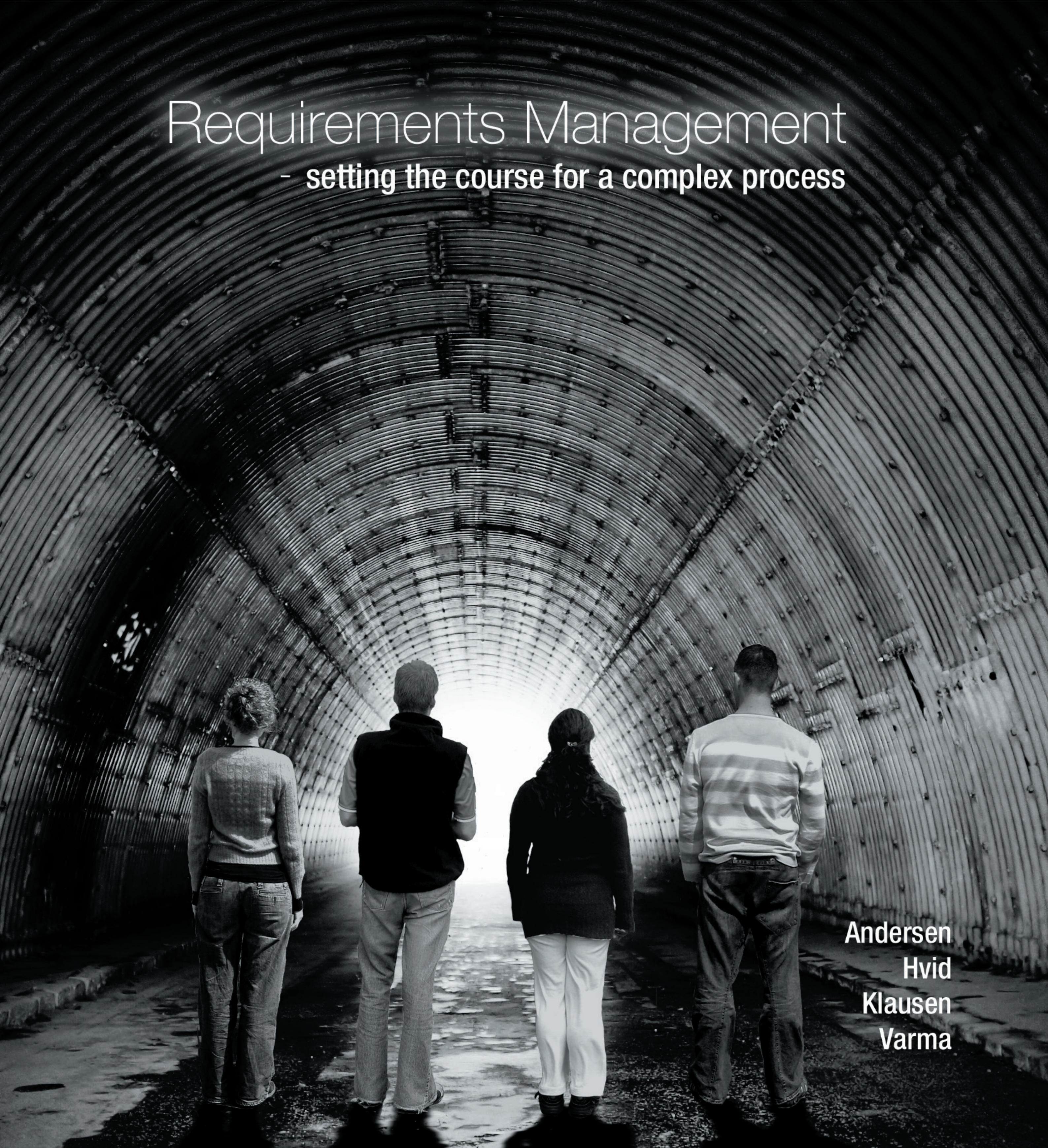


Requirements Management

- setting the course for a complex process

Andersen
Hvid
Klausen
Varma

Now I'm not looking for absolution
Forgiveness for the things I do
But before you come to any conclusions
Try walking in my shoes
Try walking in my shoes

