in the web-development department (hereafter called WebSystems) in the Company.

Until recently WebSystems has been a minor department in the Company with 15-20 employees developing web-sites basically for information publishing for different departments in a pharmaceutical organization. The requirements for documentation have been minor and were not taken very seriously. This has changed within the last two years because the developed systems have become more critical to business and the size of the projects has grown significantly. Today WebSystems builds web-applications for e-commerce, document handling, community service, etc.

A number of new employees with various backgrounds have been hired to meet the new and increased requirements, which have emerged due to the new type of developed systems. It used to be mainly engineers that were hired in WebSystems. Today WebSystems hire people with a background in the arts, graphical design, etc. We will elaborate this issue in the following sections.

The size of WebSystems increases dramatically. When WebSystems was established five years ago only with a few people developing web-related systems. Today the number of employees is approximately 70. The number of employees has doubled within the last year and is planned to double again within the next year. This has significant impact on how WebSystems is organized.

Historically the demand for documentation of the software development has been high in the Company because the software is used in the pharmaceutical industry. Errors can cause serious problems for the customers' health and pharmaceutical company's business. Hence, the Company has had a formalized development methodology since 1994, where the Company became ISO9000 certified. The used methodology is based on techniques, tools and standard documents for different activities such as data modeling, and project planning, testing, etc., and includes guidelines for the use of the mentioned techniques, tools, and documents dependent on the project type. The methodology follows the waterfall model. According to key developers and managers in WebSystems this is problematic. WebSystems uses object-oriented technology to develop their systems, which is poorly supported by traditional structured analysis and a waterfall approach. Therefore the existing methodology is assessed as unsuitable for the web-development.

We have followed a specific project in WebSystems, the Zyme project developing a large web-site, including a web-portal with approximately 10 sub-sites, a content management system, and facilities for e-commerce and dedicated community support. The sub-sites address very different user groups, for example investors, the press, education institutions, customers (including possibilities of purchasing enzymes). Thus, the demands for usability and flexibility are high. WebSystems was responsible for the design and development of all aspects of the site, the information architecture, user interface and navigation, technical infrastructure, etc.

18 people were involved in the project, most of them working full-time. The project had a product manager and two project managers, one from Web-Systems and one from the customer. The rest were divided into four groups: Information architects, Designers, Developers, and Hardware Architects equally distributed. The groups have quite different educational backgrounds. The information architects typically had a background in the arts and some training in the basic web-

technology (i.e., HTML). The designers' background was more diverse. They had a background in chemistry, graphical design, psychology, business and computer science. All the developers and hardware architects had a traditional technical background (programming, computer science or engineering).

The information architects' task was to determine the user groups for the portal and sub sites, categorize the intended and required information for each user group, and the functionality to navigate through the information. This did this in cooperation with some of the employees from the customer. The designers designed the graphical design. The developers and hardware architects implemented the technical part of the web-site, i.e., programming and configuration of the software and hardware of the system.

The project was divided into different phases. The first phase was inspired by the elaboration phase in the Rational Unified Process (cf., Jacobson et al., 1999) and included a series of meetings between a few people from WebSystems and representatives for the customer. The result was a number of general use case descriptions (Ibid.), a preliminary information architecture and tender material. The products the first phase was made by the product manager, an information architect and the project manager from WebSystems. In the next phase did the information architects establish a detailed information architecture, the designers made a 'look and feel' description of the site (defining pictures, fonts, etc.), and the developers prepared the technical development (they were basically trained in the tool the site was going to be implemented in). In the third phase the site was implemented and content was established and uploaded. Lastly the site was launched on time and meets the deadline.

## **4. OBSERVATIONS**

This chapter briefly illustrates and comments on some of the observations made during our study.

### **4.1 Development of web-information systems of today**

One of our basic assumptions was that web-applications are becoming increasingly business critical, and that this would be reflected in the organization of the development work. The Zyme project was an example of a web-application that was considered strategically important for the customer. It was a very large project and the customer required a thorough pre-analysis resulting in tender documents to ensure that different web-development companies could bet on the implementation. Furthermore, the deadline for version one was considered vital to the customer, and the site was considered the most important public relation activity. Thinking of IT as essential for public relation is new to most developers. The term 'branding' became important in the Zyme project. The designers and developers were informed that the sites should signal the attitude of the organization and high quality. These requirements must still be combined with the traditional requirements, such as informative, easy to use, quick to glance, etc. This was new to the actors, and obviously they had a very abstract and uncertain understanding of what it meant for their application. As one of the information architects phrased it:

"One of the ideas is that the site should contain something with 'a kick' - you know - some energy! It is important for them [the customers] that this brand is pushed in the head of the user. One of our solutions will thus be that beside the main navigation there must be room for the system to present interesting stuff from one of the sub-sites on the portal entry and thereby push information into the face of the users" (our translation).

The changes in the importance of the applications have implications for the requirements to documentation and systems stability, security, etc. In the Zyme project much effort was spent on designing a proper and useful hardware and software platform to ensure high stability and good opportunities for expanding the system.

Another basic assumption for our study was that we expected to find indications that web-based applications become more 'interactive' in the sense that they support interaction among different users of the application (cf., DIWA, 2000). The Zyme site is an example of interaction in the sense of electronic commerce. But apart from that we found less usage of web-applications as means for interaction among collaborating actors than we expected. Most web-usage is still organized mainly in terms of publishing information.

The experienced problems in WebSystems are not only related to the used methodology (or lack of), to the introduction of new competencies, etc. Also WebSystems have to cope with a fast changing technology. In the Zyme project it was decided to use a completely new platform to support the development of the web-site. The hardware was new and the product to which the server-software should be developed was new. Two of the developers in the Zyme project had a one-week course in the product. The remaining developers and hardware architects had to setup and program the server-product during the project without training. To support the process consultants from the supplier were allocated to the project. It was, however soon realized that the consultant had very limited knowledge of the product and could spend very little time at WebSystems due to limited resources at the product provider. It was the first Danish implementation of the product, and it was used for implementation of a strategic application!

When it was decided to use the new product it was already known that a new and better version would be released before the programming was about to start. It was not clear for the project participants, which features the new release would provide, but they knew the web-site should be ported to the new version shortly after the first version was released. The designers had more or less to design without knowing the technical possibilities. According to the involved actors this was one of the biggest problems in the project. The limited knowledge of the product caused frustration in all the involved groups and not only within the developers. The developers found that it would have been easier to implement the system in the products they usually used, and the information architects and designers encountered problems because they did not know the technical limitations of the new product. Consequently the information architects had to redesign part of the information architecture resulting in a new design of the user interface, which again involved the designers and a the creation of a new graphical design. This example illustrates the extraordinary pace at which new tools and techniques are invented creating an 'interoperability nightmare' for application developers and users and makes it difficult to manage the development process (cf., DIWA, 2000).

Another key aspect in the project was the customer's project manager. It was important for the customer to have a representative in the project, especially to ensure a good 'branding' and get the right 'look and feel'. To ensure this the customer's project manager was located together with the rest of the project members during the development. He collaborated with most of the project members in the different phases of the project. He was involved in making the information architecture, gave feedback to the designers, discussed functionality with the developers, and contributed to the preliminary and 'wild' ideas for the branding. Involvement in the 'brainstorm phase' made it possible for him to come up with new ideas and request changes in the original design described in the tender material and use cases.

Most of the project members from WebSystems stated that the influence of the customer's project manager was far too high and difficult to control. Often the project got a long series of comments to

preliminary ideas that had not yet been carefully considered by the information architects and designers themselves, and many of the request established this way were either not in the tender materiel (i.e., not in the contract) or impossible to implement in the tools selected. One developer phrased the problem as follows:

“... it is the first time that the customer is so close to a project. They are actually positioned right in the middle of the project and are able to take part in everything and — they do it!”  
(our translation)

And a designer stated the following about the customer’s project manager;

“The customer has been allover the project: They participated in the analysis and they intervened the development of the information architecture, and more or less also in the design. Thus he has been playing back and forth with the information architects and the designers. The developers have been focusing on their own problems and have not all the time been involved in the design. So they have not been aware of some changes for instance if we have moved some graphics or something like that. The customer has been involved in this and has in practice been the first to see all the things we have suggested. This is a consequence of the structure we have in the project with a customer as project manager. It has made it hard to manage the customer’s expectations...” (our translation)

The fact that the customer has had so much influence on the design process made it difficult to manage requirements. Each sub-group in the project had more or less their own version of the requirements dependent on the feedback, they had got from the customer. Opposing requirements were often discovered and had to be resolved, sometimes by telling the customer that a specific requirement could not be implemented or by changing the original requirements.

The relation between the developing company and the customer is changing. In web-information systems the ‘look and feel’ is even more important. Frequent feedback from the customer has always been essential in information systems design. The concept of a web-site is relatively easy to understand and comment on, especially if we compare it to understanding techniques like data modeling. The nature of the systems is easier to understand for the customers and the technology itself invites to tinkering and quick changes. Requirements management becomes even more complicated.

## **4.2 New competencies**

The main purpose of the Zyme project was to 'brand' the enzyme company (i.e. the customer) through the web-site. In the Zyme project terminology branding meant that the web-site should promote the enzyme company and signalize the company's values and high quality products. To achieve this and make all the traditional hardware and programming successful too require people with very different competencies in a project.

As described there were very different groups of actors (graphical designers, information architects, programmers hardware architects, domain experts, etc.) in the Zyme project. They had to collaborate closely to develop the application. Compared to traditional software development the target group is divergent and the project team have a less homogeneous background. This fact caused problems with the organization of the project and communication between the different groups involved. We observed several indications of this during our study. For example, the information architects could not understand the use cases and they invented their own syntax for specifying the information architecture, which was difficult for the designers and the developers to grasp. When asked how they informed the other groups about their work all the actors indicated that

this was done on oral basis only, and mainly at meetings. They had very little formalized notation or language that were used for sharing information and knowledge.

According to company managers, project managers, and different project members the general understanding in WebSystems is that web-development requires new and different competencies in contrast to the competencies represented in the Company in general. All the different types of actors working in WebSystems indicate this. The new people employed have primarily competencies in areas like visual design, communication and information handling.

It is, however not only a question of involving people with new areas of expertise. Also the 'old developers' have to be trained in new skills, tools, and techniques. All the developers and programmers we interviewed saw major differences in the kind of design and development work conducted in WebSystems compared to the rest of the Company where they develop more 'traditional information systems'. Developers moving from 'traditional systems development' have to learn new approaches. They must learn to work in development environments where the software to a large extent is hidden and spread in numerous small components, modules, scripts, tools, etc. The software does not have the same 'visible' overall structure and flow as they have been used to. As one of the developers said:

“It is damned difficult to get a complete picture of the software when it is distributed in hundreds of small bits and pieces, and no tool provides you with an overview or access to the complete source code” (our translation).

An interesting observation was that a shift in attitude from some of the young and recently educated developers was required. Both the project manager and some of the more experienced developers mentioned that young developers have no training in and understanding of what happens when software development grows and becomes large-scale. The habits from design of small (toy) systems become problematic. One of the software engineers talked about the problems of ensuring consistency and develop a stabile architecture:

"...we have to do things in a standardized manner, the same every time, and know what each other are doing, etc. etc. The web-world is characterized by young guys, you know, starting here just having finished their education, and wauuw, 'we just have to do this - damn fun yeahh!' Then it's hard because they ask 'why? Why should I do that?' They have never been in a situation where everything goes haywire." (our translation).

#### **4.3 New roles and forms of collaboration**

We have encountered some of the challenges the different competencies create. The four groups of people in the Zyme project coordinated their work and collaborated with each other to develop the web-site. The information architects designed the functionality and information architecture and communicated it to the developers. The developers communicated the technical possibilities and limitations back to the information architects. Beside these four main groups of competencies a typical project in WebSystems included two project managers (one internal and one from the customer) and people from the quality assurance department.

Apart from the groups mentioned the Zyme project also had actors from the server supplier and a marketing and advertising company involved. Furthermore there was a close interaction with domain experts from the customer. All these people were organized in small special groups and worked in the same physical place as the people from WebSystems.

Interaction between the groups (especially between the technical and non-technical people) was a critical part of the Zyme project. According to our observations and comparisons with other similar studies this is a general problem in complex web-development. Most of the actors involved in the Zyme project had no or very little experience in collaborating with the actors from the other groups. The interaction was problematic in several ways. A designer stated:

“..at the end [of the project] they [the developers] just stood and said ‘ohh.. that’s how it should be. I knew what it should look like but I did not know that this one should change’. In some way we needed some material we could use to hand over stuff with. I can’t explain how... we should have had some kind of document we could start our work from, where we could read all these facts about the design and information architecture, for instance structure, menus or something like that” (our translation).

It was problematic for the groups internally at WebSystems to communicate their results, findings and ideas. It was planned that the information architects should describe the overall functionality and navigation by means of use cases. They had, however no experience with using use cases and the result was that the use cases were specified by a project manager and a developer. The use cases were then discussed at meetings between the information architects and the developers. The agreement at these meetings was based upon the verbal presentation only since the information architects still were not able to understand the use case specifications. Despite this problem it was the information architects that specified the information architecture and established a matrix specifying which user groups should have access to which information structures. Both the specifications were made in self-designed structures unfamiliar to the developers.

The division of labor and the responsibilities for each group were difficult to define for the actors involved in the project. The most common comment to this was that 'this is the first time I'm involved in this and I think we must be better in handling this in our next projects.' Some of the people in the project considered it a more permanent problem, not necessarily related to the fact that things were new and more or less unknown to everybody. Normally, it would be expected to be the responsibility of the project manager to ensure clear interfaces between the different groups and activities. However, the Zyme project obviously had so many uncertainties that such a clear division of labor could not be established up front in the project.

The different groups had different ways of working. When the information architects needed an overview of the web-site they invented their own diagrams specifically for this. They also made simple paper-based mock-ups of the user interface. The designers used these mock-ups to get a rough idea of the user interface and designed user interface prototypes in visualization tools (e.g., PhotoShop). The developers used the prototypes, the information architecture diagrams, and the original use cases from the tender material as basis for the implementation of the web-site. There was often lack of consistency because the groups worked with different requirements and understandings of the system. As mentioned the main reason for this was problems with handing over information. A developer stated:

“It is difficult to figure out how the hand over of information should happen. They had a fine hand over from IA [information architects] to design but it [the information] stopped there and has not proceeded to development. It is probably not enough that the developers talk with the designers, we probably also need to get in contact with IA to get an idea about the functionality besides what they should look like, what are IA’s thoughts, etc. So we need some common hand over. That would have been smart.” (our translation).

The members of the project lacked an understanding of each other’s work and of the information the other groups needed to do their work. A common understanding is needed but can be difficult to

achieve because of the different background, traditions, etc. Many of the project participants indicated, that a solution might be new development methodologies that addresses the different tasks in the development of web-based information systems. This still needs further investigation. None of the attempts with new methodologies and techniques that were taken in WebSystems appeared successful. All actors agreed that a better understanding between the expertise groups of the other groups' needs is required. Some of the project members stated this could be a matter of more experience, but this might not be sufficient.

## **5. DISCUSSION - CHALLENGES FOR WEB-BASED INFORMATION SYSTEMS DEVELOPMENT**

One of the central characteristics we have observed in web-development is a shift in the basic characteristics of web-applications. Web-applications have been used for publishing information, with limited functionality, and have been peripheral to the core business. From our study and others we have compared to we can observe that *web-applications are becoming critical to the companies' business*. Thus, the demand for a higher quality of web-applications is increasing significantly. This is not reflected in the approaches taken, the methodologies applied, and the organization of the work used in much development of web-applications of today. Web-applications are to some extent still developed as if they were 'toy-systems' primarily used for publishing information. Web-applications will be used for more strategically purposes in the future and therefore be important for business.

The point made here can also be observed from the fact that *much web-application development is driven by technology push* in high speed innovative settings. In parallel with the shifts in importance of the web-applications developed, we expect to see a shift towards more technology pull approaches where needs, requirements, and high quality becomes essential. Hence, *a major challenge for development of web-application is a shift in attitude* and maturity of the work. The approaches taken and the practice applied must be changed. Careful planning of the process and investigations of needs and possibilities must be undertaken. The roles to be involved and their responsibilities and interaction must be outlined much better than today, and the importance of careful considerations of design and architectural choices needs to be understood by the involved actors, i.e., taught in classes and communicated by the key people in the actual development efforts.

We have observed and followed a number of attempts to introduce new methodologies and techniques for coping with some of the problems related to the challenges in web-applications development. The success of these initiatives were very limited. Some of the encountered problems were caused by introducing a methodology, which was perceived as designed primarily for product development. This did not conform to web-application development. *A better understanding of the specific requirements for methodologies* supporting development of web-based applications and information systems is required. The general knowledge of what characterizes web-development is too limited. From our observations we would argue that the methodologies should not be too complex or require a lot of skills in formal modeling and specification. However, the systems are becoming complex, and it is difficult to think of solutions not involving some structured formalization. Iterations and prototyping might not be useful either in order to cope with some of central characteristics of today's web-applications, e.g., systems with known, but highly complex information structure. A more elaborate discussion on relevant approaches along the dimensions of complexity vs. uncertainty and modes of operation vs. means of expression (cf., Mathiassen and Stage, 1992) is required.

In parallel to the challenges concerning methodology and approach is the problem of *how to organize web-application development work*, and how this should be reflected in the methodologies and approaches we choose. The aim is still to handle the process efficiently and effectively. We also have to understand which roles and competencies that are needed in development of web-applications. There is obviously a demand for many different competencies, but which are essential? At WebSystems many different types of expertise were involved. Most of the project and department managers we interviewed had severe doubts on which skills are essential to have in the projects.

An aspect of the organization of the development work is also the role of, and interaction with, representatives of the users or the customer. Since the web has been used mainly for publishing it has been fairly easy to have strong user-involvement in all phases of the projects. However, the nature of the products are changing, and that has to be reflected in order to *identify the most appropriate models for user-involvement*. The customers and users will probably play a much more active role by furnishing the design process with information to both the information content, the actual design, the architecture, etc. due to the fact that the customers have more knowledge and insights about how applications can be build by means of web-based technology. The relation between developers and customers changes (see also DIWA, 2000). This raises customers' expectations and increases the demands on the developers when negotiating with the customers and users. The trend will furthermore increase the needs for methods and tools supporting the communication between the developers and the users and customers.

As mentioned several times the need for different types of expertise will increase. This causes one of the central challenges to web-application and web-information systems development: *How can employees with heterogeneous backgrounds be qualified* to handle technical, management and communication problems? How do we provide these very different groups of actors with a '*common language*' that enables them to not only communicate, but actually to exchange central information about the needs, the design and the implementation of a complex web-based information system? In our studies we observed that the designers and developers had severe problems in understanding each other. They were not able to read each others diagrams and specifications, and at some occasions the responsibility for certain tasks were changed simply because some groups of actors did not master the tool or method intended. Until these problems have been coped with much web-application development will be very inefficient. Of course, better education and teaching might be part of the answer, but we believe *that new concepts and tools are required to really make progress* on this. This requires a deeper and more coherent understanding of the problem and a detailed analysis of requirements for such concepts and tools.

Although the different groups have very different background and traditions for organizing their work they have to collaborate on developing the systems, i.e., they are highly interdependent in their work and they have to coordinate their activities. This has further implications for the approaches and methodologies we provide. Methodologies and techniques have to find a balance, which can conform to both technical and non-technical people and can be understood and used by the different groups. The methodologies must *ensure means for coordinating activities among actors that might not really understand the core problems and challenges in each others activities*. Even more clear cut interfaces between the different activities than what is common today and more explicit support for establishing the required division of labor should be provided. This might be somewhat conflicting with the common approach in much web-development today having a high degree of trial-and-error iterations, very loose specifications, and little or no documentation. Again the right balance has to be found.

