Communications of the Association for Information Systems

Volume 23 Article 27

11-2008

Managing Uncertainty in Organic Development Projects

Lars Mathiassen
Georgia State University, lmathiassen@gsu.edu

Keld Pedersen
Aalborg University, Denmark

Communications of the Association for Information Systems

CAIS

Managing Uncertainty in Organic Development Projects

Lars Mathiassen

Centre for Process Innovation,

Georgia State University

lmathiassen@gsu.edu

Keld Pedersen

Computer Science,

Aalborg University, Denmark

Abstract:

A variety of organic models for systems development have been recommended for more than three decades. These models rest on the assumption that the uncertainty is high and additional team capabilities have to be developed during the project life-cycle. In contrast to the single-pass and document-driven waterfall model, organic models impose less rigid structure on the process, and they are geared toward exploration. We know little, however, about how uncertainties are managed over the life-cycle of organic systems development projects.

In response to this challenge, we adapt task uncertainty theory to conduct a qualitative study of management practices in a project based on two-phase funding, staged delivery, and a combination of prototyping and specifications. We provide detailed narratives of how uncertainties emerged, interacted, and were addressed in the project. The subsequent analyses suggest that the adopted organic model facilitated management of uncertainty, but it also introduced surprising and demanding management challenges that were not accounted for in the model.

The study adds to our understanding of management practices in organic systems development. It shows how combinations of offensive and defensive responses can help managers address the uncertainties they face. In addition, managers are advised to differentiate between developing know-what and know-how capabilities and to dynamically adapt their uncertainty response mode to fit a project's evolving context.

Keywords: Organic systems development, uncertainty, project management

Volume 23, Article 27, pp. 483-500, November 2008

I. INTRODUCTION

The waterfall model [Royce 1970] offers detailed guidelines for how to structure the development process, distribute tasks, monitor progress, assure quality, and integrate solutions. While the waterfall model is a powerful sense-making device and provides easy-to-follow support for project management, it assumes the task is well structured and team capabilities fit the task [McFarlan 1981; Mathiassen and Stage 1992]. These assumptions are seldom met in practice because it is difficult to specify and stabilize requirements early, because developers might have insufficient experience with technology, and, because the complexity might exceed developer experiences [McFarlan 1981].

Thought leaders of our profession have therefore for more than three decades promoted a variety of organic models based on iteration, prototyping, two-phase funding, staged delivery, and agile principles [e.g., Basili and Turner 1975; Boehm 1988; McConnell 1998; Cockburn and Highsmith 2001; Larman and Basili 2003]. These models assume the task uncertainty [Galbraith 1973] is high and additional team capabilities have to be developed during the project life-cycle [McFarlan 1981; Mathiassen and Stage 1992]. In contrast to the single-pass and document-driven waterfall model [Larman and Basili 2003], organic models impose less rigid structure on the process and they are geared towards exploration. As a consequence, projects are more unpredictable and require considerable project management attention. However, we know little about how to manage uncertainties in organic development projects.

The purpose of this research is therefore to investigate project management practices in organic systems development. To that end we adapt theories about task uncertainty [Galbraith 1973; Iivari 1992] to conduct a qualitative study of a project based on two-phase funding, staged delivery, and a combination of prototyping and specifications [McConnell 1998]. Inspired by the classical notion of task uncertainty - “the difference between the amount of information required to perform the task and the information already possessed” [Galbraith 1973, p. 5] - we present in-depth analyses of how uncertainties emerged, interacted, and were addressed. Our analyses suggest that the adopted organic model facilitated management of uncertainty, but it also introduced surprising and demanding management challenges that were not accounted for.

The study adds to our understanding of organic development projects and suggests tactics for their management. In the following, we present the background literature and describe our research approach. We then provide two detailed narratives of the investigated project. Subsequently, we adapt task uncertainty theory to make sense of the data. Finally, we discuss the contribution and implications for research and practice.

II. ORGANIC SYSTEMS DEVELOPMENT

A fundamental reason for adopting organic models in systems development is high task uncertainty [Galbraith 1973; Mathiassen and Stage 1992]. McFarlan [1981] distinguishes between three general types of systems development uncertainty, the first being a project’s conception of the problem and its solution. Requirements are often hard to define, communicate, and validate and once they are specified and agreed upon they often change [e.g. Naumann and Jenkins 1982; Gould and Lewis 1985; Curtis et al. 1988; Rising and Janoff 2000; Orr 2004; Sarkkinen and Karsten 2005; Turk et al. 2005]. As a consequence, most researchers promote prototyping, iterative models, or agile approaches to quickly deliver partial solutions to customers in order to support feed-back and progress [e.g. Naumann and Jenkins 1982; Beck 1999; Turk et al. 2005].

A second type of uncertainty relates to a project’s experience with technology; these increase as the project’s familiarity with the technical platform decreases [McFarlan 1981]. Brooks [1975] states that we must be prepared to throw away the first version and build and deliver a second whenever we use a new concept or rely on new technology. Organic models typically reduce these uncertainties by allowing early use [Parnas and Clements 1986] and by providing information about team productivity acquired in previous steps [Lott 1997].

A third type of uncertainty relates to a project’s complexity [McFarlan 1981]. Brooks [1987] argue that it is impossible to build complex systems faultlessly, and it is therefore advisable to iterate or adopt staged delivery. Sotirovski [2001] adds that if we are going to fail it is better to do it fast and in small scale. Parnas and Clements [1986] argue human errors are unavoidable and even when all information is available up-front, we cannot fully comprehend it. Generally, we can manage complexity by separating concerns and by using stepwise refinement [Wirth 1971].

To address these types of uncertainties, the literature discusses a variety of organic models. There is prototyping where the basic idea is to present a series of mock-ups or draft versions to users and improve the design based on their feed-back [Baskerville and Stage 1996]. There is structured use of documentation to gradually build and adjust specifications [Parnas and Clements 1986]. There is the spiral model that combines risk management with a cyclic approach to problem identification, experimentation, and documentation [Boehm 1988]. There is the combination of two-phase funding and staged-delivery that allow for contractual adjustments and stepwise delivery of a solution [McConnel 1998]. There is extreme programming that includes the use of on-site customers, minimal documentation, pair programming, and continuous integration of new components [Beck 1999]. There is the recent focus on agile methods [Cockburn and Highsmith 2001; Orr 2004; Baskerville and Pries-Heje 2004; Boehm and Turner 2004; Turk et al. 2005]. Finally, there is the inclusion of iteration as a key tactic in development methods [e.g. Jacobsen et al. 1999].

While organic models allow project teams to develop additional capabilities in response to high task uncertainty, the literature also suggests that they imply a number of challenges. First, managers must structure the process and combine use of prototypes and specifications [Boehm et al. 1984; Boehm 1988; Mathiassen and Stage 1992; Mathiassen et al. 1995; Sotirovski 2001]. Alavi and Wetherbe [1991] found that if prototyping was preceded by a systematic analysis, fewer iterations were needed and more efficient solutions were achieved, but at the cost of making the process more complicated and stressful. Second, organic projects must manage contracts and distribute economic risks in a way that is acceptable for both customers and vendors [Boehm 1988]. Third, they must facilitate exploration, but at the same time ensure convergence. Chillarege suggests that an agile approach that allows for fast feedback from the market is best in the early stages while predictability and control are more important in later stages [2002]. This is well in line with the spiral model [Boehm 1988]. Fourth, managers must design iterations to fit the particular project. Several authors suggest limiting the duration of iterations, except the first, to a few weeks [Beck 1999; Rising and Janoff 2000; Baskerville and Pries-Heje 2004] and also and placing the most valuable and important functionality in the first iteration. The rationale is to make sure that customers get the important requirements satisfied even if the project runs out of money or gets delayed [Beck 1999]. Sotirovski adds that systems mature over time, and better quality is achieved by building the most critical parts first [2001]. Organic projects must finally recurrently manage agreements between stakeholders [Sarkkinen and Karsten 2005], emerging interactions between developers and users [Gallivan and Keil 2003], make decisions between alternative options, and measure progress and nearness-to-completion [Baskerville and Stage 1996].