- Depeche Mode

Summary

The topic of this project is requirements management within information systems development, and more specifically how the requirements process is managed in the plan-driven and agile paradigm. These paradigms are exemplified by the development methods Scrum and OOA&D. The plan-driven paradigm is considered to be the dominating approach in requirements management. This assumption is based on the first part of the project, which is a research paper concerning a literature survey on how changing and evolving requirements are perceived and advised to be handled. The purpose of this survey was to give an overview of the research concerning requirements management issues.

Our aim, in the second part of the thesis, is to examine how the functionalist development paradigm can be nuanced. The approach for examining it is inspired by C. West Churchman's presentation of five different traditions of inquiry, which is based upon five different philosophical directions. The five traditions each entail particular strategies for understanding and behaving in the world.

To dissect the impact of the philosophical strategies within systems development, two main literary sources are introduced. The first defines the characteristics of four different philosophical paradigms by means of Gibson Burrell and Gareth Morgan's book *Sociological Paradigms and Organisational Analysis*. Using this as a basis, Rudy Hirschheim and Heinz K. Klein wrote the article

Four Paradigms of Information Systems Development in 1989. In this article the relation between the approach applied for system development and the design, and configuration of the end system is discussed. The article describes how systems development is perceived within each paradigm, as well as who should be involved in the development process and what kind of system should be developed.

Hirschheim and Klein's article is used to put the paradigms into a systems development context. The article takes the sociological theories of Burrell and Morgan from a macro level, and uses them at a micro level. We have supplemented Hirschheim and Klein with Burrell and Morgan's work in order to refine their paradigms, as we found these too simple in relation to requirements management.

Halfway through the second part of the project we change to a narrative writing style. Here Hirschheim and Klein's paradigms are exemplified by four project managers. In order to identify their world view, approach to systems development, and opinion on Scrum and OOA&D, a complex fictitious project is introduced. This project is presented as an assignment to the functionalist project manager, for which he seeks advice from the three other project managers (relative socialist, radical structuralist and radical humanist). He takes their advice into consideration, when he has decided upon which method to use for the project.

The functionalist uses risk analysis in order to make a decision on which development method to apply for the project proposal. In general the rankings of the two methods are fairly equal, but in the end he chooses OOA&D as his preferred method, when taking the inputs from the other three project managers into consideration.
In order to make the method applicable for the project he decides to make some refinements. But the advice of the three project managers has not been in vain, as the functionalist makes these refinements according to risks and concerns pointed out by them. The refinement of OOA&D divides the development process into two stages. The first stage is incremental and concerns the elicitation, negotiation, and documentation of requirements. The next stage is split into two processes, which are run in parallel. One process is plan-driven

and concerns the design and implementation of the system architecture, with no user involvement. The other process is incremental, where the user interface is implemented and validated in order to reach a mutual understanding of the system. This makes the method open to changes from the users, and facilitates a common understanding of the system. Further, the division of the implementation stage, into the architectural and user interface processes, has been done to assure a solid architectural design.

As a closure to the project, a critical standpoint is taken to the literature, which we have based the project upon. It is mentioned, that the paradigms introduced by Burrell and Morgan, both can be perceived as guidelines for researchers and being restrictive on innovative research. Further, we criticize Hirschheim and Klein for not properly adapting Burrell and Morgan's conflict term.

Finally, thoughts, on how our contribution can inspire further research in the field of computer science, are outlined. Here the importance of focusing on social sciences is emphasized. We find it worrying that even if the changes occurring in software developing processes are becoming a natural part of the development process, the plan-driven approach still seems to be dominating. Like any other field, computer science has to evolve with the environment in which it rests.

Preface

Requirements management is an essential part of information systems development and it is the main topic of this thesis.