Development of web-applications will often be confronted with requirements combining 'traditional requirements for information systems' (e.g., easy to use, task relevant facilities, response time, etc.) and *requirements related to advertising and branding*. The latter is new to many developers within web-development. Design and development must be provided with tools that support a conceptual understanding of the two sets of requirements (that are often in opposition to each other) and how they are interrelated. Such tools should also support actual design that fulfills the requirements. Most of today's traditional design tools are designed to optimize effectiveness, efficiency, usability, etc. They are not designed for development of applications that are intended to be used for branding and have the users to keep browsing at the web-site as long as possible.

Apart from the changes in the amount of user involvement in the projects our study has also indicated that the division of labor between developers and users shifts in other areas too. Other studies have also indicated that in most web-based information systems the information content and structure is developed and maintained by users, typically local super users or web-administrators (cf., e.g., Bansler et al., 1999). These users are often distributed in time and space. There are therefore no centralized overall instrument for maintaining the overall structure and coherency of the site. The approaches and methodologies for developing and maintaining large web-applications have to take this into account. *Support is needed for building a stable web-site infrastructure that is firm enough to cope with distributed and non-coordinated maintenance, and flexible enough to provide proper support for the individual needs*. Furthermore, the tendency to change the division of labor means that future designers of information systems have to accept that they are not 'in charge' of the information structure and content.

## 6. CONCLUSION

The development of information systems of all kinds has been heavily influenced by the introduction of the web-technology as a platform for many different types of systems. This implies major changes in the approaches, organization principles, methodologies, tools, etc. we use for systems (web-application) development. In this paper we have aimed at informing this work. We have presented findings from a field study conducted at a fairly large web-development department. Our findings have illustrated that: 1) the web-technology is used for many different kinds of applications, including business critical usage; 2) there are severe problems in just adapting 'traditional' software development approaches in web-application development; 3) many new competencies must be involved, and this involvement complicates collaboration and interaction between the involved actors; and 4) the tools and platforms are shifting extremely fast in the web-development world, causing problems both to development and management.

From this we have discussed a preliminary list of overall challenges for the web-development of today. These challenges include:

- New approaches methodologies supporting the dedicated needs;
- Changes in the organization of the development work, ensuring a better interaction among the different groups of expertise and interaction with customer representatives;
- Experimentation with new roles and expertise to be involved in development of web-based information systems;
- Changes in attitude to rigidity of the processes conducted during development;
- Improved concepts and tools for interaction, collaboration and coordination among very heterogeneous groups of designers and developers; And

- Improved concepts and tools for interaction between developers and users and customers.

The essential challenges for development of web-based information systems of today presented here are derived from a single field study only. Further investigations, field studies, experimentation with approaches, tools, etc. are, of course, required. We do, however believe that our observations are quite general in the sense that they fit with other observation we and others have conducted, cf., the discussion in the introduction and similar studies from the DIWA project (DIWA, 2000). The conclusion is then: A lot of rethinking, conceptualization and development is required in order to provide proper support for web-based information systems development in the future.

## Acknowledgments

This research could not have been conducted without the invaluable help from numerous people at WebSystems and the Company. The data collection and analysis work has been undertaken in close collaboration with Lars Kofod. The work has been funded by The Danish Research Councils.

## REFERENCES

- Balasubramanian, V., and Alf Bashian (1998). Document Management and Web Technologies: Alice Marries the Mad Hatter. *Communications of the ACM*, **41** (7), 107-115.
- Bansler, Jørgen P., Erling Havn, Jacob Thommesen, Jan Damsgaard, and Rens Sheepers (1999). Corporate Intranet Implementation: Managing Emergent Technologies and Organisational Practices. in *J. Pries-Heje at al. (eds.): 7th European Conference on Information Systems, Copenhagen*, Copenhagen Business School, , vol. 3, pp. 750-757.
- Baron, John P., Michael J. Shaw, and Andrew D. Bailey (2000). Web-based E-catalog Systems in B2B Procurement. *Communications of the ACM*, **43**(5), 93-100.
- Braa, Kristin, Carsten Sørensen, and Bo Dahlbom (2000). Changes - From big calculator to global network. in K. Braa, et al. (eds.): *Planet Internet*, Studentlitteratur, 13-39.
- Burdman, Jessica (1999). *Collaborative Web Development. Strategies and Best Practices for Web Teams*, Addison-Wesley, Reading etc.
- Carstensen, Peter H., Carsten Sørensen, and Tuomo Tuikka (1995): Let's talk about bugs!. *Scandinavian Journal of Information Systems*, **7** (1), 33-53.
- DIWA (2000). <http://www.diwa.dk>.
- Grinter, Rebecca E.: (1997). Doing software development: Occasions for automation and formalisation. in J. A. Hughes at al. (eds.): *ECSCW '97. Proceedings of the Fifth European Conference on Computer-Supported Cooperative Work, 7-11 September 1997, Lancaster, U.K.*, Kluwer Academic Publishers, Dordrecht, pp. 173-188.
- Isakowitz, Tomas, Michael Bieber, and Fabio Vitali (1998). Web Information Systems. *Communications of the ACM*, **41** (7), 78-80.
- Jacobson, Ivar, Grady Booch, and James Rumbaugh (1999): *The Unified Software Development Process*, Addison-Wesley, Reading etc.
- Kraut, Robert E., and Lynn A. Streeter (1995). Coordination in Software Development. *Communications of the ACM*, **38** (3), 69-81.

- Mathiassen, Lars, and Jan Stage (1992). The principle of limited reduction in software design. *Information Technology and People*, **6 (2-3)**, 171-185.
- Mowshowitz, Abbe (1997). Virtual Organizations. *Communications of the ACM*, **40 (9)**, 30-37.
- Orlikowski, Wanda J. (1993). CASE Tools as Organizational Change: Investigating Incremental and Radical Changes in Systems Development. *MIS Quarterly*, no. September 1993, 309-340.
- Patton, M. Q. (1980). *Qualitative Evaluation Methods*, Sage Publications, Beverly Hills, CA.
- Schmidt, Kjeld, and Peter Carstensen (1990). *Arbejdsanalyse. Teori og praksis [Work Analysis. Theory and Practice]*, Risø National Laboratory.
- Turoff, Murray, and Star Roxanne Hiltz (1998). Superconnectivity. *Communications of the ACM*. **41 (7)**, 116.
- Yin, Robert K. (1989). *Case Study Research: Design and Methods*, Sage Publications, Beverly Hills.

## **Paper B:**

### **User involvement in web development**

Vogelsang, L. "User involvement in Development of Web Based Publishing," *Proceedings of the 11th European Conference on Information Systems*, CD-Rom, Naples Italy, 2003.

# User involvement in Development of Web Based Publishing

Lasse Vogelsang, The IT University of Copenhagen, Denmark, E-mail: Vogelsang@itu.dk

## Abstract

*The theme of this paper is development of Web applications used by an organization to publish information to groups of users outside the organization. The paper focus on the challenge for the Web development organizations to help organizations finding and characterizing target groups for web sites and to create Web applications that can be used to communicate certain information and values to these specific target groups. The paper reports from a field study of a project developing a Web application to be used for information publishing to a number of target groups. The field study showed that such a project is different from other types of software development with respect to the point in time the different groups of users are identified and to what extend these groups are involved in the development process. The conclusion is that development of Web based information publishing is characterized by the importance of branding instead of selling products, a characterization of the user of the information system instead of involvement in the development process, that there is no work to be supported by the Web application, and finally that the task of "branding" includes more than usability issues.*

## Keywords

Web development, software development paradigms, user involvement

## 1. Introduction

The use of information systems has changed within the last decades. In the 1980s the Personal Computer was introduced and the computer was primarily perceived as a tool used for work related tasks. Today with the wide spread use of the World Wide Web, information systems are also used for information publishing, eCommerce, and other services (Braa, K., C. Sørensen & B. Dahlbom 2000). According to Turoff (1998) a Web based information system is a new medium for human communication. In order to develop this type of systems, expertise in information architecture and graphical design are required in the development projects (Burdman 1999; Carstensen & Vogelsang 2001; Murugesan & Deshpande 1999).

With the introduction of the Personal Computer ease of use became important so the average user could use the systems as intended (Grudin 1991). Today the World Wide Web has become widely used for communication purposes. In recent years Web applications have become business critical to many organizations. Web applications are used for e-commerce, information publishing, communication, and for other purposes that includes involvement of users that are outside the organization that owns and run Web applications. Today it is critical for organizations to have their web sites up and running all the time so people do not think that the organization is out of business or just not competent enough to maintain a web site. This was not the case just a few years ago when web sites were taken less serious and web sites that were down for a period of time were considered part of being on the Internet.

Another typical problem for organization's web sites is when it is difficult for the users to search for information and navigate on the web site. Creating a web site with the appropriate information and

making this information easy to navigate and search are not only a technical challenge. An understanding of the intended users of the web site is needed. Understanding the users of a web site to such an extent that the content on the web site can be structured and navigation created is a big challenge for the development organization. Today web development organizations must be able not only to ensure the Web applications for technical problems, but also have to provide a usable information architecture making users feel comfortable and create a "look and feel" that communicates the values of the organization that buys the Web application. An information architecture is the organization of the information on the web site and the functionality to navigate on the web site that enables users to get find information and answer their questions (Rosenfeld & Morville 1998).

This paper concentrates on the development of Web applications used for information publishing from an organization to potential external groups of users. This specific type of development is important because a significant number of Web applications are used for publishing information to users outside the organization that owns the system. In order to characterize this specific type of development a field study has been undertaken. In the field study the development of a major web site in an organization that develops Web applications was followed.

The paper describes some of the characteristics in the development of Web applications used for information publishing. The next section describes the approach taken in the field study followed by section that describes the studied project and its setting. After that section the involvement of users in the studied project is analyzed and compared to the software development paradigms described by Grudin (1991). Then some characteristics of this specific type of web development are presented. Finally it is concluded that in order to understand this specific type of software development and characterize the user involvement we should think about users as external and internal users. It is furthermore concluded that the involvement of external users seems to be quite limited in this type of projects because it is sufficient to characterize the user without involving them.

## **2. Approach**

The research described in this paper is based on a field study conducted in a software development company that develops Web applications. The objective was to obtain an in-depth understanding of the development of Web applications used for information publishing. More precisely the implication caused by the shift from developing information systems for internal business processes to information systems used for communication and information publishing to people outside the organization was studied. The study had an emphasis on the context of the project, activities in the project, involved expertise, and the use of software methodologies, techniques, and tools.

The field study was based on qualitative data collection techniques. The main methods for data collection were semi-structured qualitative interviews, document analysis, and observation of project meetings (Lofland & Lofland 1995; Yin 1989).

Four interviews were conducted at the outset of the project in order to understand the initial ideas with the project and the company the project was conducted in. At the outset of the project a project manager, an information architect, a developer, and a hardware architect were interviewed. After the web site was launched and the project was finished a new set of interviews with a graphical designer, the same information architect, developer, and project manager were interviewed in order to follow up on their experiences of the project. In total eight semi-structured interviews was conducted. Each interview lasted from one to two hours and was all tape-recorded and transcribed.

The study included observation of approximately twenty project meetings. The observations were mainly used to validate the researchers' interpretation of the data from the interviews. Furthermore documents used in the projects were analyzed for instance the tender material, diagrams used for creating the information architecture, Use Cases, descriptions of the software method used in the project, minutes from meetings, and material about the customer (brochures, web sites, news papers, etc.). After these activities a working paper was produced describing findings from the study. This working paper has been discussed and validated by actors in the project and discussed with a large group of researchers who have made similar studies of web development (see DIWA, 2002 for further details).

### **3. The Case**

The study took place in a software development company (hereafter called the Company) that develops software for one of the larger Danish pharmaceutical companies. The Company is part of the same holding company as the pharmaceutical company. The Company develops business systems, production systems, web based systems, and they support and maintain the IT-systems they build. The study took place in the Company's web development department (hereafter called WebSystems). WebSystems had existed for 5 years and there were approximately 70 persons in the department when the study took place.

#### **3.1 The Zyme Project**

The project studied is called the Zyme project (a pseudonym). The purpose of the Zyme project was to develop a web site for a company called ZymeCorp (a pseudonym), which is a company in the same holding company as the Company. The web site that was developed consisted of a main web site with a number of sub-sites and a web based content management system for the web site. The sub-sites address different groups of users or target groups, such as investors, the press, students, and customers. WebSystems was responsible for preparing a call for tender during a two months period. Based on the tender material WebSystems made Use Cases, an information architecture, and a specification for the technical equipment. Three companies made a bid on the project and WebSystems won the project.

WebSystems had responsibility for the design and the development of all aspects of the web site, i.e. the information architecture, the user interface, the technical infrastructure, programming, testing, etc. The whole project including the call and creation of the tender material lasted ten months. The web site was launched on time and the Zyme project was considered a success by WebSystems, although some features were not implemented due to delay in the design process and the project as such. Today, two years after the web site was launched, ZymeCorp has added the possibility to sell its products to the web site and is now selling one third of its products through the web site. The "branding" on the web site has not been changed significantly since the web site was launched and is another indication of a successful project.

#### **3.2 Groups, tools, and techniques in the Zyme Project**

The project lasted ten months and 8-18 people were allocated to the project. Most of them worked full-time. The project had two project managers, one from WebSystems and one representing ZymeCorp. The rest were organized in the following four groups: Information Architects, Graphical Designers, Developers, and Hardware Architects (see table 1). Each group consisted of 3-5 persons and was formed after WebSystems was awarded the project. The people in the groups had quite

different educational backgrounds. The Information Architects typically had a background in the humanities and some training in the basic web technology (i.e. HTML). The Graphical Designers had backgrounds like psychology and graphical design. The developers and hardware architects all had a technical background in computer science or engineering.

The information architects created the structure of the web site. They did this by dividing the users of the web site into target groups, finding and categorizing the content for each of the target groups, and creating sketches for the navigation through the content. The structure of the web site was created in cooperation with some of the employees from ZymeCorp and ZymeCorp's project manager in the project. The Developers and hardware architects implemented the technical part of the web site, i.e., programmed and configured the software and hardware for the system. The Graphical Designers made guidelines for the web site, i.e. colors, fonts, form of images etc. and the exact placement of the elements on the web pages. The Graphical Designers made the guidelines in cooperation with the ZymeCorp project manager. There was no guideline the Graphical Designers could base their work upon because ZymeCorp became an independent company the day the web site was launched. Therefore did ZymeCorp need a new "look and feel" for the web-site to present ZymeCorp. The new "look and feel" had to be made almost from scratch.

<b>Group</b>	<b>Main activity</b>	<b>Tools used</b>	<b>Educational background</b>
Information Architects	Gathering and structuring of information	Developed their own diagramming technique	The Humanities
Graphical Designers	Creation of the graphical design	Screen dumps, Graphical Program	Graphical Design, Psychology
Developers	Technical implementation	Application platform	Computer Science, Engineering
Hardware Architects	Decide server type and architecture		Engineering

*Table 1. Groups involved in the Zyme project.*

Due to the different expertise involved in the project a number of different tools and techniques were used. The Information architects created their own type of diagrams to describe the structure of the web site. The diagramming technique was simple but sufficient for the information architects to express the relation between the web pages on the web site. The information architects used their diagrams and created some sketches of the interface, which included the rough location of elements on the screen and short descriptions of the functionality in order to communicate the structure of the web site to the graphical designers, so the graphical designers would be able to create the graphical appearance of the web site. The graphical designers used this information to create the visual appearance, which was created in a graphical program and the result was a number of bitmap-files showing location of pictures, colors, and location of elements like buttons, navigation panels, etc. The developers used the information architects' diagrams and sketches, the graphical designers' bitmap-files, and the Use Cases from the tender material as information sources to build the system.

## 4. Observations from the Zyme Project

In this section the Zyme project is compared to the three paradigms of software development described by Grudin (1991). The paradigms provide a vocabulary that is useful to characterize an important part of this specific type of web development: The involvement of users. But first a brief description of Grudin's three paradigms.

### 4.1 Software development paradigms

Grudin (1991) identifies three typical paradigms in software development; contract-, product-, and in-house development. He distinguishes each paradigm by the time at which developers and users are identified in the software development project. The developers and users are either identified at the outset of the project or later in the project (see table 2). In Grudin's terminology the users are the people who are directly engaged with the system and the developers are the active members of the development project [ibid]. The three paradigms are ideal types of software development projects and are not intended to clarify the fundamental assumptions as it is in other descriptions of software development paradigms (e.g. Hirschheim & Klein 1989, Dahlbom & Mathiassen 1993)

<b>Paradigm</b>	<b>User identified</b>	<b>Developers identified</b>
Contract development	At the outset of the project	After a contract is awarded
Product development	When product is bought by customers	At the outset of the project
In-house development	At the outset of the project	At the outset of the project

*Table 2. Development Paradigms. Adapted from (Grudin 1991).*

Grudin (1991) defines contract development as the type of development where the user organization is identified from the outset of the project and the development organization is identified after a contract is awarded. Grudin states that the main focus in contract development is software methodologies (especially the waterfall model) in order to control that the developed product meets the contract. In product development the developers are identified from the outset while the actual users are not identified until the product is marketed, although there is some notion of the users throughout the development of the product usually in terms of a potential market to sell the product. The focus in product development is ease of use and to some extent "look and feel" to make sure the product can be sold. In in-house development the developers and the users are both identified from the outset [ibid] and the focus is on user participation and user acceptance.

A software development project can have aspects of more than the one paradigm and does not necessarily conform to one specific paradigm. Despite this, it is still useful to characterize the user involvement in web development in order to characterize aspects of the user involvement in development of web applications for information publishing. This paper uses the time of identification of users to characterize the development and acknowledge that there are other aspects that influences the Zyme project.

### 4.2. User groups identified at different times

The main task for WebSystems in the Zyme project was to develop a major web site intended to be used by students, the press, researchers, customers etc. and to create a content management system

for the web site to be used by ZymeCorp. WebSystems helped ZymeCorp to find and structure the information to be published on the web site by ZymeCorp. This was necessary in order to create the structure of the web site and the functionality (primarily navigation and search functions).

WebSystems had to understand two different groups of users; the internal users of the web site in ZymeCorp and the external users of the web site i.e. the target groups of the web site. WebSystems had to have a notion of the external users of the web site to create a useful information architecture that would enable the external users of the web site to find the information they needed and could be interested in. WebSystems also had to understand the internal user, i.e. the users of the web site and the content management system in ZymeCorp. This was necessary in order to create an information architecture that would enable the internal users to publish additional information on the web site after the launch of the web site.

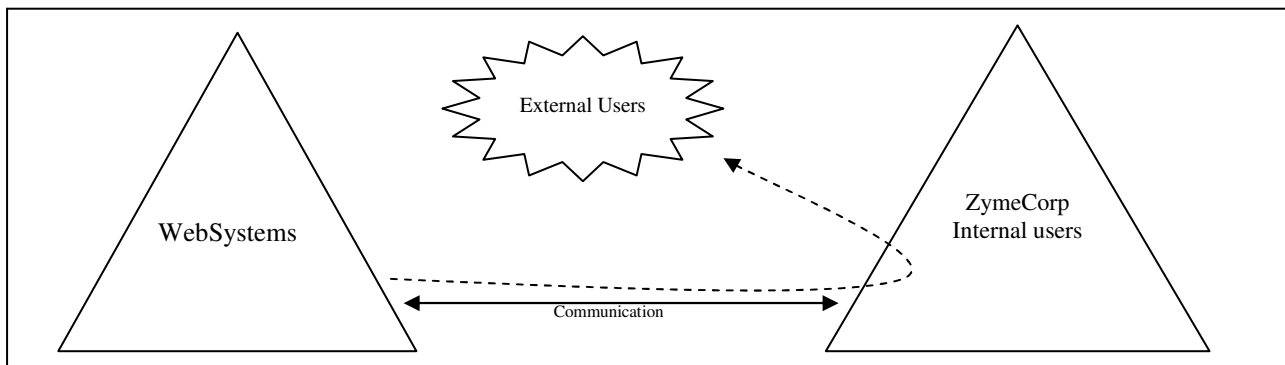


Figure 1. The groups involved in the Zyme project

Besides being the information publisher and the information consumer the two groups of users were distinct in respect to the time at which they were identified in the project. WebSystems could identify the internal users in ZymeCorp at the outset of the project because ZymeCorp was the customer and were able to point WebSystems to the future users in ZymeCorp. The external users were more difficult to identify. WebSystems had only a weak notion of the external users because ZymeCorp only had a notion about what the target group the web site should have and ZymeCorp had not decided all the target groups the web site should be aimed at and what information to be published to the target groups. It was WebSystems task to cooperate and help the project manager from ZymeCorp to find the target groups and the information to publish to the target groups. This cooperation was a challenge for WebSystems because they had to make detailed decisions about the information that should be on the web site almost throughout the whole project. On some occasions the decisions demanded new functionality that made it hard to settle the requirements for the web site, which the Developers occasionally could base their work upon. The problem for WebSystems was that a part of the project was to help the project manager from ZymeCorp to make the information architecture. This took months before the information architecture was developed sufficiently to enable the ZymeCorp project manager to accept the information architecture. One of the problems was that the information architects only had a limited understanding of the information that was going to be published on the web site. In one of the interviews an information architects said that she only had brochures from ZymeCorp and some materials about enzymes (the product produced by ZymeCorp) as information source for the information architecture and found it difficult to understand what the products were and consequently what information that should be on the web site.