Key solutions to address these challenges include: cycles that combine prototypes and specifications [Boehm 1988], two-phase funding to allow for contractual flexibility [McConnel 1998], risk management [Boehm 1988; Mathiassen et al. 1995], faking of a rational design process [Parnas and Clements 1986], risk analysis to control prototyping [Baskerville and Stage 1996], staged delivery of solutions [McConnel 1998], and balancing agility with disciplined approaches [Boehm and Turner 2004].

Organic models are, in summary, motivated by different types of uncertainty and there is a portfolio of supporting techniques available that allow project teams to develop additional capability. While there are definite challenges involved in managing organic development projects, there are no studies of how uncertainties emerge and are addressed over the life-cycle of organic projects.

III. ORGANIZATIONAL TASK UNCERTAINTY

There is a long tradition for adopting information processing as an integrating concept in studying organizational practices [e.g., Burns and Stalker 1961; Tushman and Nadler 1978; Gresov 1989; Goodhue and Thompson 1995]. The basic approach is to study how information processing requirements can be linked to information processing options to obtain a satisfactory fit [Iivari 1992].

Galbraith's classical theory [1973] suggests that bureaucratic organizations, with hierarchical structures, centralized control, and standard operating procedures, in general are the most efficient in processing information. A bureaucratic organization relies, however, on the assumption that the task uncertainty is low so the information it needs to process outside its pre-programmed procedures is minimal. As the task uncertainty increases, the hierarchy becomes overloaded with ad-hoc information processing requests and the organization must be modified choosing from among four strategies [Galbraith 1973; Van de Ven and Drazin 1985]. Two of these, creation of slack resources and creation of self-contained units, reduce internal interdependencies and the need to process additional information, while two other strategies, investment in vertical information systems and creation of lateral relations, offer mechanisms to process more information. In the extreme case with high task uncertainty, organizations become organic with network structures, decentralized control, and ad-hoc mutual adjustment.

Galbraith's theory [1973] provides a useful foundation for studying uncertainty in systems development for a number of reasons. First, there is a strong analogy between Galbraith's understanding of organic organization as a response to increased uncertainty in bureaucratic organizations, and organic development as a response to the limitations of the waterfall model. Second, uncertainty has played and continues to play a key role in studying systems development [e.g. Alter and Ginzberg 1978; McFarlan 1981; Rai and Al-Hindi 2000; Turk et al. 2005]. Third, applying the task-uncertainty lens led us to practically relevant and theoretically interesting insights.

In the following, we adapt Galbraith's concepts to systems development projects as follows. Uncertainty is, at any point in time, understood as the difference between the capabilities required to successfully execute the project and the capabilities currently possessed by the project. As a consequence, projects can address the uncertainties they face either by reducing the additional capabilities required or by increasing their current capabilities.

IV. RESEARCH APPROACH

On this background, we organized an exploratory, interpretive case study [Yin 1994; Walsham 1995, 2006] of an outsourced government development project in a Scandinavian firm, SoftConsult, to address two research questions:

1. How do uncertainties emerge and interact over the life-cycle of an organic development project?
2. How does an organic development project respond to uncertainties over its life-cycle?

The research was conducted in close collaboration between the two authors [Mathiassen 2002]. One author was project manager of the studied project and the other is systems development researcher. One authors' participation in the project gave access to data that are normally very difficult to obtain. Relying on several data sources supported triangulation and helped establish a valid basis for analysis [Yin 1994]. Data about events, activities, and decisions were systematically collected based on the project manager's participation and unlimited access to the project's complete set of documentation, products, and measurement data. Additional data were collected using estimates, time used on individual program modules, errors found, change requests, and changes in plans that were collected during the project. Productivity data from the staged delivery phase were used to analyze the impact of iteration based on learning curves [Epple et al. 1991; Lapre et al. 2000]. Productivity was defined as the number of hours used to program and unit test a system function. Also, to compare across functions, we categorized them as simple, medium, or complex based on function points [Jones 1991] and inter-subjective evaluations by team members.

We adopted a combination of narratives and alternate template analysis to make sense of the data [Langley 1999]. The advantage of narratives is that they provide rich and detailed account of events, activities, and decisions as an important part of the research contribution [Langley 1999]. The disadvantage is that authors are likely to suffer from bias. The narrative was therefore developed iteratively as a joint tale [Van Maanen 1988] between the two authors; also, key project members and colleagues critically reviewed the narrative and subsequent corrections were implemented. The narratives were developed by asking for each project phase: Which uncertainties emerged? What impact did the uncertainties have? How were uncertainties addressed? What were the conditions for addressing uncertainties?

Alternate template analysis offers interpretations based on different, but internally coherent theoretical lenses. While narratives are basically inductive drawing primarily on raw data, the alternate template approach is primarily deductive drawing on theory [Langley 1999]. Our interpretations were based on Galbraith's theory [1973] and benefited from two lenses focusing on uncertainty dynamics (i.e. research question 1) and organic responses (i.e. research question 2).

V. INDUSTRIAL CONTEXT

The project was conducted by SoftConsult, a large Scandinavian supplier of IT-based solutions, on behalf of a Danish government agency. SoftConsult operates in a dynamic and complex environment and each delivered solution tends to be unique. The company engages highly trained experts that are deployed to work in multi-disciplinary teams. Projects share resources and a basic quality assurance system, but they are based on different technologies and relate to a great variety of customers. Projects are typically designed with customers and the technical architecture is designed to conform to their preferences. The company's core asset is its ability to run medium to large projects, more than knowledge about specific application areas and technologies. The company operates at different locations, and projects are occasionally staffed across locations. Whenever new technologies or applications are in demand, SoftConsult develops the necessary skills as an integral part of a customer contract. Recent large IT failures in the public sector make government agencies focused on avoiding failure.

The project took place from February 2000 to October 2001. The effort was approximately 17,000 man-hours and the system covered 3,200 function points. The project manager adopted an organic model composed of two-phase funding, staged delivery, and a combination of prototyping and specifications [McConnell 1998]. Project management was heavily inspired by state-of-the-art literature [McConnell 1998; Humphrey 1989]. During project start up, Capability Maturity Model level 2 processes [Humphrey 1989] were defined and established except for subcontract management. Analysis and overall design followed an object oriented method [Mathiassen et al. 2001] and user interface prototyping of the complete interface [McConnell 1998]. During staged delivery, the first three out of four deliveries were released to user test, not for production. The overall intention was to ensure adequacy of the design compared to user needs and to foster learning so that experiences from one phase could improve the system and processes in the next phase [McConnell 1998]. The project was reasonably successful. Compared to the original schedule it was one month late, but the delivered system included more functionality than originally agreed upon. The major phases were:

1. Getting in position (01/11/99-14/02/00): Invested in initial explorations to make a better bid
2. Preparing the bid (14/02/00-06/04/00): Developed proposal based on stated system requirements
3. Negotiating contract (06/04/00-01/06/00): Presented project proposal to customer and negotiated contract
4. Project planning (02/05/00-07/06/00: Detailed processes, a project organization, and technical infrastructure put in place
5. Analysis and design (22/05/00-09/10/00): Requirements analyzed, system designed, and development of technical architecture initiated
6. Staged delivery (23/10/00-20/08/01): Detailed design, construction, and test performed through four stages of delivery

Approximately 1,000 hours were spent on preparation and sales (01.01.2000-01.05.2000), approximately 2,600 hours on project management, education, and development of technical infrastructure, and 2,500 hours on installation, pilot tests, and handling of errors found during the first year after customer approval. Key data, including earned value [Fleming and Koppelman 2000], are summarized in Table 1.