Within software development two extreme approaches are the plan-driven and the agile development approaches. The older approach is plan-driven, where requirements are analyzed and specified up-front. On the other hand, the more recent agile development approach promotes requirements change as an integrated part of the development process. The thesis has the aim to give an overview of this particular topic and to give a theoretical discussion on two different approaches for system development from four different philosophical world views.

We would like to thank our supervisor, Ivan Aaen, at the Department of Computer Science at Aalborg University, for his guidance throughout our thesis.

June 2006

Karsten Stig Andersen, kost@cs.aau.dk
Louise Dalum Hvid, hvid@cs.aau.dk
Jeppe Klausen, syrex@cs.aau.dk
Gauri Varma, gauri@cs.aau.dk

Introduction

> *"Faced with the conflicting pressures of accelerated product development and users who demand that increasingly vital systems be made ever more dependable, software development has been thrown into turmoil. Traditionalists advocate using extensive planning, codified processes, and rigorous reuse to make development an efficient and predictable activity that gradually matures toward perfection. Meanwhile, a new generation of developers cites the crushing weight of corporate bureaucracy, the rapid pace of information technology change, and the dehumanizing effects of detailed plan-driven development as cause for revolution."*
>
> ***- Boehm [1 p 64]***

The prevalent approaches used for developing information systems today can be characterized as plan-driven development methods, where requirements management is a matter of controlling changes to requirements, which are agreed on early in the process. This is supported by recognized books on software engineering as well as the SWEBOK. Two recognized authors of software engineering literature, Sommerville and Pressman, both pay little attention to newer approaches in software development, like the agile approach. They pay attention to activities concerning development where documentation and thorough analysis are central [6, 4]. Further the presentation, the SWEBOK gives of the generally accepted knowledge areas in software engineering, also indicates that documentation is central, and attention is mainly on the product and not the process [2].

The **SWEBOK** is a joint IEEE and ACM project with the aim to give a snapshot of the state in the field of software engineering.

The concept of **wicked problems** was originally proposed by H. Rittel and M. Webber [5].

It is stated, that many of the problems system developers face have the characteristics of *wicked problems* [3]. A *wicked problem* occurs when there is no consensus on what the actual problem is. Formulating the problem is essentially the same as devising a solution for the problem. A specific solution implies a certain understanding of the problem. Further, as it is not possible to define the problem, there is no way of telling, when the problem is solved. Also, various stakeholders will have different views of acceptable solutions. *Wicked problems* arise when an organization needs to deal with something new, with change, and when multiple stakeholders have different ideas about how the change should take place.

The Complexity of System Development

Developing and introducing an information system into an organization can be a very complex task. Sommerville follows Poppendieck's perception of the tasks, that system developers face:

> *"The problems that software engineers have to solve are often immensely complex. Understanding the nature of the problems can be very difficult, especially if the system is new. Consequently, it is difficult to establish exactly, what the system should do."*
>
> ***- Sommerville [6 p 98]***

A software system is built according to some requirements for, what the system should do. However, it is impossible to fully specify systems requirements for a system that addresses a problem, that is so complex that even the problem cannot be fully specified. A better understanding of the problem emerges, as development proceeds and a solution is developed.

Requirements for large systems are always changing due to various reasons. One reason is that requirements cannot be fully defined from the beginning; and as the understanding of the problem increases, the requirements change. Another reason is that larger systems have a diverse group of users, who have different priorities and express different requirements [6]. Besides users, there are many other stakeholders in relation to a system, each with different and changing needs that they may even have difficulties expressing [4]. In addition to the diversity of stakeholders, there can be more reasons for changing requirements; for example, the business environment can change, legislation can change, or newly discovered technical opportunities and limitations can call for changes in requirements. All these reasons cause requirements for a system to be complex and volatile, and therefore discovering and managing requirements is a complex process.