## **5. Characterizing the paradigm for the Zyme project**

In each of Grudin's paradigms the users are perceived as a more or less homogeneous group in respect to the time at which they are involved in the development project (while the group of users can be heterogeneous in respect to issues such as computer literacy, tasks to support, etc.). But the Zyme project did not have a single homogenous group of users identified at specific time in the project, but two distinct user groups identified at the outset of the project (internal users) and later in the project (external users) respectively.

If the focus is on the internal group of users in ZymeCorp the development paradigm of the Zyme project is contract development or in-house development because the ZymeCorp users are identified from the outset of the project. Focusing instead on the external users the development paradigm is product development because this group of users is not identified at the outset of the project. Thus, the paradigm of the Zyme project is depending on the group of users in focus and therefore it is not a pure instance of any of the three software development paradigms.

Grudin uses the identification of developers to distinguish between contract development on the one hand and product and in-house development on the other. It is not clear whether the Zyme project was in-house or contract development because WebSystems made the tender material themselves and afterwards won the contract (contract development) and at the same time is a company in the same group of companies as ZymeCorp (in-house development). This aspect will not be further elaborated in this paper because the process and awarding of a contract in this case has little relevance for the involvement of users in the development process.

## **6. Characteristics of development of Web Based Information Publishing**

This section describes some characteristics of development of web based information publishing based on the Zyme project.

### **6.1. Branding instead of selling**

It is assumed by Grudin (1991) in his characterization of product development that there are a number of user organizations to which the developed product or software package can be sold after the product is released. In many cases user organizations are identified before the development of the product is initiated to ensure that the product can be sold. This makes it possible for the development organization to find potential users although it is sometime difficult to reach them due to competition and lack of time (Grudin 1991). But it is possible for the development organizations to make interviews, observations, workshops, and so forth to gain knowledge about the work the product is supposed to support.

In the Zyme project there was no product or software package to be sold. Instead the purpose was to publish information to the target groups. The goal for ZymeCorp and thereby WebSystems was not to make money on selling the web site as such, but to "brand" ZymeCorp. Obviously ZymeCorp had a notion about how the web site could improve their business for instance by selling products through the web site, but it was not the intention to create income from the web site as such.

In an action research project, Vidgen (2002) found that it is essential to have an external orientation in an eCommerce project, because the aim is to sell products and services to customers. In the Zyme Project there was also an external orientation although there were no products or services to be sold.

The orientation in the Zyme Project was primarily towards the user organization and its representatives (the ZymeCorp project manager).

## **6.2. Characterizing instead of involving external users**

In the eyes of the information architects their main task was to see the web site from the users' perspective (in contrast to the developers' technical perspective) and try to figure out what the different target groups could be interested in reading and searching for on the web site. In order to do this, the information architects needed some notion of the external users. Therefore the information architects described a number of relevant characteristics of potential external users such as employment, age, gender, education, and so forth.

The information architects did not involve any potential external users in the project. One explanation for the lack of user involvement is that the web site should not support specific work and consequently the notion of the tasks to support through the web site was vague and unspecific. Some very general tasks like search and e-mail notification were described in Use Cases, but were not targeted towards specific target groups on the web site. The information architects had to determine which Use Cases (i.e. functionality) that should be included on each sub site. In order to do this WebSystems found it necessary to characterize the external users to such an extent that they could help ZymeCorp to determine which information to publish, in what form (for instance the visual appearance, fonts, etc.), and with what functionality. Another reason for not involving external users was that WebSystems had to involve internal users from ZymeCorp in the process and did not have the sufficient resources for involving external users.

As stated earlier the Zyme Project has similarities to product development if the cooperation between ZymeCorp and Web Systems is seen as one company developing a product. ZymeCorp and Web Systems had the same possibilities for involving users in respect to involvement of users although some potential customers sometimes are identified in product development, which makes user involvement more feasible and practical.

The work made by the information architect and the ZymeCorp project manager is quite similar to the work advertising companies do in their work with target groups. Advertising companies do not necessarily involve people from the target groups. It seems that it is not necessary, in the development of web applications for information publishing, to have users involved in the project to the same extent as it is in contract and in-house development, where the users often are more involved in the projects because specific work has to be supported. This is in contradiction to the requirements put forward to methodologies for developing web application (e.g. Howcroft, D. & J. Carrol 2000; Standing 2001). The external users can be involved indirectly through their use of the web site that can be analyzed through web logs that can be a foundation for further improvements of the site. The web application build in the Zyme project was the first version, so no data like web logs existed for analysis.

## **6.3. Branding is more than usability**

In product development the software is developed in order to be sold and used for specific work related activities that usually can be studied and the product to be sold has competition from other similar products. Therefore product development is concerned with "look and feel" and usability to make the product "look and feel" good and make it easy to use. Grudin states that "...attention to "look and feel" reflects attention to usability" (Grudin 1991, pp. 65). In the Zyme project attention to "look and feel" was much broader than creating a usable web site. Obviously it was important to create a web site that would be easy to use. But the purpose of the Zyme project was not just to

publish information that would be easy to find and presented in a "good looking" way. The purpose was also to "brand" ZymeCorp. That is to communicate values and an impression of ZymeCorp to external users. Thus the "look and feel" did not only mean "looking good" but should also communicate a specific impression or image of ZymeCorp. In the ZymeCorp case the general values that should be communicated were serious, ethical, and trustable and varied a bit for each of the target groups. These values should be expressed through the colors, images, fonts, and the text on the web site and were found by the graphical designers in the Zyme project.

The information architects and the graphical designers spend a significant time (months) on finding information for the web site and creating the "look and feel" in order to communicate values. These type of activity are almost absent in other types of software development. It indicates that development of web application for information publishing to external users includes activities and tasks that are not present in other types of software development. Despite this, a range of classical software developments tasks like creating a software architecture, programming, testing, and configuration management still exist in development of web applications used for web publishing.

## **7. Conclusion**

An increasing number of organizations use web sites to publish information to specific target groups outside the organization. It is important for such organizations that their web sites are designed to give external users a certain impression of the organizations. The field study showed that the development organization is heavily involved in the task of designing the web application so it will communicate the values intended by the organization that owns the web application. The paper puts forward the following three conclusions:

There seems to be two distinct groups of users in development of web applications for information publishing, the internal and external users. Applying Grudin's (1991) paradigms of software development to the Zyme project, the development can be described as product development when the focus is only on the external users and in-house or contract development when the focus is on the internal users. Consequently the Zyme project is not a pure instance of one of the three software paradigms described by Grudin (1991), but has aspects of all three paradigms. This is a general conclusion for developments of web based information publishing targeted to groups outside a user organization, because there will be different types of relations to both internal and external groups of users.

It seems that user involvement is as important in development of web applications for information publishing as it is in other types of development. In the development of the ZymeCorp web site no external users were involved. It seemed that a characterization of the target groups was sufficient to decide what information to include on the web site and how this information should be communicated. The task for WebSystems to understand the external users has similarities to the work in advertising agencies.

In order to "brand" the user organization the development organization has to understand the values the organization would like to communicate, the information to communicate, and the external user that the organization wants to communicate to. This demands that the development organization has some domain specific knowledge about the information to structure. This can be a problem for the development organization because the information to be published on the web site can be very specialized and demand years of training in order to understand it.

## Acknowledgements

This research could not have been conducted without the invaluable help from numerous people at WebSystems and the Company. The research is part of the DIWA program and has been funded by The Danish Research Councils. The field study was conducted with Peter Carstensen and Lars Kofoed.

## References

- Braa, K., C. Sørensen, & B. Dahlbom (2000). *Changes: From big calculator to global network*. Chapter 1 in Planet Internet. K. Braa, C. Sørensen, B. Dahlbom (eds.) Studentlitteratur, Lund, Sweden.
- Burdman, J. (1999). *Collaborative Web Development. Strategies and Best Practices for Web Teams*, Addison-Wesley, Reading etc.
- Carstensen, P. H. & L. Vogelsang (2001). Design of web-based information systems - new challenges for systems development? in *Proceedings of the 9th European Conference on Information Systems*, p. 536-547, Bled, Slovenia.
- Dahlbom, B. & L. Mathiassen (1993). *Computers in Context: The Philosophy and Practice of Systems Design*, Blackwell Publishers.
- DIWA (2002). Design and Use of Interactive Web Applications (DIWA) Research Proposal, <http://www.diwa.dk>.
- Grudin, J. (1991). Interactive Systems: Bridging the Gaps Between Developers and Users in *IEEE Computer*, Vol. 24 (4), 59-69.
- Howcroft, D. & J. Carrol (2000). 'A Proposed Methodology for Web Development' in *Proceedings of the 8th European Conference on Information Systems*, Vienna, Austria.
- Hirschheim, R. & H. K. Klein (1989). 'Four Paradigms of Information Systems Development'. In *Communications of the ACM*, 32 (10), 1199-1216.
- Lofland, J. & L. H. Lofland (1995) *Analyzing Social Settings: A Guide to Qualitative Observations and Analysis* (3rd ed.). Belmont, CA, Wadsworth Publishing.
- Murugesan, S., & Y. Deshpande (1999). 'Web Engineering: A New Discipline for Development of Web-based Systems' *First ICSE Workshop on Web Engineering (WebE-99)*, Los Angeles, USA.
- Rosenfeld, L. & P. Morville (1998). *Information Architecture for the World Wide Web*, O'Reilly, Sebastopol.
- Standing C. (2001). 'The Requirements of Methodologies for developing web applications' in *Proceedings of the 9th European Conference on Information Systems*, p. 536-547, Bled, Slovenia.
- Turoff, Murray, & S. R. Hiltz (1998). 'Superconnectivity' in *Communications of the ACM*, 41 (7), 116.
- Vidgen, R., (2002). Wisdm: constructing a web information system development methodology. *Information Systems Journal*. Volume 12, Issue 3, July 2002.
- Yin, R. K. (1989) *Case Study Research: Design and Methods*, Sage Publications, Beverly Hills.

## **Paper C**

### **Managing networked software development**

Jeenicke, M., Nielsen, P.A., Vainio, M., and Vogelsang, L. "Managing Networked Software Development," *the 26th Information Systems Research Seminar in Scandinavia, CD-rom, Haikko Manor, Porvoo, Finland, 2003.*

# MANAGING NETWORKED SOFTWARE DEVELOPMENT

M. Jeenicke, Nielsen, P.A., Vainio, M., and Vogelsang, L.

## Abstract

Software development is increasingly performed in network organisations. The focus of attention is gradually shifting from the project as a the basic organisation towards a more networked type of organisation where relationships between the core project and other partners are established, exercised, maintained and dismantled. In this paper we propose a framework intended for software managers to make sense and diagnosis of the network and based on that points to possible action to be taken. The framework comprises two dimensions. In the one dimension the framework is based the theory of social capital that is used to distinguish between the coherent group of co-workers with a well-established community-of-practice where trust prevails and on the other hand a new relationship where formal control replaces trust. In the other dimension the framework is based on the concept of task uncertainty to distinguish between well-defined tasks that call for problem solving and on the other hand ill-defined tasks that call for problem definition. The framework is applied to four cases and is used to illustrate the network management in the four cases and explain some of the processes involved in the management of networked software development.

## 1. Introduction

The organization of software development is becoming more and more networked. Web engineering requires collaboration between artists, designers, engineers, and various managers. New business models of software development emerge. Outsourcing as a strategy for reducing development costs appears attractive to many software organisations, but it certainly has its challenges

Software organisations respond to increasing complexities of software development:

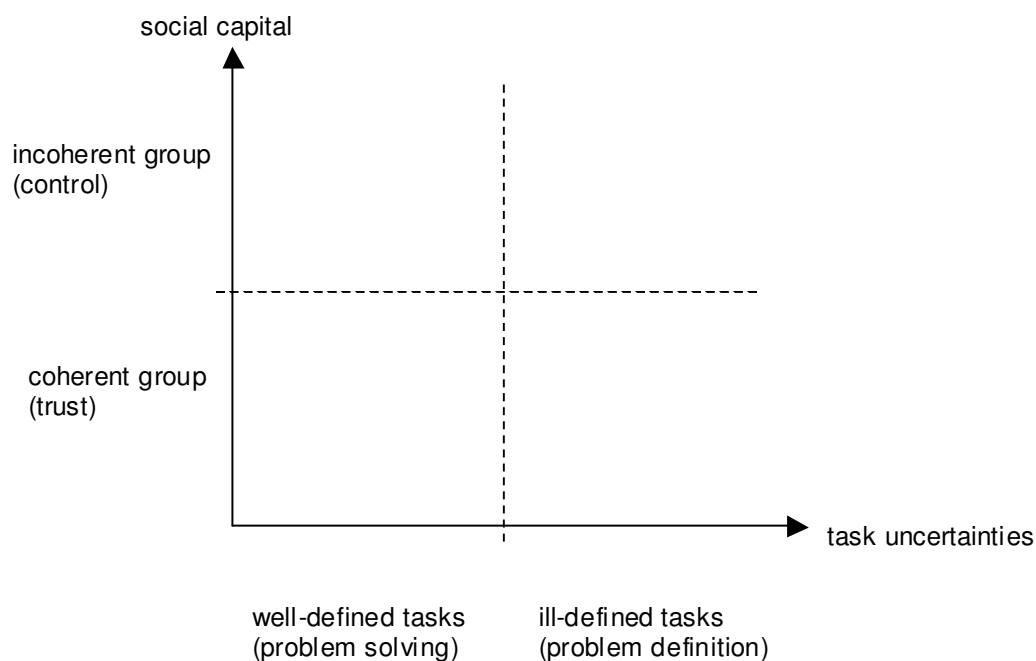
- **Teamwork:** with many people contributing their knowledge and talent to the development of a particular software system. The different team members are often separated both geographically and temporally, as they work on different parts of a larger system in different locales and on different schedules.
- **Networking:** computation is becoming a social process as software frequently runs as distributed applications on several machines at once, with many people interacting through them.

Therefore, embodying more and more interdependent knowledge, the development process by which all that knowledge has to be integrated becomes more challenging.

The paper is organised as follows. A two-dimensional framework is presented in the next section; the framework forms the lenses through which we study the cases. Section 3 provides four cases of software development illustrating different aspects of the conceptual framework. In section 4 we discuss in which ways the framework may be applied in diagnosis of development projects' situations and in suggesting management strategies and techniques for defining problems and for increasing social capital.

## 2. Framework

The framework comprises two dimensions. In one dimension the framework is based on the concept of task uncertainty to distinguish between well-defined tasks that call for problem solving and on the other hand ill-defined tasks that call for problem definition (Schön 1983). In the other dimension the framework is based on the relational view (Dyer and Singh 1998; Lane and Lubatkin 1998) and on the theory of social capital (Larson 1992; Nahapiet and Ghoshal 1998). Social capital is used to distinguish between the coherent group of co-workers with a well established community-of-practice where trust prevails and on the other hand the incoherent group and new relationship where formal control replaces trust and where network ties between co-workers are weak or absent. Figure 1 illustrates the two dimensions.



**Figure 1: The two dimensions of the framework**

Software development activities are more often carried out in heterogeneous networks of large and small organisations, universities and other knowledge institutes (Huff 2000; Rip and Groen 2001). Therefore, co-operation between participants of the software project has become an increasingly important aspect of management and is playing a major role when ensuring the success of software development. It is the complexities facing the managers and participants of software development that we depict in the two dimensions of the framework, see Figure 1.

The first dimension (the horizontal axis) shows the task uncertainties as perceived by the development project. In organisation theory, task uncertainty is often seen as one of the factors characterising an organisation's environment (Mintzberg 1992). The task uncertainty increases when the organisation delivers new products and services to its customers, when deliverables are unique, and when the organisation therefore cannot stabilise internal work practices. Task uncertainty decreases on the other hand when deliverables are mass produced, when deliverables are well defined, and when the organisation therefore can stabilise internal work practices. According to (Mintzberg 1992), the machine bureaucracy (based on standardisation of work) and

the professional bureaucracy (based on professions with common education and values) fit low task uncertainty while the simple structure (based on the top managers' leadership) and the adhocracy (based on mutual coordination) fit high uncertainty.

(Schön 1983) offers an account of the dimension that explains how the professional work practices differ with low and high task uncertainty. Facing low task uncertainty the project participants are in a routine situation that may be handled by technical rationality. Facing high uncertainty the project participants are in a unique situation that may be handled by what Schön calls reflection-in-action.

Technical rationality is characterised by the project participants being engaged in instrumental problem solving. The starting point for any task is a set of given objectives. The participants choose the optimal means to realise the objectives. Scientific knowledge for professional work is necessary to perform specific task and to select techniques that apply to different tasks. Tasks are well defined and categorised scientifically (Schön 1983, p. 21-26).

Reflection-in-action, on the other hand, is characterised by the project participants being engaged in unique situations that are both complex and uncertain. It is important for the participants to be aware of the uniqueness of the situation at hand. Knowledge and action are intrinsically related in the process of problem setting and problem definition. The project participants rely on a repertoire of techniques and action. The reflective participants conduct 'what-if?' experiments to advance the problem setting (Schön 1983, p. 128-140).

The second dimension (the vertical axis) shows the network ties in terms of social capital in a development project. We adopt the view similar to (Nahapiet and Ghoshal 1998) in arguing that social capital facilitates the creation of new intellectual capital in networked software development. Through close social interaction, participants of the development project are able to increase the depth, breadth, and efficiency of mutual knowledge exchanges in dyadic and network relationships (Dyer and Singh 1998; Lane and Lubatkin 1998). In line with the relational view, our focus is on how social capital in dyadic relationships between software project participants affects the management of networked software development. We also try to explore what capabilities are needed when managing software development in network context.

Knowledge is particularly important in software development: maintaining software's value demands that knowledge be continually replenished. One critical aspect of this exchange between participants is that the acquisition and exploitation of knowledge-based resources are predominantly social processes (Kogut and Zander 1992). Therefore, by building relation-specific assets and relational governance mechanisms into relationships, project participants are able to tap into the knowledge resources of their exchange partner. With the help of our framework, we present that the degree to which project participants are able to use dyadic relationships for value creation in software development is regulated by the amount of social capital embedded in such relationships. This perspective helps to explain how and why some software development projects are able to succeed in achieving goals within budget, delivering time, and managing risks despite the lack of significant project-specific resources.

We define social capital as the sum of the actual and potential resources embedded within, available through, and derived from the network of relationships possessed by participants of networked software development (Nahapiet and Ghoshal 1998). Central to the argument is that social capital influences:

- The resources actually exchanged between participants (a relational component, operationalized as social interaction).

- The resources available of the participant through his/her network of relationships (a structural component, operationalized as network ties).
- The efficiency of resulting knowledge transfers and exchanges (a cognitive component, operationalized as social control).