Table 1. Overview of Project Phases

Activity	People	Hours	Deviation	Earned value
Project planning	5	450	-7%	2.8
Analysis & design	7	2,530	+14%	15.8
Stage 1	7	1,480	+18%	7.8
Stage 2	11	2,930	+10%	18.3
Stage 3	11	2,550	+3%	16.5
Stage 4	9	1,680	+27%	7.6

VI. NARRATIVES

In the following, we present two project narratives [Van Maanen 1988; Langley 1999]. The first focuses on how uncertainties emerged, interacted, and were addressed; the second focuses on how the adopted organic model impacted uncertainties during the project. The uncertainties are defined in Table 2 and the Appendix provides a summary of how they were addressed during the different stages of the project.

Uncertainty Dynamics

Getting in Position

SoftConsult knew that a major development project, to be outsourced by a government agency, was underway. The project would generate business for many years to come, the customer was already buying other services from SoftConsult, and the company believed it could win the contract. At the same time, the would-be project manager was between projects and was given time to prepare for the project. The project manager was at this point the only person engaged in the project. His initial strategy was exploratory because there was little information available. He had limited resources because there was a risk that SoftConsult would not get the contract.

The project manager addressed system requirements (#1) by seeking relevant information from colleagues that knew the government agency and by studying publicly available reports on the agency's strategy. He identified two alternative technical solutions (#2), a Web-based and a three-tier client server architecture. The project manager also started to address the development strategy (#8). The government agency preferred the waterfall model. However, the project faced many uncertainties and the project manager believed the agency would accept an organic approach if properly presented.

Table 2. Uncertainties

#	Uncertainty	Type	Concern
1	System requirements	Product	What are the customer's requirements?
2	Technical solution	Product	What technical solution is appropriate?
3	Change implications	Product	What are the implications of requirements changes?
4	User interface	Product	How will users respond to the interface?
5	Design conformance	Product	Does the design meet the requirements?
6	Implementation strategy	Product	How can we implement the design?
7	System quality	Product	Is the quality satisfactory?
8	Development strategy	Process	What strategy fits this project?
9	Developer capability	Process	What is the capability of each developer?
10	Process improvement	Process	How can we improve during execution?
11	Competition	Process	What is the competition for the contract?
12	Customer economy	Process	How big is the customer's project budget?
13	Development cost	Process	What are the development costs?
14	Development process	Process	How can we get the job done?
15	Project status	Process	How are we doing?

An organic approach would allow developers to learn through traditional analysis and design activities and at the same time provide user feedback from prototypes and versions of the system [Mathiassen et al. 2001]. Two-phase funding would further facilitate early adjustment of commitments and expectations and releasing often and early through staged-delivery would also provide valuable information [McConnell 1998]. Second, an organic approach would help develop important team capabilities. Two-phase funding would help tailor the project to the situation and staged delivery would allow "end-of-stage wrap-ups" [McConnell 1998] to improve team capabilities in subsequent stages.

The project manager addressed developer capability (#9) by collecting quantitative data from similar projects to support estimation, by committing selected colleagues to participate in the project, and by continuing to select process models and solutions from within SoftConsult. Finally, at this stage, the project manager started to identify tactics for implementing process improvements (#10) based on the staged delivery approach [McConnell 1998; Humphrey 1989]. He hoped in this way to allow learning to influence planning and management as the project unfolded.

These responses were constrained by several factors. Legislation aimed to secure fair and open competition and made it difficult to acquire customer information or take part in the customer's requirements elicitation. Such involvement would disqualify SoftConsult from participating in the bid. Also, technical decisions were not only focused on creating ideal solutions. Choice of technology had implications for where in SoftConsult the project would be placed, which people would be involved, and what career possibilities would open after project completion. Finally, quite limited resources were available because it was still unclear whether SoftConsult would get the contract. Solutions were therefore sketchy. Despite these limitations, the project manager succeeded to lay the foundation for an organic approach with improvements at each stage of delivery.

Preparing Bid

The bidding was regulated by legislation. There were six weeks for preparation and customer communication had to follow strict procedures. During this process an ad-hoc team, including the project manager, sales people, managers, and future team members analyzed 400 pages of requirements. These activities involved designing the process, preparing questions to the customer, analyzing prizing and competition issues, producing an overall design of a solution, developing a budget, conducting internal quality reviews, and finally producing and delivering the bid. The strategy was to balance between a solution the customer would prefer and one based on existing capabilities within SoftConsult. In addition, the adoption of two-phase funding split the project. The first step would lead to a detailed specification of the system. The specification was subsequently to be agreed upon by the customer. This would then form the basis for contracting the second phase based on staged delivery.

The ad-hoc team addressed system requirements (#1) through systematic studies of stated requirements, by consulting application domain experts within SoftConsult, and by addressing vague and conflicting requirements in the bid. Technical solutions (#2) were developed with internal experts on hardware, software, network technologies, and security. Also, it was decided to adopt the three-tier client server architecture building on previous projects. Finally, detailed technical decisions were postponed whenever feasible. Potential competition (#11) and the customer's budget (#12) played important roles at this stage. The team addressed these uncertainties through informal sources. The competition was considered moderate and the customer's budget sufficient to develop a satisfactory bid. The final development cost (#13) was based on function point counts, budgets from similar projects, and risk and stakeholder analyses. Slack was build into the analysis and design phase because of outstanding requirements. Some highly uncertain requirements were also separated out based on an hourly rate rather than fixed price. The rationale for the overall development strategy and a detailed description of the process (#14) was included in the bid.

Uncertainty responses during this phase were severely restricted by the short time period and limited opportunities to interact with the customer. Customer requirements were very detailed and badly structured and the customer wanted to extend contract coverage while the competition was still on. There was also internal rivalry between departments at SoftConsult. A satisfactory outcome was achieved despite these conditions. The combination of partially reduced requirements, a technical solution based on previous experience, the adoption of an organic approach, and a budget based on systematic estimates created a satisfactory basis for the project. Uncertainties related to competition and customer budget were never completely resolved; but they became irrelevant after this phase. The team knew the adoption of the traditional architecture resulted in strong developer capabilities, but it also created a new uncertainty: would the solution meet customer requirements?

Negotiating Contract

Different vendors presented their proposals and the customer decided to initiate negotiations with SoftConsult. Uncertainties were addressed by the ad-hoc team of people, now including SoftConsult's legal advisor. The team's strategy was conservative. It strongly favoured current team capabilities over exploring new opportunities. In some areas both parties wanted to reduce uncertainties (e.g. creating a common understanding of vague requirements), in other areas (e.g. changes because of releases from Microsoft, IBM, or Oracle) negotiations distributed economic risks in a way acceptable for both parties. The customer tried to improve the solution and the vendor tried to minimize implementation risks.

The technical solution uncertainty (#2) was dramatically simplified by adopting the three-tier architecture. Emerging requirements were difficult to assess (#3). The team generally refused to guarantee implied changes from operating systems, office packages, and database systems. Instead, additional analysis and design activities were included into the first phase of funding. The development process (#14) was discussed in detail so the customer was committed to the organic approach. It was agreed that the customer should develop the online help and all user-documentation based on guidelines from SoftConsult.