Based on the operationalizations, we argue that the degree to which project participants can use the dyadic relationships for value creation depends on three aspects of social capital in the relationship: 1) the level of social interaction between the participants, 2) the level of network ties created through the relationship, and 3) the control based on social relations in terms of trust, norms of co-operation and reciprocity. Social interaction refers to the extent of social relationships between participants of the software project (Larson 1992; Nahapiet and Ghoshal 1998) (Nahapiet & Ghosal 1998; Larson 1992). Network ties denote the extent that the counterpart provides the participant access to a broader network of contacts that could open opportunities (Uzzi 1997). Control based on social relations refers to the extent that this interaction is marked by the development of trust, norms of co-operation and expectations of reciprocity (Larson 1992; Nahapiet and Ghoshal 1998). Thus, we argue that high levels of social capital enhance value creation in software development in a network perspective by improving access to sources of knowledge resources, by increasing the willingness and ability of actors in a relationship to identify, exchange, and assimilate knowledge and by improving the breadth and efficiency of activity links consisting of knowledge transfer.

### 3. Cases

In this section present cases of development projects where the two dimensions are useful in understanding the complexities and how they might be handled.

The research approach we have applied is informed by longitudinal field research. Based on the research data we seek to provide contextualized understanding of the complex development projects we have studied. We take Pettigrew's stance to study organizational change in context, namely that it requires multilevel analysis (i.e., varying levels of analysis) and processual analysis (i.e., analysis of sequential, temporal and historical dependencies) (Pettigrew 1985; Pettigrew 1990). Crucial is also the element of time in the longitudinal study of organizational change. In our analysis of the research data we have focused on the organizational aspects that are put forward in our framework. More specifically, we have been through the research data seeking first of all to crystallize the most significant relationships and practices. We have then tried to see the patterns emerging in the data.

#### 3.1. The "Customer's Project Manager" Case

This case reports from the development of a web site used for publishing information and 'branding' (i.e., establish a specific impression of the organization) to a number of target groups outside the customer's organization. The task of the software company was to implement the web site, create a visual appearance, and help the customer with structuring the information to be published on the web site in order to target it towards the target groups, and finally design a navigation mechanism to navigate in the information on the web site.

The project lasted for 6 months and had an average of about 12 people working on the project and 18 people at the most. Different roles were present in the project. A group of information architects made information architecture for the web site i.e. they structured the information to be published on the web site. A group of graphical designers created the visual appearance for the web site. And finally a group of software developers implemented the web site. Each group consisted of 3-5

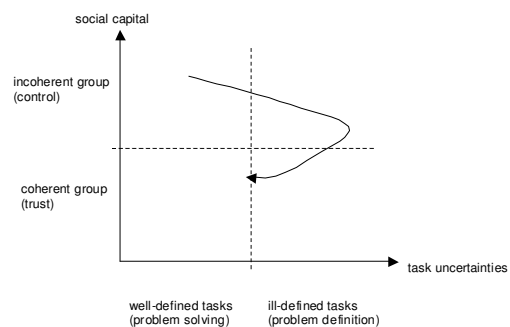
participants. Additionally, a project manager and a quality assurance person from the development company participated in the project.

The web site should reflect the customer's needs and requirements. To insure this, the customer had a project manager present in the project (we call him the customer's project manager). The customer's project manager's role was to develop ideas and provide input to the information architecture and the visual appearance. In order to do this, the customer's project manager talked and worked with persons from the different groups in the project. All the project participants were located in two offices. This made it possible for the customer's project manager to go to the different project participant and contribute with ideas and discuss the web site on a daily basis. The different parts of the design were created in each group, so a lot of communication and coordination among the project participants were needed in order to handle the contributions and ideas from the customer's project manager. The different contributions and ideas from the customer's project manager was not coordinated and communicated sufficiently in the project. This made it hard to establish a common understanding of the design of the project among the different groups in the project. Consequently different versions of the design were present in the project.

The problems with different versions of the design appeared when the software developers started working on the actual implementation. At that time different functionality and appearance could not be made (it was never intended to have different versions of the web site. There should be only one). The software developers should implement the web site based on the work of the information architects, the graphical designers, and project manager's Use Cases. The differences between these three versions were significant. This problem was resolved through conversations between the customer's project manager, the information architect, and the software developers. Transforming the different versions of the design was time consuming and frustrating for the project participants and was one of the reasons for the lack of time in the project which eventually forced the customer's project manager to decide not implement all the planned functionality.

This lead to some confusion about what the web site actually should be and the task became more ill-defined during the project. The problem was to decide what actually should be build based on the different designs. The designs described the web site differently, so some had to be changed and quite a significant amount of resources were wasted on designs that had to be redesigned and as important the resources that went into bringing the different designs into one final design.

In the model, the introduction of the customer's project manager made the project incoherent because a new project participant had to be included in the project. After a while the project manager was accepted as a contributing participant in or a member of the project working on particular tasks together with the project participants from the software company. On the well- and ill-defined task axis the project moved slightly towards ill-defined tasks because the different versions of the design were present in the project and became unclear for the project participant. Despite these problems the project was considered a success by the software company in terms of meeting a strict deadline and budget although some functionality was not implemented.



### 3.2. The “Whitebox” Case

In this section we present a case on the development of a web site used for collecting and sharing patient data on a specific disease. The patient data is planned to be collected by and shared between physicians, researchers, and other specialists all over the world. The purpose of the web site is to gather a large amount of data on the patients (more than hundred data fields) and share the data in order to find new areas where the medicine can be used and to give patients a more appropriate amount of medicine.

The web site was redesign and complete rebuild of an existing web site. The purpose of rebuilding and redesigning the web site, was to gather data of higher quality (a large parts data in the first version was written in text-fields and were difficult to base statistical analysis upon) and to improve the usability for the web site (to meet users' critique of the first version of the web site).

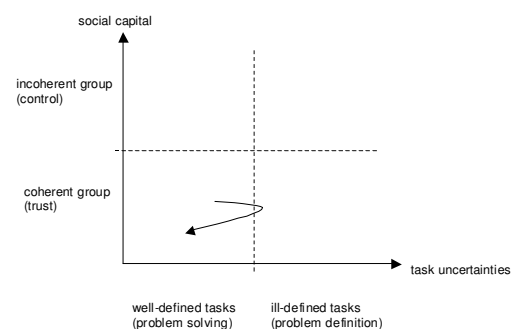
In the initial phase of the project, the requirements for the web site were primarily described in a data specification and in Use Cases. The data specification and the Use Cases were created by a product manager from the development company. The product manager had the initial contact with the customer and produced the data specification and the Use Case. After this initial phase and after a contract was signed, the project was staffed with a project manager, an information architect, a graphical designer, and four software developers. The project lasted for four months and was generally considered a successful project.

The data specification and the Use Cases are abstract requirements for the web site and something more specific has to be made to actually be able to implement the web site. In the "White box" case this process mainly consisted of specifying which elements that should be on the web pages and where these elements should be placed on each web page.

To specify which elements and where the elements should be placed on the web site the information architects in the project created a white box for each page (white boxes are also known as Wire Frames). A white box is a sort of paper-based prototype or mock-up, showing the elements (text boxes, radio buttons, menus, etc.) to be implemented on the web pages. A white box sometimes has a brief description or clarifications about the elements. For instance do some elements have be visible on a specific events such as a mouse click on button. Creating the white boxes does not include the design of the visual appearance of the web pages.

In the "White box" case the white boxes were not only used to specify what elements that should be on the web pages, but were also the developers' primary information source when they implemented the web site. Furthermore the white boxes were used by the graphical designer to create the visual appearance. Finally the white boxes were also showed to the customer in order to clarify the information architects interpretation of the data specification and Use Cases (the data that were going to be gathered on the web site were highly domain specific and the relations between the data are only understood by specialists). All the project participants were more or less depending on the white boxes. The white box was a critical artifact used to exchange information between the participants in the project and with the customer.

Based on the conducted field study the white boxes seemed to be used quite successful in respect to clarification of the design of the web pages and the exchange of information among the project participants. A reason for this is that the project participant had been sharing the same office for more than a half year working closely together on former projects and therefore knew each other well. Furthermore the participants had worked with the white box technique



before and knew what to expect and which of the other project participants they should ask if they had trouble in interpreting the white boxes. Informal talks about the white boxes were a part of the daily work practice. This made the work with the white boxes easier and consequently the project participants could focus more on the design and the implementation of the web site than on how to exchange information and use the white box technique.

At the outset of the project, the project participants knew the main idea with the project and a couple of them had been on the project that developed the first version of the web site, so the project was well defined at the outset. In our model the described project has moved from right to the left during the project because the design was created and the task became better defined. For instance did the use of white boxes show that the search functionality could not be used as intended, but the idea of having a search function was not discarded?

There were no significant move up or down in the model because the project participants knew each other from the outset and there were no replacement of the core participants in the project, so the project had a high social capital at the outset and during the project. The high social capital had a significant role in the use of the white boxes because the project participants knew when to pass problems to the other groups in the project and consequently an artifact like the white boxes became much more useful and valuable for the project.

### **3.3. The “Content Management System” Case**

In the third case, we describe a software project, in which the supplier (the software company) delivered the content management system to a large corporation (the customer) operating in the energy industry in 1999. The customer is a national grid operator that is responsible for ensuring the technical reliability of the electricity transmission system. The goal of the software project was to develop a content management system that would integrate all kinds of documents to the single database and therefore help the customer to make the transition to the digital age by transferring the manual report review process into the web-based system. With the help of the system, the customer would be able to review the documents more efficiently and effectively.

The project lasted one year and employed about twenty people during the development. The original goal was to accomplish the task within six months with the help of ten people. The permanent project team consisted of a project manager, an account manager, three programmers, two graphical designers, and three representatives from the customer, but the composition of the group varied a lot during the project. Also, a technology manager and a quality manager of the software company occasionally participated in the project.

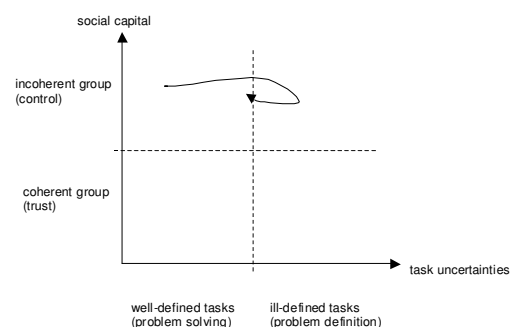
The project manager was responsible for the co-ordination of the project and the communication with the customer. The account manager had sold the project to the customer and during the project his role was mainly to keep track that everything was going well and according to the agreements. The sphere of responsibilities of three programmers consisted of design, programming, and testing of the software system. In addition to the development work, the programmers guided by the technology manager carried out the implementation effort. Two graphical designers designed the interface and the visual appearance of the software system. The quality manager offered his help as managing the follow-up reporting of the software project. The role of the customer’s representatives was to follow the project by reviewing the results accomplished within the milestones, such as memos and accomplishments of each development phase. One of the customer’s representatives was acting as a project manager on behalf of the customer organisation.

In the beginning of the project the requirements seemed static and easy to articulate. However, assuming static and clear requirements turned out to be wrong because the customer was mainly describing the processes of the current manual system ignoring the possibilities of improving the system with the help of new technology. Programmers had already started the software development, when the customer began changing the needs and requirements for the system. Basically, the users started to refine their requirements as they better understood what system features were possible with the means of new technology. Even though, they had difficulties to articulate and describe their needs explicitly because the system's requirements could not be fully known due to the lack of common understanding. Also, the customer expressed the need of increasing the extent of the project to cover more functions.

In addition to system requirements, there were many changes in the project organisation. The software company was scaling its operations aggressively and hired new employees almost weekly and made many reorganisations in project teams. Therefore, only the project manager and the account manager were not replaced during the project, but all programmers and designers were replaced many times. Two of the customer's representatives were replaced in the middle of the project with new managers. The project organisation was located in three different cities, which made social interaction between the project participants even looser.

In applying the framework to the case, we can suggest that the project shifted from well-defined to ill-defined and the amount of social capital remained low during the whole project. In the beginning of the project, the task seemed to be very well-defined and static. Gradually the goal started to evolve because of the changes in requirements, the changes in the extent of the project, and the replacements in the customer's project team. The need for a communication medium through which designers and users could reach reasonable confidence that they understood one another was high. Instead of freezing the requirements document on behalf of the customer, the requirements and software actually started to evolve to different directions throughout the lifecycle of the project. The lack of common understanding between the customer and the software company was due to low social interaction between project participants, formal meetings, and the lack of trust and mutual interdependence between project members. Because of the high rate of replacements, the project participants did not make friends with one another and were lacking of experience of working together.

Basically, the software company tried to manage the project with the means of traditional methodologies, which implicitly view that the requirements of the system can be analysed and understood adequately before the development begins. When the problems started to exist, the software company just tried to extract more information from the customer's representatives without considering the fact that the representatives were clearly unable to say what they wanted. Maybe an iterative approach to software development would have been a better option, if the users had "stayed in loop" and had been able to refine the requirements, as they better understood the system features. Also, the replacements in project group would have needed a different management approach.



### 3.4. The “Municipality Web Site” Case

The fourth case example is the development of a municipality web site. In 1999 the city of Hamburg decided to re-build the city’s web site in order to offer new and innovative online services to their citizens. In order to achieve this goal a company (called hamburg.de) was founded on the basis of a public-private-partnership. The city of Hamburg put this company in charge of managing the development process and operating the web site. Hamburg.de subcontracted a number of other companies in order to achieve this goal. First of all there was a newly founded organization for the development (and operation) of the municipality’s system as a product that could be offered to the councils of other towns. This organization hired three freelancers for development activities. Another big group in the project was formed by three members of a consulting company that was supposed to build the back-end system. A design company was hired for creating the layout of the web pages. In addition the Hamburg University’s IT transfer centre was involved for consulting the project.

The task(s) was/were not clear at the beginning of the project - only the goal of creating a new and innovative web site was defined. In order to define the system’s functionality in greater detail and to discuss relevant topics for the project as a whole, as well as current questions that arose in the ongoing development, weekly meetings were held. Attendees were always 5 to 7 persons from the above-mentioned organizations. When it came to the discussion of technical issues external persons were invited as well. E.g., a consultant from the producer of the content management system used took part in the meeting on a regular basis.

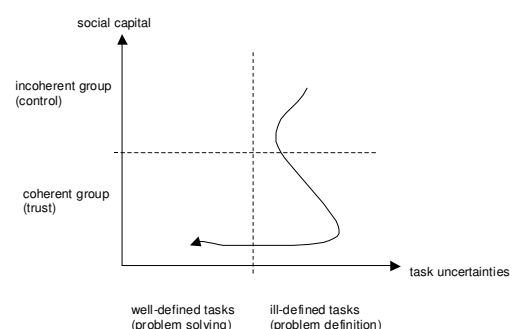
The weekly meetings were also held to foster communications between all project members. Yet, communication was often indirect and problematic throughout the whole project. One symptom was that only the manager of the hamburg.de company and very few people (not including the project participants) in the other companies knew the content of the contracts that were signed with the other companies. On the side of the commercial consultants this led to fear, which resulted in efforts to make the project a fixed price project. As a result the consultants suddenly wanted to create a complete system specification after half a year. And even though the tasks were not fully clarified, they wanted to implement exactly what was defined by this system specification.

Communication problems not only arose over organizational boundaries but also within the participating organizations. E.g., the three commercial consultants came from different departments of their organization and developed their parts without talking to each other. This resulted in parts that didn’t fit together very well.

The problematic communication also resulted in a number of misunderstandings. E.g., the screen designers’ task was to develop a layout for the web pages. They did it by designing it as a drawing, which was then divided into certain parts that needed to be put into an html-file. But nobody felt responsible for doing this work. E.g., the designer thought that their work was done by actually designing the layout while the (commercial) consultants said that were only supposed to build the back-end system.

Despite all problems a team spirit rose within the core team of about 4 persons that included the freelance developers and the consultant from the university.

The underlying development process can be called a “big-bang” approach, i.e. the development was going on for about 1¼ year without a single new release. But this approach failed because of a number of technology-related (e.g., acquirement of technology, performance,



interconnections and interoperability with other systems) as well as relationship-related (e.g., (those mentioned) misunderstandings, conflicts of interest) problems.

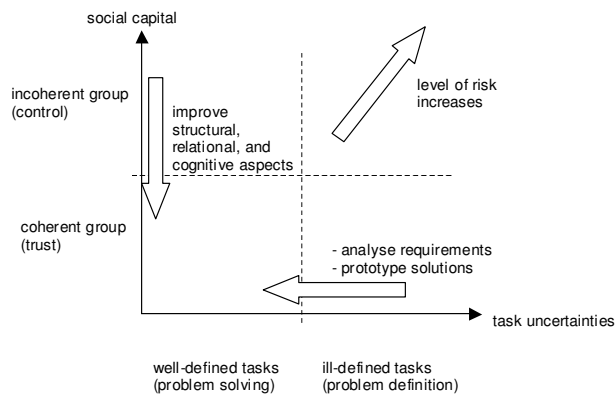
The project got a second chance but dramatic changes were made in the project structure. First of all the development team was reduced to a small size, including people that knew each other quite well. Also short development cycles were introduced and a consistent functionality visible at a glance, enabling a short “time to release/time to market”, was defined. The resulting system could go online about 3 months later.

In the model the project as described so far can be placed in the right upper corner. The group setting was new, the project members came from an open market and were chosen to join the project in a public bidding. The task was rather ill defined: To build new innovative online services is always a task of exploring new applications and technologies.

In addition the participants got to know each other better (at least the ones attending the meeting). As a result the project was moving down.

## 4. Managing Networked Software Development

The cases show that the management of networked software development has to consider the two dimensions of the framework. Risk management appears to be a management philosophy that is relevant and may be useful in addressing how to analyze a networked software development project. Risk resolution along the two dimensions appears to address how to manage these risks. The figure shows the relationships between risks and resolution techniques.



### 4.1. Risk Management

Software risk management approaches assume that a risk is an aspect of a development task, which will increase the likelihood of the project becoming a failure (Lyytinen, Mathiassen et al. 1998). Risk management approaches are applied to analyze various risks associated with software development (Boehm 1988; Charette 1989; Boehm 1991; Fairley 1994). (Davis 1982) is a stepwise approach for assessing the level of uncertainty and linking this to requirements techniques. (McFarlan 1982) links aggregate risk items as project size and experience with technology to one of eight strategies. It is common to see risk approaches as consisting of two activities:

1. Analyze risks

## 2. Manage these risks by acting to reduce, avoid, and handle risks

In our framework we take an increase in task uncertainty to increase the level of risk in general and task risks in particular. Similarly, we take a decrease in social capital to increase the level of risk in general and network organization risks in particular. It is not the purpose of this paper to provide a detailed approach for analyzing risks, yet the framework does in itself provide a simple overview of risks.

Risks may be managed by managing the application of a number of techniques. In our framework risks may be reduced by reducing the task uncertainty or by increasing the social capital. These two dimensions are related as the framework indicates. As Pohl (1994) states, the main characteristics of the desired output of the requirements engineering process are a complete system specification expressed using a formal language on which all people involved agree. This emphasises also the social context of software development because achieving common understanding between all members involved in the process requires the integration of different views at the representation level and the agreement on the integrated view among the people (Pohl 1994). A detected conflict must be solved through communication among people. Support for conflict resolution can be found in the area of computer supported cooperative work (e.g. Winograd and Flores 1986; Conklin and Begeman 1988). Additionally support can be offered through different representation, e.g. by providing informal knowledge for explanation of formal representations, by offering graphical representation for overview of the system, or by automated detection of differences between formal specification.

Three kinds of essential assistance for achieving common agreement can be identified (Pohl 1994):

- Different views and specifications must be maintained during the software development process.
- Support for detecting dissimilarities and inconsistencies between the different views must be offered. Contradiction for example can be made explicit through automatic reasoning and of course the work out of solution can be supported.
- An agreement can only be gained through communication among the involved people. Hence supporting communications, conversations, coordination and collaboration between people as well as decision support leads to better and possibly faster solutions.

## 4.2. Managing Task Uncertainty

Reducing risks due to task uncertainty is to manage techniques for analyzing requirements and prototyping software solutions. In order to move to better-defined tasks the management of a networked software project can make use of a number of well-known techniques.

- Requirements engineering is a set of activities that covers discovering, documenting, and maintaining requirements for a computer-based system (Kotonya and Sommerville 1998, p. 8).
- Negotiation with stakeholders also covers the gathering of information about existing systems, the stakeholders' needs, organisational standards, regulations, and domain knowledge (Kotonya and Sommerville 1998, p. 54) and what is also referred to as problem analysis and understanding (Davis 1993)

- Comprehensive methodologies like Soft Systems Methodology (Checkland 1981; Checkland and Scholes 1990), ETHICS (Mumford 1989), and User-Centered Design Approach (Eason 1988).
- Scenario analysis is a particular way of communicating with users; by writing a scenario the developers create an artefact that can then be the basis for a discussion with the prospective users (Carroll and Rosson 1990; Bødker 1995; Bødker 1999; Carroll 1999).
- Prototyping has emerged an approach that can be used with users to experiment with different use situations as well as technical issue (Floyd 1984).
- Scope reduction based on extreme programming (Beck 2000) and frequent and small releases (Abrahamsson, Salo et al. 2002).

### 4.3. Managing Social Capital

Reducing risks due to social capital is to manage an increase of social capital. Building social credit or a good reputation for cooperation in order to access resources or information as needed (Starr and Macmillan 1990). Improving the social capital falls in three dimensions: structural, relational, and cognitive.

The structural dimension of social capital refers to the overall pattern of connection between project members – that is, who you reach and how you reach them (Burt 1992). The appropriate organization of network ties, connectivity, and density may be managed by:

- Creating and improving ties between actors (Scott 1991)
- Improving network configuration (Krackhardt 1989)

The relational dimension of social capital refers those assets created and leveraged through relationships. These may be managed by:

- Improving trust and trustworthiness (Fukuyama 1995; Putnam 1995) – trust increases as uncertainty decreases
- Norms and sanctions (Putnam 1995) – creating and building the ways of working together
- Obligations and expectations (Granovetter 1985; Burt 1992) – commitment, keeping promises, building trust
- Identity and identification (Håkanson and Snehota 1995) – getting familiar with other project members in social level, improving coordination of information exchange and communication, establishing contacts

The cognitive dimension of social capital refers to those resources providing shared representation, interpretation, and system of meaning among parties (Cicourel 1973). These may be managed by:

- Shared language and codes (Cicourel 1973) – establishing ways of communicating together (reporting, memos, project management system) shared narratives (Orr 1990)

## 5. Conclusion

The goal of this study was to propose a new perspective on networked software development that places developers also in their social context. Research has already recognised that the social context provides information and resources for the innovators, both managers and all other project members (e.g. Starr and Macmillan 1990; Yli-Renko, Autio et al. 2001). Social context also defines the rewards, and therefore may shape the incentives of the members of software development project. But the dimensions, as important as there are, may need to be more specially linked to the cognitive process at a social level, if there are to illuminate the process of networked software development. Therefore, risk management needs to be extended to include the management of social capital and not only task uncertainties.

The foundation of managing risk in this broader sense is the analysis of the current project situation. This is an activity that can be supported by applying our framework. Just as a map is a tool helping the captain of a ship to navigate to his destination, our framework provides a navigation map for the project managers. In addition it includes valuable advises for the measures to be taken against a high level of risks should project be moved in a certain direction. Thus it helps coping with the new management challenges of networked software development.

We have not tried the model in ongoing projects yet. It has only been used as an analytical tool to look at documented projects. An interesting and important question is whether or the project managers are able to determine the “right” position in our framework or not. The project members probably see their social capital and the task uncertainty different than an external consultant. We do need to verify this question by applying the model in ongoing projects. Another open point is the question if there is something like a destination that a project should reach, something like a desirable state to be in for a project.

Social capital is often defined as consisting of both strong and weak ties between actors. Weak ties are defined Granovetter (1973) as relationships which involve infrequent contact and rarely have stable nature, while strong ties characterise more close friends or established relationships for pursuing a specific opportunity. This argument can be related to networked software development, too.

The project manager must find a balance between the need to establish stable ties for concept creation and technological cooperation and to build weaker ties for information gathering. It would be interesting to study more deeply the characteristics of different relationships among networked software development project and the effectiveness of such relationships for a particular task. Are some relationships used only for concept creation and technological cooperation (consultants, technology suppliers, customer’s IT department, other technology partners) and others for information gathering (users, customers).

## References

- Abrahamsson, P., O. Salo, et al. (2002). Agile software development methods: Review and analysis. Oulu, VTT Publications.
- Beck, K. (2000). *Extreme Programming Explained: Embrace Change*. Boston, Addison-Wesley.
- Boehm, B. W. (1988). "A Spiral Model of Software Development and Enhancement." *IEEE Computer* **21**(5): 61-72.
- Boehm, B. W. (1991). "Software risk management: principles and practice." *IEEE Software* **8**(1).

- Burt, R. S. (1992). *The Structural Holes: The Social Structure of Competition*. Cambridge, MA, Harvard University Press.
- Bødker, S. (1995). *The EuroCODE Conceptual Framework: Framework finalization- the process*. Aarhus, Aarhus University.
- Bødker, S. (1999). *Scenarios - setting the stage for reflection and action in user-centered design*. Proceedings of the Thirty-Second Annual Hawaii International Conference on System Sciences (CD-ROM).
- Carroll, J. M. (1999). *Five Reasons for Scenario-Based Design*. Proceedings of the 32nd Hawaii International Conference on System Sciences, Hawaii, Computer Society Press.
- Carroll, J. M. and M. B. Rosson (1990). *Human-Computer Interaction Scenarios as a Design Representation*. Proceedings of the 23rd International Conference on System Science (HICSS-23), Hawaii, Computer Society Press.
- Charette, R. N. (1989). *Software engineering risk analysis and management*, McGraw-Hill.
- Checkland, P. (1981). *Systems Thinking, Systems Practice*. Chichester, Wiley.
- Checkland, P. and J. Scholes (1990). *Soft Systems Methodology in Action*. Chichester, Wiley.
- Cicourel, A. V. (1973). *Cognitive Sociology*. Harmondsworth, England, Penguin Books.
- Conklin, J. and M. J. Begeman (1988). "gIBIS: A Hypertext Tool for Exploratory Policy Discussion." *ACM Transaction on Office Information Systems* **6**(4): 303-331.
- Davis, A. M. (1993). *Software Requirements: Objects, Functions and States*. Englewood Cliffs, New Jersey, Prentice-Hall.
- Davis, G. B. (1982). "Strategies for information requirements determination." *IBM Systems Journal* **21**.
- Dyer, J. H. and R. Singh (1998). "The Relational View: Cooperative Strategy and Sources of Interorganizational Competitive Advantage." *Academy of Management Review* **23**(4): 660-679.
- Eason, K. (1988). *Information Technology and Organisational Change*. London, Taylor and Francis.
- Fairley, R. (1994). "Risk management in software projects." *IEEE Software* **11**(2).
- Floyd, C. (1984). A Systematic Look at Prototyping. *Approaches to Prototyping*. H. Züllighoven. Berlin, Heidelberg, New York, Springer.
- Fukuyama, F. (1995). *Trust: Social Virtues and the Creation of Prosperity*. London, Hamish Hamilton.
- Granovetter, M. (1973). "The Strength of Weak Ties." *American Journal of Sociology* **78**: 1360-1380.
- Granovetter, M. (1985). "Economic action and social structure: The problem of embeddedness." *American Journal of Sociology* **91**: 481-510.
- Huff, A. S. (2000). "Presidential address: Changes in Organizational Knowledge Production." *Academy of Management Review* **25**(2): 288-294.
- Håkanson, H. and I. Snehota (1995). *Developing Relationships in Business Networks*. London, Routledge.
- Kogut, B. and U. Zander (1992). "Knowledge of the Firm, Combinative Capabilities and the Replication of Technology." *Organization Science* **3**(3): 383-397.
- Kotonya, G. and I. Sommerville (1998). *Requirements Engineering : Processes and Techniques*. Chichester [u.a.], Wiley.
- Krackhardt, D. (1989). "Graph Theoretical Dimensions of Informal Organisation." *Paper presented at the annual meeting of the Academy of Management* Washington, DC.
- Lane, P. J. and M. Lubatkin (1998). "Relative Absorptive Capacity and Interorganizational Learning." *Strategic Management Journal* **19**(5): 461-478.

- Larson, A. (1992). "Network Dyads in Entrepreneurial Settings: A Study of the Governance of Exchange Relationships." *Administrative Science Quarterly* **37**(March): 76-104.
- Lyytinen, K., L. Mathiassen, et al. (1998). "Attention shaping and software risk: a categorial analysis of four classical approaches." *Information Systems Research* **9**(3).
- McFarlan, W. (1982). "Portfolio Approach to Information Systems." *Journal for Systems Management*(January): 12-19.
- Mintzberg, H. (1992). *Structure in Fives: Designing Effective Organizations*, Prentice-Hall.
- Mumford, E. (1989). User Participation in a Changing Environment - Why We Need It. *Participation in Systems Development*. K. Knight. London, Kogan Page.
- Nahapiet, J. and S. Ghoshal (1998). "Social Capital, Intellectual Capital, and The Organizational Advantage." *Academy of Management Review* **23**(2): 242-266.
- Orr, J. (1990). Sharing knowledge, celebrating identity: Community memory in a service culture. *Collective Remembering*. D. Edwards. London, Sage.
- Pettigrew, A. M. (1985). Contextual Research and the Study of Organizational Change Processes. *Research Methods in Information Systems*. E. Mumford, North-Holland.
- Pettigrew, A. M. (1990). "Longitudinal Field Research on Change Theory and Practice." *Organization Science* **1**(3): 267-292.
- Pohl, K. (1994). "Three dimensions of Requirements Engineering: framework and its application." *Information Systems* **19**: 243-258.
- Putnam, R. D. (1995). "Bowling alone: America's declining social capital." *Journal of Democracy* **6**: 65-78.
- Rip, A. and A. J. Groen (2001). Many Visible Hands. *Demands, Markets, Users and Innovation*. R. Coombs, K. Green, V. Walsh and A. Richards, Edward Elgar.
- Schön, D. (1983). *The Reflective Practitioner: How Professionals Think in Action*, Basic Books.
- Scott, J. (1991). *Social Network Analysis: A Handbook*. London, Sage.
- Starr, J. A. and I. C. Macmillan (1990). "Resource Cooptation via Social Contracting: Resource Acquisition Strategies for New Ventures." *Strategic Management Journal* **11**: 79-92.
- Uzzi, B. (1997). "Social Structure and Competition in Interfirm Networks: The Paradox of Embeddesness." *Administrative Science Quarterly* **42**: 35-67.
- Winograd, T. and F. Flores (1986). *Understanding Computers and Cognition: A New Foundation of Design*. NJ, Ablex Norwood.
- Yli-Renko, H., E. Autio, et al. (2001). "Social Capital, Knowledge Acquisition, and Knowledge Exploitation in Young Technology-Based Firms." *Strategic Management Journal* **22**: 587-613.

## **Paper D**

### **Managing knowledge in method utilization**

Mathiassen, L. and L. Vogelsang (2005) "The Role of Networks and Networking in Bringing Software Methods to Practice", *International Journal of Business Information Systems*, Vol. 1(1/2), pp. 102-117.

# Managing knowledge in software method adoption

Lars Mathiassen\*

Center for Process Innovation  
J. Mack Robinson College of Business  
Georgia State University  
P.O. Box 4015, Atlanta, GA 30302-4015, USA  
Fax: +1(404) 463-9292  
E-mail: lmathiassen@gsu.edu  
\*Corresponding author

Lasse Vogelsang

IT-University of Copenhagen  
Rued Langgaards Vej 7  
2300 København S. Denmark  
E-mail: vogelsang@itu.dk

## Abstract:

The complex and problematic relationship between software development methods and practices is well documented. Methods are difficult to adopt. There are several barriers toward successfully bringing methods to practice, and practices are, by the end of the day, quite different from method prescriptions. Methods are, however, continually developed, and organisations spend considerable resources trying to improve practices through adoption of new methods. We know relatively little about the long-term dynamics involved in such adoption efforts. This research reports from a three-year effort within an IT department of a large multinational company in which a new method was first introduced, then used to support organisation-wide projects, and subsequently extended based on emerging practices. The study applies two complementary knowledge management perspectives – networks and networking – to interpret the experiences from the case. The study reveals how method perceptions and approaches shift radically through different stages of software method adoption.

**Keywords:** software development; method adoption; knowledge management.

## 1 Introduction

For decades, software development methods have been invented and promoted to support and improve software development. Software development is arguably a knowledge-intensive human activity, and methods for software development attempt to codify relevant knowledge to be shared among practitioners of the discipline. Several studies suggest, however, that the relationship between software practices and development methods is far from simple and straightforward. Often, methods are not used as prescribed; they are used differently than intended, or they are used in fragments in combination with other tools and adopted practices (Bansler and Bødker, 1993; Fitzgerald, 1997; Baskerville and Stage, 2001; Lyytinen and Rose, 2003).

This research studies in detail how a particular method was introduced, used and further extended within SoftPharm, the IT department of a large multinational pharmaceutical company. The purpose of the study is to understand the different ways in which codified knowledge in the form of methods impact and are impacted by software practices over longer periods of time. Such longitudinal studies might reveal complementary insights into the relationship between methods and practices

that can help us understand and manage methods as useful knowledge resources for practice. In particular, they can suggest different types of tactics to bring methods into practice at different stages of adoption.

## 2 Theoretical background

Our paper draws upon three streams of research. First, it builds on existing knowledge on the use of methods in software practice (Section 2.1). Second, it relates to the existing knowledge on the adoption of software innovations in organisations (Section 2.2). These two bodies of knowledge provide insights into our area of concern – the adoption and use of software development methods. Finally, we present two complementary perspectives from knowledge management, those of networks and networking, which we use as lenses to make sense of the experiences from the studied case (Section 2.3).

### 2.1 *Software development methods*

It is generally assumed that methods have a positive influence on software development (Fitzgerald, 1996). In a survey, Chatzoglou (1997) found that the use of methods is beneficial in terms of economy and process improvement. At the same time, however, field studies show that methods are not as widely used as expected and not used in their entirety (Bansler and Bødker, 1993; Fitzgerald, 1997; 2000). Fitzgerald (1998b) found that methods are not the panacea for problems in software development, and they are not applied rigorously or in a uniform fashion. Mathiassen and Munk-Madsen (1986) argue that methods are appropriate in some situations and not in other situations. Each method has an application domain in which it works well. The literature suggests that methods are useful, but also that their use is highly situational, limited to particular application domains and often different from what one expects.

The relationship between methods and practice is indeed complex. Methods can have different roles in practice. Fitzgerald (1998a) found that methods play two diametrically opposed roles, an overt (rational) role and a covert (political) role. Baskerville and Stage (2001) argue that methods emerge in practice through interaction between the environment and the method. Bansler and Havn (2003) found that software developers work in more complex and less stable situations than assumed in most methods; therefore, we should abandon a rational and method-dominated view of practice. Method engineering is situational and offers method fragments for tailoring methods to specific situations (Brinkkemper, 1996; Welke and Kumar, 1991). Van Slooten (1996) proposes a framework for situated method engineering that includes a set of situational factors to guide the selection of method fragments. This approach also supports that methods can change as software projects improve their capabilities over time.

The inherent complexity of the relationship between methods and practice has implications for how we adopt and work with methods. Truex and Avison (2003) identify five main types of method engineering. They range from having a narrow technical focus to a broader focus that includes processes and organisational issues. Furthermore, they argue that the purpose of using methods has changed from aiming at universal support to supporting software practices in particular contexts. Nuseibeh et al. (1996) extend method engineering to include different project participants' perspectives on the system and the business domain it supports. This makes method engineering dependent on both the situation and the persons involved. Hidding (1997) investigates to what extent methods are read by practitioners and how to present a method to people with different roles. Hidding found that practitioners want methods to be presented in summary and additional information to be accessible as needed. Persons in different roles need different versions of a

method; this is a challenge to method engineers to provide a variety of building blocs for effectively communicating methods and method fragments.

Fitzgerald (2000) argues more fundamentally that contemporary software development methods are based on assumptions about the organisational context that are derived from the period from 1967 to 1977. The organisational context has, however, changed during the last 30 years. Organisations and applications have become increasingly complex. Fitzgerald claims that new methods and work practices are needed to meet this challenge. Further, he argues that we can learn from current best practices to explicate and create new technologies because practice often is ahead of theory and research. This contention is supported by empirical studies. Bansler and Bødker's (1993) field study of how developers apply structured analysis in practice reveals that some concepts and tools from structured analysis were used by the developers, but the procedures in structured analysis were not followed. They conclude that software development methods are based on assumptions and offer prescriptions that seldom are practiced. Fitzgerald (1997) found in his field study that methods are tailored to meet needs and that developers omit certain aspects of methods, not because they are ignorant, but because the omitted aspects are irrelevant for practice in the specific context. Further, he found that method use is correlated with developers' experience. Developers with low experience use methods as a template for practice (high usage), but eventually find out that methods are not universally applicable (low usage). They end up with a tailored version of methods (middle/average usage).

Our existing knowledge on the practical use of software development methods shows that practices are quite different from the intended use of methods, that methods are often used in a fragmented way in combination with other practices and, finally, that software practices encompass many complex issues that cannot be covered as general, codified knowledge in methods.

## *2.2 Adoption of software innovations*

Our knowledge about the practical use of methods is supported by a portfolio of studies of organisational adoption of software innovations. Leonard-Barton (1987) has studied factors that have a positive influence on the adoption of software methods. He found that programmers are more likely to use methods if supervisors, influential peers and their clients support the methods' use. Sharmaa and Rai (2003) studied factors affecting the adoption of CASE tools. They found that positional power and job tenure of software managers are negatively related to the adoption of CASE tools. Premkumar and Potter (1995) found that the following five variables were important to differentiate adopters from non-adopters of CASE tools: existence of a product champion, strong top-management support, lower IT-related expertise, perception that CASE tools have an advantage over other technologies, and cost-effectivity of CASE tools. The non-distinguishing factors for adoption and non-adoption were complexity and organisational and technical compatibility. Kozar (1989) found adopters to be younger, to have spent less time in their organisation and in software practice, not to currently use a method and to be more optimistic about being able to fit methods into organisational practices. Zmud (1982) investigated centralisation and formalisation as factors influencing adoption of software technologies. Zmud (1984) applied the push-pull theory and found that the theory was not validated for software technology adoption. Instead, he found that the type of innovation, to some extent, influences the organisational adoption process. Sherif and Vinze (1999) investigated the barriers to adoption of software reuse at the organisation and individual levels. They found that barriers at the individual level were caused by barriers at the organisation level.

A wide range of factors that affect adoption of software innovations is summarised by Mathiassen and Sørensen (1997) into five questions that need to be addressed when adopting new software technologies. The questions are the following: why to implement, what to implement, which technology to implement, where to implement and how to implement the technology. They recommend that software technology implementation is undertaken as an iterative process, re-addressing the five questions before starting a new iteration. Mathiassen and Sørensen (1996) also discuss the adoption of CASE tools. They argue that practitioners lack knowledge about CASE tools and suggest experiments as means for gaining this knowledge. Furthermore, they suggest that not all aspects of CASE tools have to be utilised, especially when the development organisation is on the lower levels of the capability maturity model. Finally, it should be taken into consideration that CASE tools are not introduced in an organisational 'vacuum' and that CASE tools influence the organisation while the organisation at the same time influences the CASE tool. In line with this, Fichman and Kemerer (1997) argue that the assimilation of complex technologies takes place through a process of organisational learning (Attewell, 1992). They found that organisations are more likely to initiate and sustain assimilation of software innovations when they have a greater scale of activities over which the cost of learning can be spread, when they have more extensive knowledge related to the innovation and when they have a greater diversity of technical knowledge and activities.