It was difficult for the ad-hoc team to openly acknowledge major uncertainties in front of the customer, especially on issues where the customer expected SoftConsult to be expert. There was limited time available and the customer was eager to avoid any responsibility for possible failures. The project continued to converge despite these limitations. It was, however, still quite uncertain whether team capabilities would align with the contract and whether the technical solution would satisfy customer requirements.

Project Planning

To meet deadlines, the planning process started before the contract was signed. The goal was to create appropriate conditions before development started. Detailed plans were made, the project organization defined, and process and product standards were developed. During project start up, processes for Capability Maturity Model level 2

[Humphrey 1989] were established except for subcontract management. Commitments to plans were created through meetings and reviews between the project team and customer representatives. Finally, an official kick-off seminar was conducted to give customer representatives and team members a possibility to build relations and debate the project. Requirements were at this point clarified and detailed by the project manager, the project team, customer representatives, and the steering committee. The strategy was to systematically define processes to support project tracking and oversight and to allow new insights to influence the project.

Customer requirements issues (#1) were addressed by allocating expert users to the project and through reviews conducted with customer representatives. The expert users had worked in the customer organization for many years, they knew relevant work tasks, and they were highly regarded by co-workers. User interface design (#4) was simplified by developing a standard and by having it approved across relevant user communities. The customer found this important because other projects had experienced problems getting users to agree on interface issues. The development process (#14) was further detailed in close collaboration with the customer, in particular analysis, design, and project management. Developer capability (#9) was enhanced by forming the project team, by training team members in the adopted analysis and design methods, and by assigning individual responsibilities to all team members.

Customer requirements were constantly addressed. Uncertainties were reduced on specific issues, but also increased due to changed conditions. The uncertainty of development costs was still substantial even though the project team was formed and the task was better understood. The project manager knew little about individual developer capabilities. The team included people new to SoftConsult, without formal systems development training but with many years of industrial experience.

Analysis and Design

Uncertainties were addressed by the project team and customer representatives. The strategy was to collect information and increase team capabilities to reach a satisfactory fixed-price contract for staged-delivery. The analysis and design task was accomplished by dividing the system into subsystems. Each subsystem was analyzed and designed by one or two developers in collaboration with customer representatives. Several initiatives aimed at increasing team capability and the customer was generally satisfied with the process and results.

Product standards were defined to support design across subsystems and each working group followed a similar process, using object-oriented analysis and design and user-interface prototyping. This analysis was complemented with renegotiation of complex requirements. Some requirement issues (#1) were settled by studying how the old system was working. Sometimes, users were not certain about rules and simply stated that the solution should comply with the old system, or they would explain requirements in terms of what was to be different. The interface (#4) was elaborated through prototyping and usability tests. All designs were reviewed both internally and with customer representatives and approved by the customer to ensure design conformance (#5). Two developers started full time to implement the technical architecture and prepare for programming (#6). The project manager implemented processes for project tracking (#15) based on definition of activities and deliverables and adoption of earned value analysis. Developer capability (#9) was developed in a number of ways. The project manager thoroughly checked and revised all analysis and design documents to compensate for inadequate skills amongst team members; separation of concerns was used to isolate complex implementation issues; standards were developed to support programming activities; team members were trained in using the programming environment; expected requirements changes were over estimated to compensate for unknown programming productivity; also, relevant literature and courses continued to be available for the team. Finally, the staged delivery plan was modified and detailed (#14).

The developers visited the customer organization to observe work practices (#1). Analysis and design was, however, accomplished over the summer where key users at times were inaccessible. Also, the technical platform was not yet available for experiments. Purchasing more than a year before the platform was needed was too costly and would make the hardware outdated sooner. Despite these limitations, requirements were now well understood by developers and having the customer accept design documents reduced conformance uncertainties. The implementation strategy was now clear and preparations had been made to support programming. But there were still no reliable quantitative data to support estimation. By overestimating requirements changes, the project manager attempted to create options for increasing developer capability. Most developers had limited experience with fixed-price contracts. They had previously worked with standard software and had no experience with projects where requirements, designs, and programs had to be carefully managed, documented, and approved by a customer, and where activities were estimated and tracked in detail.

Staged Delivery

Programming, test, and delivery were divided into four stages each delivering part of the system. Uncertainties were addressed by the project manager assisted by senior developers in charge of each major subsystem and in close collaboration with management, customer representatives, and the steering committee. The strategy was to use staged delivery to manage requirements issues and to use stage-wise improvement to gradually improve developer capabilities. Each stage followed the same generic process: initial re-planning of stage, detailed design, programming and unit test, integration and system test, end-of-stage user test including management of resulting changes, and finally an end-of-stage "wrap-up" to decide what to improve in the next stage. When a delivery was formally approved by the customer, the related contractual payment was released to SoftConsult.

The adopted implementation strategy (#6) was enacted by relying on experiences from previous projects and by assigning a skilled and dedicated person. To help ensure satisfactory system quality (#7), staged delivery provided early and continued feedback; expert users prepared test data and took part in designing test cases; and, the expert users' preparation of on-line help revealed weaknesses in design. The project manager continued to collect metrics from each stage and to plan stage wrap-ups to improve estimates and processes in subsequent stages. The resulting information was used to adjust the development process (#14). Project status (#15) was addressed through earned value analysis and definition of activities and deliverables. Developer capability (#9) was improved to some extent through ad-hoc initiatives. One full time developer was dedicated to work on subsequent deliverables. Slack was increased trough deliberate underestimation of programmer productivity and overestimation of changes. More team members were included and the project manager concentrated fully on the implementation effort and a dedicated resource was allocated to manage customer relations.

Staged delivery supported customer collaboration and helped manage requirements (#1), especially related to functionality and user interface design. It also made it possible to handle changes without serious delays. Changes and errors were considered in parallel with development of the next delivery. The cost of general changes and errors were reduced because they were found and addressed early. During the first test, 10 percent of the 150 problems identified by the users applied to most other modules of the same type—e.g. users wanting the system to accept a variety of formats or more help when entering data. Such problems would not have been identified early without staged delivery and they would have been more costly to fix after delivery. As a side effect, staged delivery helped create more realistic expectations both amongst users and in the project team. Staged delivery was, however, not the only way requirements issues were addressed: by having users prepare online help and test-data in parallel with development they were forced to consider the design which led to deeper understanding of requirements (#1). During programming, developers repeatedly stumbled on requirements issues (#1). These issues were typically resolved by informal contacts to expert users.

The project manager's intention was team capabilities should continually increase through systematic process improvement (#10). Informal assessments [McConnell 1998] were conducted during and after the first stage. Likewise, attempts were made to extract lessons. But the assessments revealed few insights that the team did not know in advance and most discussions were about differences in values among team members concerning what constituted professional practices or good quality. The activities were experienced as superficial by most members and were, as a consequence, abandoned. The failure to use staged delivery to increase developer capabilities made the project manager change his role. He became a coach accepting that developers were quite different, emphasizing improvement without providing solutions, and, removing concrete obstacles for individual developers. Instead of providing improved processes, the project manager used his time to solve specific problems for developers.

Staged delivery required intense coordination with the customer. For example, planning and running the acceptance tests amounted to about 800 hours. This compares to an estimated 400–500 hours of post-project-completion needed without staged delivery in similar projects within SoftConsult. This, however, came as no surprise; it was perceived as a small price to pay to effectively reduce uncertainty. The project manager realized it was difficult to practice continuous improvement based on stage wrap-ups. He considered this a failure of the adopted organic approach and he was concerned throughout the project that insufficient learning took place. Because of these concerns and the indications provided by productivity numbers from the first stages, he ended up adding more people to the project than were actually needed.