We have found only few studies of adoption of software methods over longer periods of time; none of these studies focus on how the role of methods changes over time. Lyytinen and Rose (2003) show that several diffusion of innovation (Rogers, 1983) factors such as past experience, own trials, ease of use, learning by doing and standards strongly affect the adoption of process innovations related to information technology development. Lyytinen and Rose focus on the factors influencing the decision to adopt a process innovation (such as a method) but not to what extent the innovation is actually used after the adoption. Further, Lyytinen and Rose focus on several innovations and the reasons for adopting them, but not on how one particular innovation is used and changed over time. Another example of a long-term study of method usage is Orlikowski's study of CASE tools implementation (Orlikowski, 1993). Orlikowski (1993) shows that the implementation of CASE tools involves organisational change over time. But Orlikowski focuses on the approach to implement CASE tools (radical and incremental) and not on how the role of CASE tools changes over time.

Existing literature identifies several factors that facilitate or inhibit adoption of software innovations. The enablers include support from stakeholders, technology advantages and prior experience. Reported barriers include technology complexity, lack of compatibility and traditions. Based on these, guidelines are reported to support adoption practices. Despite this body of knowledge, there continues to be widespread assimilation gaps between initiated and completed adoption initiatives in software organisations. Most of the available research focuses on the adoption process; the literature provides few discussions of the relationship between adoption and later use of methods. We seem to know little about the long-term dynamics that take place as methods are brought into practice.

### *2.3 Knowledge management*

In the following, we analyse data from a longitudinal study of adoption and subsequent use of a software development method from a knowledge management point of view. To do that we adopt Swan et al. (1999) framework of diffusion of innovations seen as knowledge creation and sharing. This model distinguishes between two complementary perspectives on organisational

implementation of technology – networks and networking (Table 1). The networks and networking perspectives represent a fundamental distinction within the knowledge management literature. It corresponds to the distinction between codification and personalisation as suggested by Hansen et al. (1999) and the distinction between organisational and technical as described by Tiwana (2002,p.294).

**Table 1** Networks and networking

<i>Networks</i>	<i>Networking</i>
The critical success factor is technology.	The critical success factor is trust and collaboration.
Knowledge for innovation is equal to objectively defined concepts.	Knowledge for innovation is social constructed and based on experience.
Knowledge can be codified and transferred through networks: IT has a crucial role.	Much knowledge is tacit, shared and made sense of through active networking within and between groups and teams.
Gains from KM include exploitation through the recycling of existing knowledge.	Gains from KM include exploration through the sharing and synthesis of knowledge among different communities-of-practice.
The primary function of KM is to codify, capture and transfer knowledge through networks.	The primary function of KM is to encourage knowledge sharing through networking.
The dominant metaphors are the human memory and the jigsaw (fitting pieces of knowledge together to produce a bigger picture in a predictable way).	The dominant metaphors are the human community and the kaleidoscope (creative interactions producing new knowledge in sometimes unpredictable ways).

*Source:* Swan et al. (1999)

The networks perspective emphasises the use of technology as a means for knowledge sharing and integration. Knowledge is perceived as objective and unproblematic to codify and transfer through networks of technology. The gains from knowledge sharing and integration are achieved through the exploitation of knowledge. The main goal of knowledge sharing and integration is the support of human memory. In the context of software methods, the network perspective implies a focus on creating the right method for a development context. The gains from the method are its reuse and the sharing of best practices. Best practices are perceived as easy to codify, transfer and integrate into existing software practices.

In contrast, the networking perspective focuses on trust and collaboration among the involved practitioners. Face-to-face interaction and sharing of tacit knowledge is recognised as an important part of knowledge sharing. Knowledge is seen as being socially constructed and based on personal experience. Knowledge is shared and made sense of within communities-of-practice through exploration. The primary function of knowledge management is to encourage knowledge sharing through networking. In the context of software methods, the networking perspective implies sharing and integration of methods through networking within groups and teams. The critical success factor is the exploration through sharing and synthesis of the use of a method in social groups. The primary function of the method is to encourage knowledge sharing through networking.

### **3 Research method**

Our study addresses the following research question: What is the role and dynamics of networks and networking during organisational introduction, use and innovation of software development methods?

We have researched this question through a longitudinal, interpretive case study (Walsham, 1993; Yin, 1994). The strengths of this approach is that it allowed us to study the dynamics involved in bringing a method into use over a three-year period and to gain a deep understanding of how and why the relationship between method and practice evolved from a knowledge management point of view. The limitation is that our study draws upon a particular software development method and a specific organisational context. The findings are therefore exploratory in nature.

The empirical work took place between January 2000 and June 2003. The data is primarily collected in three projects: the Method Project, the Development Project and the Extension Project. The field study was undertaken as a practice study (Mathiassen, 2002).

The Method Project was undertaken to adopt a method and to overcome challenges faced in the information technology department at SoftPharm. The project lasted for ten months. Data collection was based on observation, semistructured interviews and document analysis. We observed and took notes during weekly meetings in the project. Eight semistructured interviews were conducted. Four of the interviews took place a couple of months into the project, and the remaining four interviews took place after the project was finished. The interviews focused on themes related to software development, to the adoption of the method and to the Development Project where elements from the method were tried out.

The purpose of the Development Project was to develop a web application that will enable doctors to collect patient data. The Development Project was initiated in the fall of 2002 and lasted for about five months. The main data collection technique was observation. Two researchers spent 1–2 days a week for four months on observing meetings and other activities in the Development Projects. Additionally, several informal talks took place, and documents from the project were collected. After the web application was developed, four semistructured interviews were conducted with key participants in the project. A working paper with preliminary findings was written and discussed with the interviewees.

The Extension Project was undertaken to codify a work practice. The work practice was to sketch user interfaces, and it emerged among usability people in the information technology department at SoftPharm. The work practice was codified to make it part of the method (i.e., the method that was adopted in the Method Project). The project took place during the fall of 2003 and lasted for four months. The research was undertaken as participant observation. The purpose was to investigate the process of extending the method. The main contribution to the project from the researcher was the analyses of relevant work practices and participation in the meetings as a collaborator (Baskerville and Wood-Harper, 1998). Project meetings were recorded and documents were collected for analysis. After the project ended and the method was extended with new standardised practices, a meeting with the project manager took place to find out to what extent and how successfully the extension was in use.

### **4 The case**

The introduction, use and innovation of the software development method at SoftPharm are presented in the following. First, we examine the three projects from a network perspective (see Table 1). We then examine the same three projects from a networking perspective (see Table 1).

Finally, we synthesise the findings by discussing how the method was brought into practice from a knowledge management point of view.

#### 4.1 A networks perspective

The interpretation of the case from a networks perspective is summarised in Table 2. The three columns named Introduction, Use and Extension relate to the three projects studied. The Introduction column covers the Method Project, the Use column the Development Project and the Extension column the Extension Project.

**Table 2** A changing networks view on the method

<i>Introduction</i>	<i>Use</i>	<i>Extension</i>
<ul style="list-style-type: none"> <li>• Method introduced to solve problems in <i>SoftPharm's</i> projects</li> <li>• Method is a codified and customised process</li> <li>• Same method reused across all projects</li> <li>• Method promoted as <i>SoftPharm's</i> new development practice</li> <li>• Method transferred based on information, seminars, reading, and intranet</li> </ul>	<ul style="list-style-type: none"> <li>• Method is used as an organisation-wide framework</li> <li>• Templates, standards, and process models are reused from project to project</li> <li>• Exemplar documents used to guide method use</li> <li>• Knowledge about method available on intranet</li> <li>• Method repository serves as database and frame of reference</li> </ul>	<ul style="list-style-type: none"> <li>• Method offers framework for codifying and sharing new method fragments</li> <li>• New diagramming technique was codified to fit within method</li> <li>• The codified diagramming technique was made available on the intranet</li> <li>• The purpose was to transfer and reuse new knowledge between projects</li> </ul>

The aspects of the network perspective present in the Method Project were highly related to the reason for adopting the method – to solve major problems experienced within SoftPharm's development projects. The main problems were the ad hoc development and the lack of conformity among the development projects. The idea was to use a method to describe a certain development process to be used in the information technology department. The intention was to adopt and reuse the same version of the method across all projects. The method was in this way conceived by the Method Project participants, as a codified representation of processes, concepts and notations with related tools that could be transferred among projects. The main aim in the Method Project was to customise the method to SoftPharm by selecting a subset of features and by promoting the customised method as a comprehensive representation of SoftPharm's new development practice. The transfer of the method to the developers was initially based on announcements, seminars, reading and publication in SoftPharm's intranet. The Method Project's participants were mostly concerned about getting the method right.

The use of the method in the Development Project revealed several elements of the networks perspective. First, some of the participants had worked with the method before and also shared similar approaches to development. This helped them avoid inventing and debating everything from scratch. The project participants perceived the method to be a feasible approach to develop software in general. They also found that it provided useful support through templates, standards and process models. The templates, standards and process models were reused from project to project and used as exemplar ways of using the method. The method was seen as a repository that provided a useful database and a frame of reference for projects within SoftPharm. The method and knowledge about

its use within SoftPharm was available on the intranet and used as an organisation-wide framework for software development within SoftPharm.

The extension of the method in the Extension Project related to the network perspective primarily through the attempt to explicate the sketching practice into a diagramming technique, which could then be codified and added to the method. The purpose of codifying the diagramming technique was to transfer and reuse it among projects and people within SoftPharm. The method offered the project participants a framework for codifying and sharing new method fragments. However, it also constrained the project participants through requirements given in the method to be met in order to fit the codified diagramming technique into the method. Finally, the codified diagramming technique was made available on the intranet so that the new, codified practice could be accessed and shared.

#### 4.2 A networking perspective

The examination of the same three projects from a networking perspective is summarised in Table 3.

**Table 3** A changing networking view on the method

<i>Introduction</i>	<i>Use</i>	<i>Extension</i>
<ul style="list-style-type: none"> <li>• The method represents an opportunity to reflect on and innovate practice</li> <li>• Experiments are conducted to assess the method</li> <li>• Experiences from experiments are used to customise the method</li> <li>• The adoption group shares and synthesises experiences</li> <li>• A community-of-practice develops amongst the adopters based on the method</li> </ul>	<ul style="list-style-type: none"> <li>• The method helps transfer and transform practices from previous projects</li> <li>• Project participants and process engineers tailor method to each project</li> <li>• The method complements rather than replaces developers' knowledge and experience</li> <li>• Collaboration between project participants remains a critical success factor</li> </ul>	<ul style="list-style-type: none"> <li>• The method is a mirror for debating noncanonical practices</li> <li>• Knowledge about diagramming is explored and shared across projects</li> <li>• People with complementary diagramming skills collaborate to codify method fragment</li> <li>• Creation of a minimalist method fragment because much diagramming knowledge is tacit or unique</li> <li>• Innovation leads to improved, shared understanding of diagramming issues</li> </ul>

The adoption of the method in the Method Project was an opportunity to reflect on and innovate software development practices in SoftPharm. Part of the aim of the Method Project was to conduct an experiment to assess the strengths and weaknesses of the chosen method. The experiences from the experiment were used to customise the method. Additionally, the adoption group shared and synthesised a variety of experiences from different projects and departments in SoftPharm to facilitate the adoption of the new method. One of the outcomes of the Method Project was a community-of-practice among the project participants.

In the Development Project, the use of the method was a way to inherit and transform software development practices from previous projects to the current project. The method complements, but does not replace the project participants' knowledge and experience of working with each other. Some of the work processes were unplanned and emergent, and informal interactions between different groups in the project were a key to solving coordination problems in the project. The

project participants found that their individual skills and their ability to collaborate and solve problems were a critical factor for the success of the Development Project.

In the Extension Project, knowledge about diagramming problems and solutions was explored and shared among specialists from different projects in order to develop and explicate a diagramming technique. Much of the diagramming knowledge was considered tacit or unique for specific projects, so a complete technique that could be used for all situations was discussed, but not considered to be realistic. A minimalist approach was therefore adopted to codify a diagramming technique and to establish minimum standards for future projects. Besides the creation of the diagramming technique, an important outcome of the project was an improved, shared understanding of diagramming issues. The shared understanding was achieved because the development and explication of the diagramming technique became a mirror in which non-canonical practices could be identified and debated. This was an important activity because the project participants in the Extension Project had complementary diagramming skills and needed to collaborate and share experiences to codify the new method fragment.

### *4.3 Knowledge management synthesis*

We have synthesised the two separate analyses of how the method was brought into practice into a comprehensive knowledge management analysis (Table 4). The ‘Metaphor’ illustrates the primary intention with the use of the method. The ‘Method’ provides a more specific description of the method use. The ‘Networks’ and ‘Networking’ provide our assessments of the importance of the two perspectives on knowledge management in each project.

The aim in the Method Project was to adopt a method to remedy problems that occurred in the development process due to new tasks and use of new technology in the information technology department. The Method Project was focused on codifying and customising the method in order to create a version which would both fit the context in the information technology department and at the same time provide a new development process. The new version of the method was created by selecting and modifying elements of a standard method. The selection and customisation of method fragments were the primary focus in the Method Project. The focus on the method shows that the introduction of the method into SoftPharm was dominated by the networks model. Through experiments, the adopters developed a community-of-practice around the new method. These activities were based on the networking model by emphasising the exploration of the knowledge provided by the standard method. The two complementary knowledge management approaches, networks and networking, were both present in the Method Project; the networks approach, however, played a more dominant role than the networking approach. The method was perceived as a solution for problems in the software process. Therefore, we have chosen ‘method as solution’ as a metaphor for method use in the Method Project.

In the Development Project, the method was used directly for software development. The primary purpose was to support the project and its participants with resources such as a common language, a standard process and descriptions of roles, templates and techniques. The use of the method was optional in the project. Method fragments were explored and used by the project participants to share and synthesise a useful practice. Through the process of exploring the method, the users became a community-of-practice with a shared approach to and understanding of key development practices. The method was tailored to suit the Development Project. The tailoring shows an emphasis on the networks approach, although getting the method right was not emphasised as much as personal relations and experiences. The emphasis on personal relations indicates that the use of the method in the Development Project was dominated by the networking perspective. The method

was perceived as support for the project participants in the Development Project. Therefore, we have chosen ‘method as support’ as a metaphor for the Development Project.

**Table 4** Changing perceptions of the method

	Introduction	Use	Extension
Metaphor	Solution	Support	Repository
Method	<p>Codified process</p> <p>Customised version created through selection and modification</p> <p>Customised version represents new development process</p>	<p>Resource for projects</p> <p>Language for communication about practice</p> <p>Support is provided for tailoring method to each project</p>	<p>Framework for codification</p> <p>Repository for storing method fragments</p> <p>Method fragments emerge from practice</p>
Networks	Very high weight	Medium weight	High weight
Networking	Medium weight	High weight	Medium weight

The aim in the Extension Project was to create a method fragment consisting of a technique for sketching user interfaces. The method was used as a framework for codifying an emergent development practice by providing guidelines for the creation of the new fragment. The method fragment was codified and added to the method. By codifying and including the method fragment, the method became a repository for standardising development practices and storing method fragments. The purpose of standardising and storing the method fragment was to develop a work practice that conformed better with other practices within the information technology department. The codification of development practices within the method framework was dominated by the network perspective. The main focus in the Extension Project was on codification and integration of development practices. The method innovators formed a temporary community-of-practice in which they shared and synthesised experiences. Forming this temporary community-of-practice was perceived as important, this, however, was not the main aim of the project. Therefore, networking perspective is assessed to be of medium importance. The method provided guidelines to and a repository for storing the new method fragment. The usage of the method can be perceived as a repository for work practices; therefore, we have chosen ‘method as repository’ as a metaphor for the Extension Project.

## 5 Discussion

The analysis of the case illuminates the role and dynamics of networks and networking perspectives and approaches during organisational introduction, use and extension of a software development method. The analysis shows that changes occur over time with respect to both the strategy for sharing and integrating the method and the relationship between the method and work practices.

The project participants in the Method Project emphasised getting the method ‘right’ and saw social aspects of the future method usage as less problematic (i.e., the networks perspective dominated). The project participants in the Development Project perceived the method as helpful in developing software and found the collaboration between the project participants to be more important than the

method itself (i.e., the networking perspective dominated). Finally, the purpose of the Extension Project was to create a method fragment by codifying a work practice to make it conform better to other method fragments and make it available through the method (i.e., the networks perspective dominated).

The study illustrates how the emphasis on networks and networking for knowledge sharing and integration changes over time as the method is brought into practice. This change in emphasis suggests that adoption of software innovations, like a software development method, is not so much about choosing one particular knowledge management strategy to facilitate the adoption. It is more about combining different strategies to suit the situations and the purposes of adopting the innovation as it evolves over time.

Orlikowski (1993) suggests two strategies for implementing a process innovation, which is a radical and incremental strategy. While this distinction is different from the networks and networking approaches, our research suggests that it is not an either-or situation, but that strategies such as the radical and incremental can successfully be combined in practice. The adoption at SoftPharm had definite radical aspects during the Method Project, while the Development Project and the Extension Project were approached incrementally.

McKenney and McFarlan (1990) suggest that adoption processes passes through different phases: initiating the adoption, experimenting with the innovation, controlling the way in which the innovation is used and transferring the innovation to other domains. Again, we can see the adoption process at SoftPharm following this rough pattern. The Method Project is part of the initiation and experimentation phase; the Development Project is part of the control phase; and the Extension Project is part of the transfer phase. The overall dynamics of networks and networking during the adoption of the software development method at SoftPharm is consistent with existing models of adoption of innovations. It does, however, focus on how knowledge is created, shared and integrated, and it provides valuable additional advice on how to bring software development methods into practice.

The other key aspect of our analysis is the different roles that software development methods can play as they are brought into practice. Our study shows three different roles captured through the metaphors 'method as solution', 'method as support' and 'method as repository'. There are other roles that methods can play and that are relevant for research and practice to understand (e.g., a rational role and a political role) (Fitzgerald, 1998a). In these terms, our study has focused on and elaborated the rational role by emphasising the different ways in which knowledge is created, shared and integrated to support software method adoption. Our study has not emphasised the political role of methods to the same extent. In addition, our study strongly confirms that methods are used in fragments, and it further shows how new fragments can be added as the method is brought into practice. The project participants in the Method Project selected specific method fragments to use at SoftPharm, and the project participants in the Extension Project added a new method fragment. The method fragments were chosen based on situational characteristics. This confirms that there are pragmatic reasons for not using methods in their entirety (Fitzgerald, 1997). The different purposes of using and adding fragments were driven by the desire to fit the emerging version of the method to the particular context at SoftPharm.

The three roles that software development methods can play when brought into practice are well aligned with Lundell and Ling's (Lings and Lundell, 2004; Lundell and Lings 2004) three stakeholder perspectives on software development methods: the systems developer perspective, the concept developer perspective and the product developer perspective. The systems developer perspective is the perspective of the users of a method. The concept developer perspective is the

perspective of the inventor and designer of a method. The product developer perspective is the perspective of the people who provide the artifacts and support needed to use the method. These three perspectives correspond quite closely to the three metaphors we have identified in SoftPharm's adoption of the software method. The Development Project and the support metaphor correspond to the systems developer perspective because this is the perspective of the user of the method. The Extension Project and the method as a repository metaphor correspond to the concept developer perspective. Finally, the Method Project and the method as a solution correspond to the product developer perspective because the main aim in the Method Project was to adopt a proper method.

Our findings have implications for both research and practice. The study suggests that knowledge management theory is highly relevant in helping us to better understand the challenges involved in bringing software development methods into practice. There are no simple and general knowledge management strategies available for that purpose. Each information technology organisation has to adopt a repertoire of strategies and tactics and to select and tailor these to the different needs and situations that emerge during the implementation and use of methods. The framework suggested by Swan et al. (1999) and similar approaches suggested by Hansen et al. (1999) and Tiwana (2002,p.294) are valuable frameworks for guiding further research into this area. They also serve as useful guidelines for tailoring knowledge management practices to the needs of contemporary information technology organisations (Mathiassen and Pourkomeylian, 2003).

## **6 Conclusion**

The analysis compares and contrasts three types of relationships between methods and practices. It relates them to existing knowledge about stages of technology adoption. It also discusses the implications for software practice and research. In particular, the analysis leads to complementary understanding of how organisations can effectively manage software method adoption to take advantage of the opportunities presented without falling into the well known traps of non-adoption, pseudo-adoption or literal-adoption.

Our analysis reveals three quite different types of relationships between method and practice. First, the method was seen as a solution to perceived problems in the existing software practices. Second, the method was seen as a support for planning and executing ongoing software projects. Third, the method was seen as a repository for inventing and codifying new knowledge based on software development experiences. The knowledge management emphasis on networks and networking, as suggested by Swan et al. (1999), consequently changed considerably over the different stages of bringing the method into practice.

## **Acknowledgements**

We would like to thank the numerous people at SoftPharm for their invaluable help. We would also like to thank Peter Carstensen for his involvement in the data collection. Some of the work has been conducted under the DIWA project funded by the Danish Research Councils.

## **References**

Attewell, P. (1992) 'Technology, diffusion and organizational learning: the case of business computing', *Organization Science*, Vol. 3, pp.1–19.

- Bansler, J.P. and Bødker, K. (1993) 'A reappraisal of structured analysis: design in an organization context', *ACM Transactions on Information Systems*, Vol. 11, pp.165–193.
- Bansler, J.P. and Havn, E.C. (2003) 'Improvisation in action: making sense of IS development in organizations', *Proceedings of Action in Language, Organizations and Information Systems (ALOIS)*, Linköping, Sweden.
- Baskerville, R. and Stage, J. (2001) 'Accommodating emergent work practices: ethnographic choice of methods fragments', *Presented at IFIP WG 8.2*, Boise.
- Baskerville, R.L. and Wood-Harper, T. (1998) 'Diversity in information systems action research methods', *European Journal of Information Systems*, Vol. 7, pp.90–107.
- Brinkkemper, S. (1996) 'Method engineering: engineering of information systems development methods and tools', *Information and Software Technology*, Vol. 38, pp.275–280.
- Chatzoglou, P.D. (1997) 'Use of methodologies: an empirical analysis of their impact on the economics of the development process', *European Journal of Information Systems*, Vol. 6, pp.256–270.
- Fichman, R. and Kemerer, C.F. (1997) 'The assimilation of software process innovations: an organizational learning perspective', *Management Science*, Vol. 43, pp.1345–1363.
- Fitzgerald, B. (1996) 'Formalized systems development methodologies: a critical perspective', *Information Systems Journal*, Vol. 6, pp.3–23.
- Fitzgerald, B. (1997) 'The use of systems development methodologies in practice: a field study', *Information Systems Journal*, Vol. 7, pp.201–212.
- Fitzgerald, B. (1998a) 'A tale of two roles: the use of systems development methodologies in practice', in N. Jayaratna, B. Fitzgerald, A.T. Wood-Harper and J. Larrasquet (Eds.) *Educating Methodology Practitioners and Researchers*, UK: Springer-Verlag, pp.89–96.
- Fitzgerald, B. (1998b) 'An empirical investigation into the adoption of systems development methodologies', *Information and Management*, Vol. 34, pp.317–328.
- Fitzgerald, B. (2000) 'Systems development methodologies: the problems of tenses', *Information Technology and People*, Vol. 13, pp.174–185.
- Hansen, M.T., Nohria, N. and Tierney, T. (1999) 'What's your strategy for managing knowledge?', *Harvard Business Review*, March–April, pp.106–116.
- Hidding, G. (1997) 'Reinventing methodology: who reads it and why?', *Communications of the ACM*, Vol. 40, pp.102–109.
- Kozar, K.A. (1989) 'Adopting systems development methods. An exploratory study', *Journal of Management Information Systems*, Vol. 5, pp.73–86.
- Leonard-Barton, D. (1987) 'Implementing structured software methodologies: a case of innovation in process technology', *Interfaces*, Vol. 17, pp.6–17.
- Lings, B. and Lundell, B. (2004) 'Method-in-action and method-in-tool: some implications for CASE', *Proceedings of ICEIS 2004 – Sixth International Conference on Enterprise Information Systems*.
- Lundell, B. and Lings, B. (2004) 'Method in action and method in tool: a stakeholders perspective', *Journal of Information Technology*, accepted for publication.

- Lyytinen, K. and Rose, G. (2003) 'Disruptive information system innovation: the case of internet computing', *Information Systems Journal*, Vol. 13, pp.301–330.
- Mathiassen, L. (2002) 'Collaborative practice research', *Information Technology and People*, Vol. 15, p.321.
- Mathiassen, L. and Munk-Madsen, A. (1986) 'Formalizations in systems development', *Behaviour and Information Technology*, Vol. 5.
- Mathiassen, L. and Pourkomeylian, P. (2003) 'Managing knowledge in a software organization', *Journal of Knowledge Management*, Vol. 7.
- Mathiassen, L. and Sørensen, C. (1996) 'The capability maturity model and CASE', *Information Systems Journal*, Vol. 6.
- Mathiassen, L. and Sørensen, C. (1997) 'A guide to manage new software engineering tools', in T. McMaster, E. Mumford, E.B. Swanson, B. Warboys and D. Wastell (Eds.) *Facilitating Technology Transfer through Partnership: Learning from Practice and Research*, London: Chapman and Hall, pp.257–272.
- McKenney, J.L. and McFarlan, F.W. (1990) 'The information archipelago-maps and bridges', in T.D. et al. (Eds.) *Software State-of-the-Art: Selected Papers*, Dorset House Publishing, pp.99–116.
- Nuseibeh, B., Finkelstein, A. and Kramer, J. (1996) 'Method engineering for multi-perspective software engineering', *Information and Software Technology*, Vol. 38, pp.267–274.
- Orlikowski, W.J. (1993) 'CASE tools as organizational change: investigating incremental and radical changes in systems development', *MIS Quarterly*, Vol. 17, pp.309–340.
- Premkumar, G. and Potter, M. (1995) 'Adoption of Computer Aided Software Engineering (CASE) technology: an innovation adoption perspective', *ACM SIGMIS Database*, Vol. 26, pp.105–124.
- Rogers, E.M. (1983) *Diffusion of Innovations*, 3rd edition, New York: The Free Press.
- Sharmaa, S. and Rai, A. (2003) 'An assessment of the relationship between ISD leadership characteristics and IS innovation adoption in organizations', *Information and Management*, Vol. 40, pp.391–401.
- Sherif, K. and Vinze, A. (1999) 'A qualitative model for barriers to software reuse adoption', *Presented at the 20th International Conference on Information Systems*, Charlotte, North Carolina, USA.
- van Slooten, K. (1996) 'Situated method engineering', *Information Resources Management Journal*, Vol. 9, pp.24–31.
- Swan, J., Newell, S., Scarbrough, H. and Hislop, D. (1999) 'Knowledge management and innovation: networks and networking', *Journal of Knowledge Management*, Vol. 3, pp.262–275.
- Tiwana, A. (2002) *The Knowledge Management Toolkit: Orchestrating IT, Strategy, and Knowledge Platforms*, 2nd edition, Upper Saddle River, NJ: Prentice Hall PTR.
- Truex, D.P. and Avison, D. (2003) 'Method engineering: reflections on the past and ways forward', *Presented at the Ninth Association for Information Systems Americas Conference on Information Systems (AMCIS 2003)*, Navigating the Torrents of Technology, Tampa, FL.
- Walsham, G. (1993) 'Interpretive case studies in IS research: nature and method', *European Journal of Information Systems*, Vol. 4, pp.74–81.

Welke, R. and Kumar, K. (1991) 'Method engineering: a proposal for situation-specific methodology construction', in W. Cotterman and J. Senn (Eds.) *Systems Analysis and Design: A Research Agenda*, New York: Wiley.

Yin, R.K. (1994) *Case Study Research: Design and Methods*, 2nd edition, Newbury Park: Sage Publications.

Zmud, R.W. (1982) 'Diffusion of modern software practices: influence of centralization and formalization', *Management Science*, Vol. 28, pp.1421–1431.

Zmud, R.W. (1984) 'An examination of "Push-Pull" theory applied to process innovation in knowledge work', *Management Science*, Vol. 30, pp.727–738.

## **Paper E**

### **Conceptualizing method utilization**

Vogelsang, L. and F. Kensing (2006) "Utilizing systems development methods - a conceptual framework", *Proceedings of the 14th European Conference on Information Systems, Göteborg, Sweden*.

# UTILIZING SYSTEMS DEVELOPMENT METHODS – A CONCEPTUAL FRAMEWORK

**L. Vogelsang and F. Kensing**

The IT University of Copenhagen, Denmark  
e-mail: carstensen@itu.dk, vogelsang@itu.dk

## Abstract

*There are not many frameworks of method utilization or use in organization based on empirical studies nor is there a common understanding of these issues. This paper investigates how systems development methods are utilized in practice and proposes a framework to conceive method utilization. The explanatory value of the framework is illustrated by providing an analysis of a case study. The framework highlights several issues of method utilization based on a three years long field study. The framework has two dimensions. The first dimension covers three levels in organizations at which methods can be utilized. They are the organizational level, the project level, and the individual level. The second dimension covers three aspect of utilization of the method that can take place at each level. The three aspects are adoption, adaptation, and use. Thereby the framework provides nine perspectives on method utilization that allow us to understand and guide method utilization in a broader sense than we have found in the literature, which primarily deals with a subset of the nine perspectives. Furthermore, the paper introduces a distinction between method use and method utilization to emphasize a broad view on method utilization. Method utilization includes adoption, adaptation, and use at different levels in development organizations. Method use is strictly defined as the use of methods for systems development.*

*Keywords: method utilization, systems development methods, conceptual framework.*

## 1. Introduction

This paper presents a framework that conceptualizes the work involved in bringing systems development methods to use. The framework is derived from a three years longitudinal field study that followed the introduction, use and extension of a method in a development organization. The framework covers the utilization of methods. We use the term *utilization* rather than *use* to emphasize that activities, besides the strict use of method for development, takes place when a method is taken up by an organization. We found that three aspects of method utilization take place: adoption, adaptation and use. These three aspects are one of two dimensions in the framework. The other dimension is the three levels, organizational, project and individual, in the development organization at which the utilization takes place.

The framework illustrates the scope that has to be dealt with in order to utilize a method. On top of the method utilization that takes place in systems development, the methods are also utilized for bringing the method into organizations. Thereby, the scope of method utilization becomes beyond the methods' primary focus on systems development. The effort that goes into utilizing a method in a development organization can take place at different levels in the organization and in order to adopt, adapt, and learn about the method. The framework provides a foundation that enables

analyses of method utilization and for guiding the introduction of methods into IT organizations. The next section describes our research approach in the field study.

## 2. Research approach

The framework presented in this paper is based on an interpretive longitudinal field study (Walsham 1993, Yin 1994). Our interpretation of the utilization of the systems development method is expressed in the framework presented in this paper. The main principle in process of constructing the framework was to iterate between the observations from the field study and the emergent framework as it is described in the fundamental principle in the hermeneutic circle (Klein and Myers 1999). Furthermore, the framework draws on the research literature related to method utilization. The framework was revised several times during the analysis to ensure that the framework (the whole) reflected the observations (the parts). The framework is one interpretation of the field study constructed to understand how methods are utilized in practice. We recognize that there are aspects of method utilization that the framework does not cover, for instance the methods political roles (Fitzgerald 1998), which we have not focused on these aspects. Our preconception (Klein and Myers 1999) of method utilization was that methods are primarily intended to be utilized for systems development although we recognize that some of our observations illustrate other purposes than systems development, for instance gaining recognition in an organization.

The longitudinal approach enabled us to get an in-depth understanding of method utilization, which has helped to broaden our view beyond what is traditionally called method use. The study was undertaken as a practice study (Mathiassen 2002) in order to explore and understand how methods are utilized in practice. We do recognize that the study is limited to a single organization and a single method and therefore is exploratory in nature and lack generalization. However, the study provides rich details on method utilized, which enables us to shed light on new aspects of method utilization. Furthermore, the study is the primary foundation for the conceptual framework.

The study took place over a 3 years period and followed 4 projects in an IT department. The main focus in the study was on the utilization of a method in the projects. We had the opportunity to follow how a method (Rational Unified Process) was introduced in the IT department, the initial use of the method, a more mature use of the method, and some of the activities involved in adopting and adapting the method. The following describes the 4 projects we followed, how and why the methods were utilized, and how data was collected in each of the projects.

The first project, the *Method Project*, was initiated to find and introduce a new method in the IT department. The main arguments for introducing a new method were the use of new technology (object oriented), new ways of developing IT (web development), and that the existing method was an old waterfall method and not a modern and iterative method. Our focus in the project was on 1) how the method was utilized in the process of bringing the method into use in the IT department and 2) the main purposes for adopting a method. The data was collected through interviews of the project participants, observation of all project meetings, and by analysing key documents.

The second project, the *Try-out Project*, was a development project where parts of the new method were tried out in a real development context. The main purpose of the project was to develop a web site. Therefore, the method utilization was secondary to the primary focus in the project on developing the web site. Our focus in the project was on 1) how the method was perceived in the project and 2) how the method was actually utilized in the project. We interviewed key project participants at the outset of the project and after the project finished. We had the option to observe a few project meeting, but not the daily work routines because the company lacked office space at the

time. We also collected documents from the different phases in the project especially those related to the method use. Together the Method Project and the Try-out Project lead to the adoption of a new method (Rational Unified Process (Kruchten 2000)) in the IT department.

The third project, the *Use Project*, was a development project of a web application used to collect patient data among doctors. The project took place 1 1/2 years after the new method was adopted in the IT department. During the 1 1/2 years people in the IT department had attended courses and started using the method on a regular basis in development projects. Furthermore, a significant amount of work went into adapting the method to the IT department context. Our focus was on how the method was utilized in relation to the development project. We observed meetings and office work where the method was utilized. Furthermore, we interviewed the project participants about the method and analysed documents from the project and the method.

The fourth project, the *Extension Project*, was initiated to synthesize work practices into a technique used to mock-up user interfaces. The introduced method did not have a technique for that, so the idea was to create a technique for mocking up user interfaces and add it to the method. The project consisted of 3 people who worked with the task of creating mock-ups of user interfaces and one of the authors of this paper. The author's role was to cooperate with the other three project participants on creating the technique. In other words, the author's role was collaborative participation (Baskerville and Wood-Harper 1998). The main purpose of having a researcher being part of the project was to communicate some of the problems with mock-up user interfaces that we observed in the Use Project. The construction of the standard technique was primarily done by the project participants from the IT department.

### **3. A framework for Method Utilization**

The purpose of this chapter is to present a framework for analyzing the utilization of systems development methods in practice. The reason for creating the framework presented in this paper is to provide a structure that can be used for researchers and practitioners to analyze and guide method utilization.

The framework consists of two dimensions: levels of utilization and aspect of method utilization. These two dimensions are described in detail in the first two parts of this section. The first part describes the levels of utilization. The framework consists of three utilization levels and at each level the other dimension, the aspects of method utilization, can be found. The levels of utilization are used to categorize things such as activities, aspects of use, and deliveries that are part of systems development and related to method utilization, into different levels in the development organization. The levels provide categories that help us understand and guide certain aspects of method utilization.

The framework does not cover the method industry where the development and dissemination of standard "off-the-shelf" methods to the software industry takes place. The development and dissemination of methods is usually undertaken by companies such as Rational (IBM) and Microsoft or by academia. The methods are disseminated as text books, web sites, software and through teaching and courses held by the private organizations or in an academic context. It is from the method industry that development organizations obtain standard methods. The method industry is not included in the framework because we do not have data on it. We recognize that it could be a level to add to the framework, because it has a significant influence on how and why methods are used in practice. We think it is reasonable to assume that the three aspects of utilization (adoption, adaptation and use) takes place in the method industry, which future research might show.

### **3.1. Levels of method utilization**

The framework deals with utilization at three levels in development organizations, which are: The individual level, the project level, and the organizational level. The levels are inspired by the levels found in Fitzgerald, Russo, and O'Kane (2003) and Fitzgerald, Russo, and Stolterman (2002) (see section 5 for more details). An activity, product, or aspects of method utilization belongs to a level of utilization if its current way of existence depends on the level. If a project is stopped (for whatever reason) then some activities and products become pointless to continue (e.g. project meeting and project plans). Project meetings and project plans are categorized into the project level of utilization, because their existence is dependent on that level of utilization. However, other products and activities might survive the termination of the project at other levels of utilization, for instance standards for coding, tasks, approaches, etc.

#### **3.1.1. Utilization at the organizational level**

The organizational level of utilization is where we find activities and products that are intended to affect an entire development organization or development department. At this level, adoption and adaptation of a method are intended to have an organization or department wide effect. It is at the organizational level that we find the process or method department and the organization's methods.

A reason for dealing with methods at the organizational level is reuse of knowledge, transfer of specific knowledge among projects, and to have a standard approach for systems development. By working with methods at the organizational level, the organization can achieve a common language among project participants, a division of work, standard procedures, etc., which can be reused in the different projects without starting from scratch each time. Another reason is transfer of knowledge through codification of knowledge for reuse in other or future projects. A third reason is to enforce standards on the systems development projects to meet certain industrial standards, such as ISO 9000.

The activities at the organizational level are for instance to bring a method into the organization, adapt the method to the organizational context, and maintain the method. The products we find at the organizational level are a standard or a customized method used in the organization. It is also at the organizational level that 'tools' for managing development organizational wide such as ISO 9000 and CMM are enforced on the organization. Basically, the organizational level deals with method related issues that are for the entire organization.

#### **3.1.2. Utilization at the project level**

The second level of utilization is the project level. It is at this level we find the development projects, where the systems development takes place. It is at this level that some of the central method fragments are intended to be utilized. For instance, process models such as the waterfall model and the spiral model which are intended to guide the development process, diagrams techniques, such as Use Cases, flow diagrams, and E/R diagrams, which are intended to support the work on describing and developing the system under development.

Most information systems are so complex to develop, technically and socially, that it takes several people to develop the system. The most common way of organizing systems development is to establish a project. Projects are characterized by taking place within a specific timeframe and having a specific purpose. Development projects can involve project participants with different qualifications and roles in the development projects. Methods often provide descriptions of qualifications and roles and guidelines for organizing them in the development projects. Organizing

systems development into projects is one way of dealing with the complexity in systems development. The inclusion of the project level of utilization emphasizes that methods can have significant influence on the development projects and therefore is an important aspect of their utilization.

The activities at the project level is for instance the management and organization of the development process for the development of specific system, the coordination of activities carried out by the project participants, project meetings, etc. The products produced at the project level are, besides the information system, artefacts related to the development process, for instance, Use Cases, flow diagrams, E/R diagrams, project plans, etc. Some of the products become part of the information system and others are means to develop the information system and become obsolete when the development ends.

### **3.1.3. Utilization at the individual level**

The third level of utilization is the individual level of utilization. The individual level of utilization is where a project participant utilizes the method or its parts in order to develop systems. The individual level of utilization covers the part of method utilization that are not reflected directly in the development organization or the development projects. An example is when a project participant decides to use a method or a method fragment without disseminating the use to a development project or the development organization. The individual level of utilization also covers that methods are not necessarily used by project participants as intended in the development project, the development organization or by the method. For instance, a project participant may have to change part of a method to fit the specific context in which he or she works in order to make the method fit their work. Furthermore, project participants can face problems that the method does not cover. In that case, the project participants might have to deviate from the method and find their own way of doing the job.

Some of the method utilization performed by individuals is required by either the project or the organization. This type is not considered to be method utilization at the individual level because it originates from another level of utilization. Instead, method utilization at the individual level covers that a developer adopt a method fragment to ease and conduct work. For instance, developers program, create and use Use Cases, E/R diagrams etc. to do their work. These activities and products often also play a part at the project level. The individual level also deals with the non-use of methods, i.e. that developers sometimes don't use methods as intended from the other three levels.

## **3.2. Aspects of method utilization**

The framework consists of three aspects of method utilization in practice. We use the term aspect to focus on a specific perspective or view on method utilization. The aspects enable us to focus on a certain part of method utilization. The aspects of method utilization are: adoption, adaptation, and use. Method adoption is the decision to use a method in a certain context. Method adaptation is the deliberate or non-deliberate change of a method. Method use is the enactment of a method. The following three sections go into details on each of the three aspects.