Organic Responses

The adopted organic model helped the project develop a satisfactory product. By releasing deliveries to user-tests, feedback was received in the form of error reports, change request, and informal discussions with user representatives. These and other responses related to product uncertainties are summarized in the appendix. It is difficult to rank the impact of these responses to product uncertainties, but the change requests raised by users indicate which responses triggered these requests and how each request affected redesign of system elements.

Based on this indicator, the responses related to analysis and design had the highest impact on product uncertainties, with those related to staged delivery in second place. Moreover, the impact of early releases was, as expected, more significant than that of later releases. Early releases lead to identification of general problems that were addressed before subsequent releases. Also, uncertainties about how expensive the project would be and how much the project would be delayed, made both the customer and project manager reluctant to encourage and approve changes.
In contrast, the organic approach did not effectively address process uncertainties. The systematic attempts at process improvement during staged delivery failed and the project manager had to address process uncertainties through a number of ad-hoc initiatives. The appendix summarizes the responses to process uncertainties. Again, it is difficult to rank individual responses by impact. The project manager's perception was, however, that the impact of dividing the construction into stages and using experience from an end-of-stage wrap-up to systematically improve the development process in the next stage was close to zero.

According to the project manager, the two most important responses to process uncertainties were the up-front investment in planning and preparing development (e.g. development of standards, templates, tools and plans) and the informal and unmanaged learning activities carried out by individual developers as an integral part of doing their job. Despite the failure to practice planned and managed improvements, developers in general kept improving their performances for about three to four months after they were assigned to the project before productivity stabilized, and these changes in productivity were much larger than anticipated. The improvements for most developers' resembled learning curves in the literature [Epple et al. 1991; Lapre et al. 2000]. Some developers increased their productivity more than 100 percent during the first months. These were not junior programmers on their first assignment, but well-educated programmers with more than five years of experience. These improvements were, however, fragile and highly dependent on distribution of responsibilities. Moving a programmer from a subsystem he had designed to a subsystem designed by another developer instantly decreased his productivity. The opposite effect was seen when developers did two similar tasks in sequence. Then productivity improved in the following task. As a consequence, the difference between estimated and actual productivity was reduced from stage-1 to stage-3, but then increased again in stage-4 where some radically different tasks were involved, see Table 1.

There were two important reasons that the planned improvement efforts failed. First, one assumption in software process improvement is that developers, at some level, use the same processes. Team members had, however, different qualifications, preferences, and work styles. They needed different processes, they had particularly strong preferences for specific working styles when doing technical work, and their development approach changed over time. Even though there were similarities across individuals, the idea of having common processes and improving systematically upon them was not feasible. Second, the team members' commitment to develop an appropriate solution and the stressful context with strong focus on deadlines made it difficult to engage them in thorough reflections about work practices. Frequent deadlines created a short term out-look in which process issues were pushed to the background. Moreover, learning about and improving development processes turned out to be far more difficult than learning about customer requirements, and the adopted organic model offered little advice on how to do this.

The organic model did not only fail to respond effectively to process uncertainties. It introduced additional ones. During staged delivery, a number of issues emerged resulting in increased process uncertainties:

- *Sequencing*: Which functions should be implemented in which deliveries? Criticality of functionality was not the only criterion; the project manager had to consider productivity, coherence, and politics as well.
- *Frequency*: How often should the project deliver? There were approximately three months between deliveries. That was far from the short cycles recommended in the literature; but it was difficult to reach deadlines because there was little room for unexpected events. Whenever a team member was late, resources had to be moved to meet the deadline. That had a negative impact on short-term productivity because it broke the planned continuity and made it more difficult to exploit task specific knowledge. Staffing issues also became complicated. The project was highly vulnerable to team members' absence and it was difficult to integrate new members because the team was busy reaching the next deadline.
- *Rework*: What should be redone as a consequence of requirements changes? Changes created a need for precise and current information about dependencies between modules and functions. They also created a need for information about which tests had to be redone. Customer tests at the end of each stage resulted in the identification of 313 errors and a total of 297 change requests representing approximately 1,600 hours of work. In practice, this turned two-phase funding into many-phase funding and it added significant management overhead to reestimate activities and re-negotiate contractual issues.
- *Deadlines*: When can we actually deliver? Deadlines were considered contractual events that had to be reached on time. Neither the customer nor the supplier wanted to miss a deadline given the media

interest in failing projects within the public sector. Always having a relative short time period to the next delivery combined with change requests and a great variation in personal productivity placed the project team in a permanently stressful situation.

VII. ANALYSES

In the following, we analyze the case based on task uncertainty and systems development theory. In response to research question 1, we identify insights related to understanding uncertainties at SoftConsult. In response to research question 2, we identify insights related to understanding responses to these uncertainties.

Understanding Uncertainty

Uncertainty types. Table 2 identifies fifteen different uncertainties categorized as *product uncertainties* (dealing with the client's problem and its solution) and *process uncertainties* (dealing with the organization of the project). The two key uncertainties, system requirements (#1) and development process (#14), reflects this dual nature of task uncertainty in the *SoftConsult* project. While both types were addressed throughout the project life-cycle they were impacted differently by the organic approach. As detailed throughout the uncertainty dynamics narrative earlier, two phase funding, staged delivery, and the combination of specifying and prototyping [McConnell 1998] helped reduce all major product related uncertainties. The organic approach also provided overall guidance to handle process uncertainties, in particular in relation to development strategy (#8) and process improvement (#10). However, important process uncertainties, e.g. related to the development process (#14), kept posing serious challenges throughout the project.

Interestingly, impacts of organic approaches in the literature are mainly focused on product related issues: number of system features [Gordon and Bierman 1995; Boehm and Papaccio 1988], system performance [Gordon and Bierman 1995], system quality [Gordon and Bierman 1995; Mahmood 1987], and system maintenance [Gordon and Bierman 1995; Naumann and Jenkins 1982]. While some attention is drawn to process issues like effort and user participation [Gordon and Bierman 1995], there is little emphasis on emerging process issues. On several key issues, for example frequency and sequencing of stages, rework, and management of deadlines, no or insufficient advice was provided by the adopted organic model. This lack of emphasis on process uncertainties in the organic development literature is inconsistent with the experiences from SoftConsult.

Capability types. We adapted Galbraith's notion of task uncertainty [1973] to indicate the gap between the capabilities required and the current capabilities in a systems development project. While this simple notion of uncertainty applied well to the case, further elaborations are needed to explain how project capabilities were challenged and developed at SoftConsult. We observed two quite different types of capabilities. First, *know-what capabilities* focused on what the project needed to known or actually knew about requirements, solutions, development strategy, and process. Know-what capabilities related to specific areas of knowledge (e.g. a new system function); and, they were typically created through analysis, by asking colleagues or customers, by making decisions, or through renegotiations with the client. Second, *know-how capabilities* focused on specific skills or experiences required or actually possessed by the project to support problem-solving and collaboration (e.g. to program system functions, to test modules, or to interact with users). Know-how capabilities were brought into the project through participants' earlier experiences and they were further developed through training, practicing, and collaboration with peers. Know-how capabilities were, however, experienced as quite different and as more difficult to develop than know-what capabilities.