### **3.2.1. Method adoption**

An adoption of a method is the decision to bring a specific method or a specific method fragment into use in a specific context. The context can be an organization, a project, or an individual. The

adoption of a method can include examination of methods and experimentation undertaken at one of the three levels. The examination and experimentation with a method leads to either adoption or rejection of the method. This intention can be different at the three levels of utilisation and across organizations, projects and individuals. More concrete, the claimed value of utilizing methods can be different at each level. For instance, the intend behind adopting a method at the organizational level (e.g. getting a certification) might not be the same as the intention is behind adopting the methods in the project (e.g. communication among project participants). The same applies to projects and individuals.

The organizational level carries an intention through the method utilization organizational wide. For instance by enforcing standards to meet certain certification such as ISO9000, to bring conformity among the projects, and to communicate to customers that development is done in a certain way in the organization. The project level of utilization carries an intention through the method utilization in projects. For instance by bringing method fragments such as diagramming techniques and process models into use in the projects for a specific purpose. Finally, the individual level of utilization carries the intention that an individual has. Individuals enact their intention through their preferences and working styles. The result of the different views on a method at the three levels can be that a method or some of its fragments might only be adopted at some levels in the organization. This is e.g. the case if an organization adopt a method but the method is not adopted by the individuals or vice versa.

### **3.2.2. Method adaptation**

Method adaptation is the change of a method so it becomes suitable for a specific context. The adaptation can be undertaken to suit an organization, a project, or individuals. Furthermore, the adaptation can be undertaken by an organization, a project or individuals. Adaptation can take place at the organizational level to suit needs at the project level and adaptation can take place at the project level to suit needs at the individual level. Adaptation can also take place within each of the three levels, e.g. the method is adapted at the organizational level to suit needs at the organizational level.

This section introduces the notion of adaptation style. An adaptation style conceives in which way a method or a method fragment is adapted to a specific context. The purpose of introducing the adaptation styles is to provide a vocabulary for describing ways in which adaptation takes place. Furthermore, the adaptation styles are provided to help us understand the adaptations and, perhaps in the future, help us understand what adaptation styles that can lead to successful adaptations. An adaptation style is defined by: 1) whether the adaptation is planned or unplanned and 2) the type of adaptation i.e. does the adaptation exclude, change, extend, or take the method as it is.

Method adaptation can be deliberate, i.e. planned, or something that 'just happens', i.e. unplanned. Planned adaptation can take place for instance if changes to the method is prescribed by the method (e.g. consider whether or not to use a technique or a documents). Unplanned adaptation can take place when a situation is not covered by a method or when the method does not sufficiently fit the situation and is unintentionally adapted to fit the situation. In these cases the project participants have to adjust or ignore the method to be able to do their work.

The adaptation styles consist of four types of adaptation of the methods; excluding, changing, extending, and literal adaptation. The excluding type of adaptation means that the method or a fragment of the method is ignored and not used, which might be either planned or unplanned. The changing type of adaptation happens when a part of or the method is changed to fit a specific situation, but the method is still intended to be used. The original method fragment or method in

this case do not fit the context and has to be changed to be used in the context. The change of the method can be planned, for instance by deliberately adapting the method to a specific organization or project. The change of the method can also take place in an unplanned fashion when methods are changed to suit a situation without any prior planning. The extending type of adaptation covers the cases where activities or product, needed in a development project, are not covered by the method and thus has to be added. This may take place either in an ad hoc fashion or be based on experiences from former and similar situations. Finally, the literal type of adaptation, is when the method is adopted "as-is", i.e. without changing the method. This can happen if it is assumed that the method works out of the box. Table 1 describes the eight adaptation styles.

Adaptation Styles	Excluding	Changing	Extending	Literal
Planned adaptation	Excluding a method fragment from the method.	Planned change of a method fragment.	Deliberately borrow method fragments from other methods.	To use the method as it is.
Unplanned adaptation	The method or method fragment never comes into use.	A new method emerges from the original method, because the method did not suit the situation.	Unplanned adoption of method fragments from other methods.	The provided methods are used as-is.

*Table 1: Method adaptation styles*

### **3.2.3. Method use**

Method use is when a method is brought into use to accomplish a task in systems development. The task the method is used for can take place at each of the three levels of utilization. Therefore the task the method is used for can be quite different in nature. This is the most common use of methods. However, methods are also sometimes utilized in the process of adopting and adapting the method at the three levels. The method can be used to provide information about its potential use in systems development. Thereby the method is used for assessing the method's potential, which can lead to either adoption or rejection of the method. Some method (e.g. Rational Unified process (Kruchten, 2000)) and approaches (e.g. Method Engineering (Brinkkemper 1996; Welke and Kumar 1991)) includes ways to adapt the methods themselves. By using these approaches the methods are in use in order to adapt the methods. The methods provide a meta-method fragment that supports processes for adapting methods and bringing them into use. The main point in both assessments of methods (adoption) and meta-method fragments for adaptation is that the method can play an important part and is brought into use to enable actors to do certain tasks, which are not in a strict sense systems development. These tasks are related to systems development and important to make a method valuable in systems development. The tasks are in some sense meta-tasks performed to bring the methods to use for systems development in the IT development organizations. Our understanding of "method use" thus is broader than the common "method use" term that almost entirely is focused on methods used for systems development. Therefore we have chosen to use the term method utilization to emphasize this 'broader' view on methods.

## 4. Examples from the longitudinal field study

This section provides examples from the longitudinal field study that illustrate the explanatory value of the framework.

### 4.1. Organizational level of method use

We observed two examples of method *adoptions* at the organizational level of utilization in the field study: In the Method Project where a new method was brought into the organization and in the Extension Project where the method was extended by synthesizing a current work practice. The main intention of method adoption were in both cases to standardize parts of the development process and deliveries to make the projects in the organization more conform and reduce ad hoc activities. The targeted areas were perceived as too ad hoc, which lead to problems with taking over other peoples work (dealt with in the Extension Project) and developing an understanding of where the project was heading (dealt with in the Method Project). The Method Project was initiated to make a more conform development process and was an attempt to reduce the amount of ad hoc activities and development. The main mean to this was to introduce a new method in the organization. The Extension Project had a similar goal although it was targeted towards very specific activities and delivery. Furthermore, the means to make activities and products conform in this case was to synthesize their current work practice into a new method fragment instead of adopting a new method fragment from a standard method.

The main *adaptation* at the organizational level took place in the Method Project. It was quite quickly acknowledged by the project participants that the standard method would not fit the organization. The reasons for this were that there were certain differences in use of terms and standard to be met in the organization because it was making software for the medical industry. Furthermore, the project participants realized that several work practices had to be changed and some people in the organization would resist some of these changes. These problems were to some extent solved by adapting the method. The need for adaptation was perceived so excessive that a permanent group of people were established to adapt and maintain the method. Two years after the adoption of the method at the organizational level only half of the method was actually brought into use in the projects at the project level. This was due to an extensive work with adapting the method and changing work practices.

The main goal in the Extension Project was to synthesize a part of their current work practice into a method fragment and add it to the method adopted in the Method Project. The project participants in the Extension Project did not bring a method fragment that could support the work practice. Therefore no *adaptation* of an existing method happened in the Extension Project. However, a number of possibilities for adaptation of the created method fragment were given to the ones that were going to use it. These possibilities for adaptation were primarily introduced in areas where the development projects in the organization was perceived as being too diverse to synthesize or because a standard was too difficult to explicate or the available tools could not support the work.

The method was in use as guidelines in the Method Project and in the Extension Projects. Main techniques from the method were tried out in the Method Project. Furthermore, Use Cases and iterations were tried out in the Try-out Project to test the method in practice. The initiation of these activities were inspired by guidelines from the method on adopting the method in an organization. The method was used to support the process of adopting and adapting the method to the organization. The project participants in the Extension Project also used guidelines from the method to synthesize a new method fragment. The method prescribed a set of characteristic and

requirements for the created method fragment. An example of a guideline is that every process must result in a product. Requirements like this guided the construction of the method fragment throughout the Extension Project. In general, the method was used at the organizational level as an infrastructure for creating and bringing the method into the organization.

## 4.2. Project level of method utilization

The *adoption* observed in the field study at the project level took place in the Try-out Project and the Use project. As mentioned above, two key method fragments, Use Cases and iterations, were adopted in the Try-out Project in order to try out vital parts of the method in practice. There were no attempts to *adapt* the Use Cases and iterations to the project. There was no formal adaptation of the method fragments in the Try-out Project and therefore name conventions and techniques were taken “as-is”. It is important to notice that Use Cases were never used in the entire project, only by a few programmers. Furthermore, iterations never played a significant role in the Try-out Project due to delays and lack of motivation for having the iterations.

The *adoption* in the In-Use project was more or less given because the method was adopted in the organization to such an extent that the method was expected to be used in development projects. The *adaptation* in the project was about which method fragment that should be used. The adaptation of the method was done in a formal way, by having the project manager, the lead developer and a process engineer to select which fragments to include and which to exclude. Some method fragments were excluded because the activities described were not taking place (e.g. assessment of server capacity on local servers because the application was placed on other servers) and other activities were included in the project although they were not formally a part of the method (e.g. information architecture). The role of the method in the Use Project was primarily to guide the development process and to provide guidelines for deliveries in the project. The prescriptions of activities and products in the method were perceived as guidelines that had to be followed unless there were reasons for doing something else. The actual use of the method was very complex because the method influenced many aspects of the development. We did observe that most of the terminology and some of the techniques from the method was in use in the entire project. The project used iterations although they were in reality more like status meetings than a point in time where the deliveries were tested to a standard as the method prescribed. The project manager was aware of this and it was perceived as a practical way to use iterations.

## 4.3. Individual level of method utilization

In the field study we observed several *adoptions* of method fragments at the individual level. We also observed a change in the way the adopted method was perceived in the organization. In the beginning, where the organizational adoption still hadn't taken place at the organizational level (i.e. in the Method Project and Try-out Projects), the method did not play an important role neither in the projects nor for the individuals. As the method was adapted at the organizational level to fit the organization and as people got more experience with the method, it also became more important for the individuals in their daily work. The method became part of the daily work practice and means to get the job done for each individual. In the Try-out Project, the method fragments adopted at the project level were rejected by a major part of the project participants because they did not understand the purpose and what to do with the method fragments. This was radically changed in the Use Project. The project participants understood the purpose and relevance of the method and enough about the method fragments to perceive them as beneficial. Therefore the project participants adopted the method and relevant method fragments. Besides a better understanding of

the method and its fragments, the project participants also got a better understanding of how to adapt the method. As described above, the project participants understood the method as relevant guidelines and were able to adapt the method to their situation, for instance by skipping a certain activity or by changing a technique or product.

## **5. Related work**

The framework proposed in this paper illustrates various ways in which a method can be utilized in an IT development organization. By interpreting the field study in terms of the framework we have shown how the framework can be used to analyse situations in which methods are utilized for different purposes. The following two sections relate the framework to research on methods in practice. The first section relates the framework to two of the few other frameworks on method use that are based on empirical research. The second section interprets some examples of empirical studies of method utilization in terms of our framework.

### **5.1. Method frameworks**

Andersen et. al. (1990) provides one of the earlier frameworks on method use. The framework was created as part of the MARS research project and is based on empirical work. Andersen et. al. makes a distinction between abstract and concrete concepts for describing systems development. For abstract descriptions of systems development they suggest functions (e.g. analysis, design, planning, etc) for describing intension behind actual processes or actions. Likewise they introduce terms like methods, tools, and techniques to talk about how actual processes and actions ought to be carried out. They suggest processes as the main concept for describing what actually happens at the concrete level. In the vocabulary of the framework presented in this paper, Andersen et al deal with method use at all three levels of method utilization, however they only briefly discuss adoption at the three levels and adaptation are not dealt with at any of the levels.

Fitzgerald et. al. (2002) provides a framework on method use in which they distinguish between the formal method and the method-in-action. They draw on Argyris and Schön's (1974) distinction between espoused theory and theory in use. The formal method in their framework corresponds to the espoused theory and the method-in-action to the theory in use. The formal method is the method as it is described in books and manuals. The method-in-action is the enacted method, i.e. how the method is actually used in practice, which might not be the same as the method prescribes. According to Fitzgerald et. al. a method has a role (political or rational (Fitzgerald 1998) that shapes the formal method and influences the method-in-action. Fitzgerald et al (2003) report from a case study of method tailoring where the method's role was rational. They investigated method tailoring at an industrial-, organizational-, and project level. The company adopted method fragments from the method industry (industrial level), tailored them to different organizational divisions (organizational level), and tailored the divisional methods to each development project (project level).

The main difference is that our framework emphasises the activities involved in utilizing a method. It is process centric. The framework by Fitzgerald et. al. (2002) is focused on where the method comes from, where it is changed and used. Their framework is method centric. This is reflected in the way in which the levels are used. Further, our framework has Fitzgerald et al's industrial and organizational activities at the organizational level, because the adoption of method takes place in the organizations. Fitzgerald et al perceives the method fragments as coming from the industry and therefore put them at the industrial level. We could have included an industrial level, but have left it

out because we haven't investigated, in this study, how adoption, adaptation and use take place in the method industry. Instead we did observe adoption, adaptation, and use by individuals and that is the reason for introducing the individual level. The two frameworks have different focuses, but are not contra dictionary. They are complementary in the sense that they provide two different approaches to understand methods in practice. The framework by Fitzgerald et. al. provides an understanding of the enactment of methods and our framework provides an understanding of the complex activities involved in adopting and adapting methods in development organizations.

## **5.2. Methods in practice**

Several studies have been conducted on method use in practice although we still have a lot to learn about method used in practice (Wynecoop and Russo 1993). There is a long tradition for investigating method utilization based on quantitative studies. The findings from these studies are often on the adoption rate of methods (e.g. Fitzgerald 1998b) and factors influencing use (e.g. Premkumar and Potter 1995). These studies provide snap shots of method use, but do not an understanding on how and why a method is utilized in a specific context (Wynecoop and Russo 1995). We are interested in how and why or why not methods work in practice. Therefore the research we examine to compare our framework is qualitative in nature.

Bansler and Bødker (1993) set out to explore how structured analysis is used in practice. They investigate the reasons for adopting structured analysis and how it is applied in three organizations. They go into details on which parts of structured analysis that are adopted at the organization, project and individual level. They have a few details on the adoption at the organizational level and most details at the individual level. They do not investigate adaptation of structured analysis, besides that they find that the method was used among other techniques. Adapting the method as such is not an issue they bring up in their paper. They go in to details on how the method is applied by developers to show that structured analysis is not applied as described by the method. The main point in the paper is that there is a gap between the method and its use. They suggest investigating practice to understand systems development which should enable us to create methods that are more suited for actual systems development.

In terms of our framework, Bansler and Bødker (ibid.) are interested in adoption at all three levels with an emphasis on the individual level. They are not investigating adaptation of the method at any level and investigate the use of the method primarily at the individual level. Assuming that adaptation is not described by Bansler and Bødker (ibid.) because it did not take place, our framework would have guided the focus in the organization towards adaptation of the method to make it more suitable for the organization, the project and individuals. This is in line with the suggestion of adapting methods to suit systems development practice given by Bansler and Bødker (ibid). They suggest changing the standard methods so they suit the development practice. Our framework points at the possibility to adapt the method within the organization.

Stolerman (1992) investigates how designers think about methods. He is interested in understanding how methods can support systems development and convinces developers to use them. He interviewed 20 developers about the utilization of methods by asking about why methods were adopted and how they could be used in systems development. Stolerman's study is about the individual level of method utilization and with a focus method adoption and improving the chance of method adoption by creating methods that suits systems development better.

## 6. Conclusion

The paper introduces a framework that opens up “method use” as it is usually treated in the literature and by practitioners. The framework consists of the three aspects of method utilization (adaptation, adoption, and use) and three levels of method utilization (organizational, project, and individual). The three aspects can be found at each of the three levels. Together the aspects and levels provide nine perspectives on method utilization. The nine perspectives can be used to analyse and guide method utilization. The perspectives from the framework can guide an analysis of method utilization by providing a terminology to base the analysis upon. The perspectives can also guide method utilization by providing an understanding of the activities it take in a development organization to utilize a method. However, we do recognize the guidance is at an abstract level and does not provide very specific guidelines for actual actions to be taking. Its purpose is to provide areas to take into consideration when utilizing methods.

The framework is based on a longitudinal field study and illustrates the diversity of method utilization in practice. Our field study showed that methods had an important role in other contexts than systems development. The paper introduces a distinction between method use and method utilization to emphasise that methods are brought in as a resource for other purposes than systems development. Furthermore, the paper discusses to what extent the literature related to method utilization captures the diversity in method utilization. The paper shows that other field studies on method utilization only capture a few of the nine perspectives that our framework provides.

## References

- Argyris, C., and Schön, D. (1974). *Theory in Practice: Increasing Professional Effectiveness*. Jossey-Bass, USA.
- Andersen N. E., F. Kensing, M. Lassen, J. Lundin, L. Mathiassen A. Munk-Madsen and P. Sørgaard (1990). *Professional Systems Development-Experiences, Ideas, and Action*. Prentice-Hall.
- Bansler, J. P. and K. Bødker (1993). A Reappraisal of Structured Analysis: Design in an Organization Context. *ACM Transactions on Information Systems*, vol. 11, pp. 165-193.
- Baskerville R. L. and T. Wood-Harper (1998). Diversity in information systems action research methods. *European Journal of Information Systems*, vol. 7, 90-107.
- Brinkkemper S. (1996). Method engineering: engineering of information systems development methods and tools. *Information and Software Technology* (38), 275-280.
- Fitzgerald, B. (1998a). A tale of two roles: the use of systems development methodologies in practice. in N. Jayaratna, B. Fitzgerald, A. T. Wood-Harper and J. Larrasquet (eds.) *Educating Methodology Practitioners and Researchers*. UK: Springer-Verlag, 89-96.
- Fitzgerald, B. (1998b). An empirical investigation into the adoption of systems development methodologies. *Information and Management* (34), 317-328.
- Fitzgerald, B., Russo, N., and O'Kane, T. (2003). Software development method tailoring at Motorola. *Communications of the ACM* 46 (4), April, 64-70.
- Fitzgerald, B., Russo, N.L., and Stolterman, E (2002). *Information systems development: methods in action*. McGraw-Hill, London.
- Klein, H.K. and Myers, M.D. (1999). A Set of Principles for Conducting and Evaluating Interpretive Field Studies in Information Systems. *MIS Quarterly* 23 (1), 67-93.
- Kruchten, P. (2000). *The Rational Unified Process An Introduction*. Second Edition. Addison-Wesley, Reading.

- Mathiassen, L. (2002). Collaborative practice research. *Information Technology & People* (15). 321-342.
- Premkumar, G., and Potter, M. (1995). Adoption of computer aided software engineering (CASE) technology: an innovation adoption perspective. *ACM SIGMIS Database* 26 (2-3). May/Aug, 105-124.
- Stolterman, E. (1992). How system designers think about design and methods: some reflections based on an interview study. *Scandinavian Journal of Information Systems* 4, 137-150.
- Walsham, G. (1993). Interpretive case studies in IS research: nature and method. *European Journal of Information Systems* 4, 74-81.
- Welke R. and K. Kumar (1991). Method Engineering: A Proposal for Situation-Specific Methodology Construction. in *Systems Analysis and Design: A Research Agenda*, Cotterman and Senn, Eds. New York: Wiley.
- Wynekoop, J.L., and Russo, N.L. (1993). Systems development methodologies: unanswered questions and the research practice gap. the 14th International Conference on Information Systems. ACM, New York, 181-190.
- Wynekoop J. and Russo N.L. (1995), *Systems Development Methodologies: Unanswered Questions*. *Journal of Information Technology* 10, 65-73.
- Yin R. K. (1994). *Case Study Research: Design and Methods*. 2nd ed. Newbury Park: Sage Publications.