The distinction between know-what and know-how capabilities makes sense in relation to the systems development literature. A project's perception of its task [McFarlan 1981] can, for example, be seen as a relation between the client's problem (know-what) and the team's problem-solving capability (know-how). A project's experience with technology [McFarlan 1981] expresses a relation between general knowledge about applied technologies (know-what) and the team's skills and experiences related to these technologies (know-how). Finally, a project's size and complexity [McFarlan 1981] can be seen as a relation between the project's characteristics (know-what) and the project manager's experience (know-how). However, the literature offers no explication of know-what and know-how capabilities or similar distinctions that describe the different aspects of uncertainty experienced at SoftConsult. On the contrary, Larman and Basili [2003] argue systems developers need to create know-what capabilities about the problem and its solution and iteration is in most cases required to do so. While this is definitely true, there is no mentioning of the needs for creating know-how so projects can successfully adopt and leverage organic models. Similarly, McConnell's [1998] two-phase funding and staged delivery approaches are focused on creating the necessary know-what capabilities without sufficiently detailing the consequences for know-how management.

Uncertainty dynamics. The project manager at SoftConsult not only had to deal with many uncertainties, but these uncertainties interacted throughout the project to constantly create new needs to process information. Again, we

observed considerable differences in how dynamics impacted the project across product and process uncertainties. The key product uncertainty, system requirements (#1), was for example addressed from the very start and interacted with most other uncertainties throughout the project life-cycle. All requirements changes were, however, treated as adjustments to the initial call for tender, so the overall profile and structure of requirements remained stable from the preparing bid phase and onwards. The key process uncertainty, development process (#14), was addressed from the second phase and also interacted with most of the other uncertainties. However, the development process was influenced by other uncertainties in ways that made it increasingly complex and dynamic during staged delivery. As a consequence, the project manager had to focus all his energy on managing internal process issues during the delivery stages of the project.

The literature highlights several challenges related to manage uncertainty in systems development (see Section II). Some are implications of the adoption of an organic approach (e.g. combining specification and prototyping), some are implied by other uncertainties (e.g. the need to renegotiate contracts is implied by changes in requirements), and some represent changes in the profile of major uncertainties (e.g. ensuring convergence on the solution). The literature offers, however, no account of how uncertainties interact and how uncertainty profiles change over the project life-cycle. The implicit, underlying assumption seems to be that uncertainty is the problem and organic models the solution. There is little understanding of how the complex and dynamic relationship between uncertainties and adoption of specific organic models dramatically influence a project manager's agenda as it did at SoftConsult.

Understanding Responses

Response types. We observed a variety of uncertainty responses at SoftConsult. Combining Galbraith's [1973] two generic responses, i.e. offensively developing new capabilities and defensively reducing the need for additional capabilities, with uncertainty and capability types, reveals a total of *eight different response types*. Table 3 shows examples of how each of these was adopted at SoftConsult.

Table 3. Response Types

		Product uncertainties	Process uncertainties
Offensive responses	**Know-what capabilities**	Systematically evaluating user interface prototypes.	Tracking project on progress and productivity.
	Know-how capabilities	Training users to play active role during design and test.	Training team members in analysis and design methodology and programming environment.
Defensive responses	**Know-what capabilities**	Refusing to guarantee changes in related application and standard software.	Building slack into schedule and budget.
	Know-how capabilities	Adopting three-tier architecture known from previous projects.	Separating out high uncertainty requirements based on hourly rate.

The project generally adopted a rich variety of responses. Defensive responses normally took the form of a decision or a negotiation between stakeholders, they were least time consuming, and they typically focused on ensuring the project was completed within schedule. Offensive responses also took a variety of forms, they typically required some or considerable investments, and they pushed the project in direction of a more satisfactory product.

Response context. We also observed that uncertainty responses often were shaped by the project context. The bidding was, for example, constrained by laws that excluded useful offensive responses. SoftConsult was not allowed to take part in the customer's requirements elicitation, the bid had to be prepared within six weeks, and strict procedures had to be followed to secure fair and open competition. The initial context encouraged, in this way, adoption of defensive responses even though more offensive ones could have been valuable. When resources, time, and possibilities for interaction with customer and users were limited, the project manager responded by aligning the task with available know-how within SoftConsult. Later, during staged delivery the project had to commit to specific deadlines even though major process issues had not been resolved. That created a context in which the options to effectively address product related uncertainties were severely restricted.

These findings from SoftConsult suggest uncertainty responses depend on two issues [Keil et al. 1998]. One is the project's understanding of the uncertainty it faces and how these develop over time. The other is the context in which the project operates and the degree of control the project can exercise over specific uncertainties. At SoftConsult, two events dramatically changed the context. The first was the customer's choice of SoftConsult by which the project acquired the first phase of funding. The second was the agreement about a contract for staged delivery by which the project acquired the second phase of funding.

Response modes. Changes in perception of uncertainties as well as in the context for responding to uncertainties led to three different response modes over the project life-cycle: *competitive mode*, *risk-hedging mode*, and *collaborative mode*, see Table 4.

Table 4. Response Modes

Mode	**Competitive**	**Risk-Hedging**	**Collaborative**
Phases	Getting in position Preparing bid	Negotiating contract Project planning Analysis and design	Staged delivery
Objective	Win contract	Reduce risks	Meet deadlines
Integration	Internal	External	Internal
Uncertainty	Product	Product-Process	Process
Capability	Know-what	Know-what	Know-how
Response	Offensive	Defensive	Offensive

The project initially (01/11/99-06/04/00) operated in competitive response mode with the primary objective to win the contract. Due to laws and regulations, there was little integration between the emerging project at SoftConsult and the customer, and the project manager put considerable effort into internal networking. Counter to the ideal during the early stages of a project [McFarlan 1981], there was little emphasis on external integration. The main focus was on internal integration to reduce product uncertainties by offensively generating know-what capabilities about the problem and its solution.

After the customer decided to negotiate with SoftConsult (06/04/00-09/10/00), responses changed to risk-hedging mode [Lee, 2002]. The primary focus shifted to distributing economic risks between customer and provider and to detailing the second-phase contract for staged delivery [Boehm 1988]. The objective for the project was to reduce its risks and responses were consequently defensive. There was equal emphasis on developing know-what capabilities about the product (detailing system requirements and the technical solution) and the process (developing a plan for staged delivery).

Finally, during staged delivery (23/10/00-20/08/01), the project operated in collaborative response mode with primary focus on meeting deadlines. The project manager delegated customer relationships to another team member and focused entirely on internal integration of the team including close collaboration with expert users. The focus was primarily on managing the process by offensively using and cultivating available know-how capabilities. The attempts to systematically improve available know-how based on end-of-stage "wrap-ups" failed. Instead, developers improved capabilities as an integral part of doing their job. As a consequence, the project delivered the system only one month later than scheduled, and with more functionality than anticipated.

The literature suggests that development projects generally should emphasize exploration in early phases to allow for feedback from customers and the market, and then increasingly emphasize predictability and control in the later stages as uncertainties are resolved [Boehm 1988; Mathiassen and Stage 1992; Chillarege 2002]. In the SoftConsult project, early explorations were, however, severely restricted by the context for winning the contract and by the focus on distributing risks between the customer and SoftConsult. Also, the construction mode was initiated before all product uncertainties were addressed so the contract for staged delivery included severe outstanding uncertainties. The project was, therefore, not able to realize the ideals expressed in the literature. Two-phase funding [McConnel 1998] is a simplified, pragmatic version of the multi-phase approach of the spiral model [Boehm 1988], and it represents a huge improvement over one-shot, fixed contracts because it allows for initial reduction of uncertainties before a final contract is negotiated. However, we saw at SoftConsult how early explorations were

severely restricted in the first funding phase and how the second phase was left with challenging uncertainties making the fixed, staged delivery contract a somewhat problematic proposition.

VII. CONCLUSION

The key contribution of this research is the detailed insights it gives into management practices in organic systems development. The unlimited access to data from SoftConsult made it possible to provide detailed narratives of systems development practices that are rarely seen in the literature. The presentation and interpretation of the case were based on analyses of how uncertainties and responses manifested themselves and interacted. To manage organic projects effectively, the research suggests to distinguish between product and process related uncertainties, between know-what and know-how capabilities, and to emphasize that uncertainties are highly dynamic as projects unfold. The research also suggests distinguishing between offensive and defensive responses, to focus on how responses depend on context, and to appreciate that different response modes become appropriate as a project unfolds. The adopted organic model did help respond to product uncertainties at SoftConsult. At the same time, however, the model offered quite limited and in some ways inappropriate help to respond effectively to emerging process uncertainties.

Based on Galbraith's theory [1973], it makes sense to see organic systems development models as alternatives to the single-pass and document-driven waterfall model [Royce 1970] that become increasingly appropriate as the task uncertainty increases. We adapted Galbraith's theory to propose the overall distinction between offensive and defensive responses. We could also identify specific responses in the case corresponding to creation of slack resources (i.e. build slack into schedule and budget) and to creation of lateral relations (i.e. informal contacts and discussions between developers and users). However, we didn't find the other detailed organizational responses in Galbraith's theory particularly useful in distinguishing systems development practices. As a result, we developed a uncertainty response framework dedicated to organic development projects (Tables 3 and 4).

While the adaptation of task uncertainty theory [Galbraith 1973; Iivari 1992] helped us make sense of managerial practices in the organic project at SoftConsult, our findings are of a preliminary nature and needs to be further validated and developed in relation to other organic models and industrial contexts. It is also worth noting other important contributions within our field provide complementary views to task uncertainty theory. Feldman and March [1981] have for example demonstrated that organizations not only produce information in response to requests and as a basis for rational choice and decision-making. Organizations also systematically gather more information than they need to serve symbolic purposes. Another complementary view is offered by Ngwenyama and Lee [1997] who demonstrate that critical social theory reveal informational behaviours in organizations that are not accounted for through traditional theory. It is for these reasons important to engage in complementary research that can help us understand the broader cultural and political dimensions of managing uncertainty in organic systems development.

Our findings suggest managers of organic projects need to stay pragmatic and balance the ideals build into organic models with the realities enforced by laws, contractual agreements, organizational conditions, and the highly complex and dynamic nature of organic projects. Organic development is not a solution to the uncertainty challenge. Rather it is a framework for identifying and addressing uncertainties as they emerge and develop over the project life-cycle. Organic models help reduce product uncertainties, but they also introduce and reinforce process uncertainties requiring complementary management tactics throughout the project.

Managers of organic projects are advised to combine offensive responses in which they seek to increase team capabilities with defensive responses in which they seek to reduce the need for additional capabilities. Defensive responses are executed through decisions or negotiations with involved stakeholders. Offensive responses are executed through staffing, studying, experimentation, experiencing, or improvisation. Our research suggests that defensive responses are the least time demanding and typically focus on ensuring that the project is completed within schedule. Offensive responses require some or considerable investments and typically push the project in direction of more satisfactory results. Finally, managers are advised to differentiate between know-what and know-how capabilities and to dynamically adapt their response mode to fit the project's evolving context.

REFERENCES

Alavi and Wetherbe. (1991). "Mixing Prototyping and Data Modelling For Information System Design," *IEEE Software* 8(3), 86 – 91.

Alter and Ginzberg. (1978). "Managing Uncertainty in MIS Implementation," *Sloan Management Review* 20(1), 23-31.

Basili and Turner. (1975). "Iterative Enhancement: A Practical Technique for Software Development," *IEEE Transactions on Software Engineering*, SE-1(4), 390–396.

Baskerville and Stage. (1996). “Controlling Prototype Development through Risk Analysis,” *MIS Quarterly* 20(4), 481-504.

Baskerville and Pries-Heje. (2004). “Short Cycle Time Systems Development,” *Information Systems Journal* 14(3), 237-264.

Beck. (1999). *eXtreme Programming Explained, Embrace Change*. Upper Saddle River, N. J, Addison-Wesley.

Boehm. (1988). “A Spiral Model of Software Development and Enhancement,” *Computer* 21(5), 61–72.

Boehm and Papaccio. (1988). “Understanding and Controlling Software Costs,” *IEEE Transactions on Software Engineering* 14(10), 1462 –1477.

Boehm and Turner. (2004). *Balancing Agility and Discipline: A Guide for the Perplexed*. Boston, Massachussets: Addison-Wesley.

Brooks. (1987). “No Silver Bullet,” *Computer* 20(4), 10–19.

Brooks. (1975). *The Mythical Man-Month: Essays on Software Engineering* Reading, Mass.: Addison-Wesley.

Burns and Stalker. (1961). *The Management of Innovation*. London, UK: Tavistock Institute.

Chillarege (2002). The Marriage of Business Dynamics and Software Engineering. *IEEE Software*, 19(6), 43 - 49.

Cockburn and Highsmith. (2001). “Agile Software Development: The People Factor,” *IEEE Computer* 34(11), 131-135.

Curtis, Krasner and Iscoe. (1988). “A Field Study of the Software Design Process for Large Systems,” *Communications of the ACM* 31(11), 1268 – 1287.

Epple, Argote and Devadas. (1991). “Organizational Learning Curves: A Method for Investigating Intra-Plant Transfer of Knowledge Acquired Through Learning by Doing,” *Organization Science* 2(1), 58-70.

Fleming and Koppelman. (2000). *Earned Value Project Management*, Pennsylvania: Project Management Institute.

Galbraith. (1973). *Designing Complex Organizations*. Reading, Massachusetts: Addison-Wesley.

Goodhue and Thompson. (1995). “Task-Technology Fit and Individual Performance,” *MIS Quarterly* 19(2), 213-236

Gallivan and Keil. (2003). “The User–Developer Communication Process: A Critical Case Study,” *Information Systems Journa,* 13(1), 37-68.

Gould and Lewis. (1985). “Designing for Usability: Key Principles and What Designers Think,” *Communications of the ACM* 28(3), 300 – 311.

Gordon and Bierman. (1995). “Rapid Prototyping: Lessons Learned,” *IEEE Software* 12(1), 85 – 95.

Gresov (1989). Exploring Fit and Misfit with Multiple Contingencies. *Administrative Science Quarterly*, 34, 431-453.

Humphrey. (1989). *Managing the Software Process. SEI Series in Software Engineering*, Addison-Wesley, USA.

Iivari (1992). The Organizational Fit of Information Systems. *Information Systems Journal*, 2, 3-29.

Jacobsen, Booch and Rumbaugh. (1999). *The Unified Software Development Process*. Reading, Massachusetts: Addison-Wesley.

Jones. (1991). “Applied Software Measurement Assuring Productivity and Quality,” *Software Engineering Series*, New York, McGraw-Hill.

Keil, Cule, Lyytinen and Schmidt. (1998). “A Framework for Identifying Software Project Risks,” *Communications of the ACM* 41(11), 76-83.

Langley. (1999). “Strategies for Theorizing from Process Data,” *Academy Management Review*, Vol. 24, No. 4, pp. 691-710.

Lapre, Mukherjee and Van Wassenhove. (2000). “Behind the Learning Curve: Linking Learning Activities to Waste Reduction,” *Management Science*, 46(5), 597–611.

Larman and Basili. (2003). “Iterative and Incremental Development: A Brief History,” *IEEE Software* 36(6), 47-56.

Lott. (1997). “Breathing New Life Into the Waterfall Model,” *IEEE Software* 14(5), 103 –105.

Mahmood. (1987). “System Development Methods – A Comparative Investigation,” *MIS Quarterly* 11(3), 293 – 311.

Mathiassen and Stage. (1992). “The Principle of Limited Reduction in Software Design,” *Information, Technology & People* 6(2), 171-184.

Mathiassen, Seewaldt, and Stage. (1995). "Prototyping and Specifying: Principles and Practices of a Mixed Approach," *Scandinavian Journal of Information Systems* 7(1), 55 – 72.

Mathiassen. (1998). "Reflective Systems Development," *Scandinavian Journal of Information Systems* 10(1-2), 67 – 118.

Mathiassen, Munk-Madsen, Nielsen and Stage. (2001). *Object Oriented Analysis & Design*. Aalborg, Denmark: Marko Publishers.

Mathiassen (2002). Collaborative Practice Research. *Information, Technology & People*, 15(4), 321-345.

McConnell. (1998). *Software Project Survival Guide*. Redmond, Washington: Microsoft Press.

McFarlan. (1981). "Portfolio Approach to Information Systems," *Harvard Business Review* 59(5), 142-150.

Mintzberg. (1983). *Structures in Fives: Designing Effective Organizations*. Englewood Cliffs, N. J.: Prentice Hall.

Mohr. (1982). *Explaining Organizational Behavior*. San Francisco, CA: Jossey-Bass.

Naumann and Jenkins. (1982). "Prototyping: The New Paradigm for Systems Development," *MIS Quarterly* (6:3), 39 - 48.

Orr. (2004). "Agile Requirements: Opportunity or Oxymoron?" *IEEE Software* 21(3), 71-73.

Parnas and Clements. (1986). "A Rational Design Process: How and Why to Fake it," *IEEE Transactions on Software Engineering* 12(2), 251 – 257.

Pfleeger. (2001). *Software Engineering Theory and Practice*, Second Edition. Prentice Hall.

Rai and Al-Hindi. (2000). "The Effects of Development Process Modeling and Task Uncertainty on Development Quality Performance," *Information & Management* 37(6), 335-346.

Rising and Janoff. (2000). "The Scrum Software Development Process for Small Teams," *IEEE Software* 17(4), 26 – 32.

Royce. (1970). "Managing the Development of Large Software Systems: Concepts and Techniques," Proceedings WesCon.

Sarkkinen and Karsten. (2005). "Verbal and Visual Representations in Task Redesign: How Different Viewpoints Anter into Information Systems Design Discussions," *Information Systems Journa,* 15(3), 181-212.

Sotirovski. (2001). "Heuristics for Iterative Software Development," *IEEE Software* 18(3), 66 – 73.

Turk, France, and Rumpe. (2005). "Assumptions Underlying Agile Software-Development Processes," *Journal of Database Management* 16(4), 62-87.

Tushman and Nadler. (1978). "Information Processing as an Integrating Concept in Organizational Design," *Academy of Management Review* 3, 613-624.

Van de Ven and Drazin. (1985). "The Concept of Fit in Contingency Theory," *Research in Organizational Behavior* 7, 333-365.

Van Maanen. (1988). *Tales From the Field: On Writing Ethnography*. Chicago: Chicago Press.

Walsham. (1995). "Interpretive Case Study in IS Research. Nature and Method," *European Journal of Information Systems* 4, 74 - 81.

Walsham. (2006). "Doing Interpretive Research," *European Journal of Information Systems* 15(3), 320-330.

Wirth. (1971). "Program Development by Stepwise Refinement," *Communications of the ACM* 14(4), 221 – 227.

Yin. (1994). *CASE Study Research Design and Methods*, Second Edition, Applied Social Research Methods Series, Volume 5, SAGE Publications.

ABOUT THE AUTHORS

Lars Mathiassen received his master's degree in computer science from Aarhus University, Denmark, in 1975, his Ph.D. in informatics from Oslo University, Norway, in 1981, and his Dr. Techn. degree in software engineering from Aalborg University, 1998. He is currently GRA Eminent Scholar and professor in Department of Computer Information Systems and co-founder of Center for Process Innovation at Georgia State University. His research interests are within information systems and software engineering with a particular emphasis on process innovation. He has coauthored *Computers in Context* (Blackwell 1993), *Object Oriented Analysis & Design* (Marko Publishing, 2000), and *Improving Software Organizations* (Addison-Wesley, 2002). He has served as senior editor for *MIS*

Quarterly and his research is published in journals like *Information Systems Research, MIS Quarterly, IEEE Transactions on Engineering Management, Communications of the ACM, Journal of AIS, Information Systems Journal,* and *IEEE Software.*

Keld Pedersen holds a masters degree and Ph.D. in computer science from Aalborg University, Denmark. He has been involved in systems development research, practice, and education for more than 20 years. During this period his primary interests has been project management, software process improvement, organizational learning, and knowledge management. He has always positioned himself in the borderland between research and practice, and has published several papers based on empirical data from his participation in systems development projects and software process improvement initiatives. Currently, Keld Pedersen is engaged in a European collaborative research project investigating the possibilities for supporting knowledge sharing in systems development projects with wiki, semantic web and personalization technologies.

APPENDIX: SUMMARY OF UNCERTAINTY MANAGEMENT

#	Uncertainty	Getting in position	Preparing the bid	Negotiating contract
1	System requirements	Asking colleagues Studying public reports	Analyzing requirements Consulting domain experts Addressing uncertain requirements	
2	Technical solution	Identifying alternative technical solutions	Using internal expertise Using experience from similar projects Postponing decisions	Adopting three-tier architecture
3	Change implications			Rejecting uncertain change requests
4	User interface			
5	Design conformance			
6	Implementation strategy			
7	System quality			
8	Development strategy	Proposing organic approach		
9	Developer capability	Estimating based on similar projects Selecting project members Adopting *SoftConsult* processes and solutions		
10	Process improvement	Tactics for process improvements Adaptive planning and management		
11	Competition		Information from informal sources	
12	Customer economy		Information from informal sources	
13	Development cost		Counting function points Using experience from similar projects Risk and stakeholder analyses Planning with slack Hourly rate on uncertain requirements	
14	Development process		Including strategy and process in bid	Committing customer to organic approach
15	Project status			

#	Uncertainty	Project planning	Analysis and design	Staged delivery
1	System requirements	Allocating users to project Conducting reviews with customer s	Collaborating with users Analysis and design Prototyping Studying old system Observing work practice	Feedback on increments Users preparing on-line help and test data Addressing uncertain requirements
2	Technical solution			
3	Change implications			
4	User interface	Agreeing on standard interface	Prototyping and usability tests	
5	Design conformance		Design reviews Customer approval	
6	Implementation strategy		Implement architecture Prepare for programming	Using project experience Assigning skilled person
7	System quality			Staged delivery feedback Users preparing on-line help and test data
8	Development strategy			
9	Developer capability	Forming project team Training team in analysis and design Assigning individual responsibilities	Document quality check Separation of concerns Programming standards Training in programming environment Over estimating requirements changes Team access to literature and courses	Ad-hoc initiatives
10	Process improvement			Informal assessments Extracting lessons Coaching developers Removing individual obstacles
11	Competition			
12	Customer economy			
13	Development cost			
14	Development process	Involving customer in detailing of process	Modifying staged delivery plan	Stage-wrap-ups to improve process
15	Project status		Earned value analysis	Earned value analysis.

Communications of the Association for Information Systems

ISSN: 1529-3181

Communications of the Association for Information Systems