Requirements

Requirements for a software system specifies, what services the intended system should provide. It seems, that it should be reasonably straightforward to describe an intended system in terms of the requirements it should meet; but it is more complicated than that. Besides the dynamism and uncertainty mentioned above, expressing requirements is complicated due to the fact that requirements can be expressed at different levels of abstraction, and they can be functional or non-functional requirements. User requirements are typically more abstract than system requirements, as user requirements are natural language statements supplemented by models; system requirements are more detailed descriptions of user requirements. A software design specification is an even more detailed description of a system. Requirements at all levels of abstraction can be either functional, describing the functions of the intended system, or non-functional describing emergent properties, for instance performance requirements such as reliability or response time [6].

From the above, it should be quite clear that a requirement is not just a requirement; it is part of a very complex collection of many different types of requirements, which must be weighed according to each other in order to form a synthesis that can become an integrated part of an organization, both technically and socially. In other words, we claim that the prevailing approach for requirements management within system development may, at first glance, be perceived to be positivistic in nature. Knowledge of social aspects is typically sought from a traditional point of view, where some part of society employs a certain function and work together to promote social stability. The dominance of positivism may result in important issues in regard to the interplay between an organization and an information system being missed.

The Structure of the Thesis

The scientific contribution of this project is divided into three parts. The first part contains a literature survey of requirements management issues that examines how they are discussed and handled in the field of computer science. The survey is based on articles collected from the digital libraries of IEEE and ACM.

In the second part we look upon requirements management from four different philosophical paradigms. The understanding of requirements management is nuanced through the four paradigms, which form the basis for a discussion of two development methods, Scrum and OOA&D. With a fictitious software project in mind, we present a strategy for organizing the development process.

The third part contains a reflection of what knowledge we have obtained and how this knowledge can contribute to the field of *software engineering*. In the end we take our theoretical foundations up for a revision.

References

[1] BOEHM, B., "Get ready for agile methods, with care," *IEEE Computer Society Press.*, vol. 35, no. 1, 2002, pp. 64-69.

[2] BOURQUE, P., DUPUIS, R., ABRAN, A., MOORE, J. W., AND TRIPP, L., *Guide to the Software Engineering Body of Knowledge*, Version 2004, IEEE Press, 2004, 0-7695-2330-7.

[3] POPPENDIECK, M., "Wicked Problems," May 2002, http://www.poppendieck.com/wicked.htm., date: 07-06-06

[4] PRESSMAN, R. S., *Software Engineering - A Practitioner's Approach*, 6th ed., McGraw-Hill, 2004, 0-07-123840-9.

[5] RITTEL, H. AND WEBBER, M., "Dilemmas in a General Theory of Planning," *Policy Sciences*, vol. 4, 1973, pp. 155-169.

[6] SOMMERVILLE, I., *Software Engineering*, 6th ed., Addison Wesley, 2001, 0-201-39815-x.

Contents

PART ONE
A Literature Survey

The first part of the thesis is a research paper presenting findings from a literature survey on how changing and evolving requirements are perceived and handled. The purpose of this survey is to give an overview of the research concerning requirements management issues.

The survey is concerned with the classification and research overview of three main directions in software development: Plan-driven development, agile development, and incremental development. The classification of the requirements topics is from SWEBOK. The classification shows the most dominant requirements topics and the focus of researchers and practitioners. The survey is based on articles from the digital libraries of ACM and IEEE Computer Society and the time span of the articles is of 10 years, from 1995 to 2005. The results show that much attention has been given to specific tools and methods for handling changing and evolving requirements, while much less attention has been paid to why specific tools and methods should be preferred.

The literature survey has also revealed that most of the contributions are concerned with plan-driven approaches for handling changing and evolving requirements. Characteristic for the plan-driven requirements management approach is up-front requirements elicitation, planning, a plan-driven development process, and focus on product instead of process. Only a smaller percentage of the contributions are concerned with agile requirements management. This does not necessarily imply, that requirements are predominantly handled traditionally by means of documentation.

Part one A Literature Survey

The paper gives a status on what has been given attention to in this field of research. We conclude that there is a heavy overweight of articles on requirements management in the plan-driven paradigm.

The initial version of the paper was written on our 9th semester but has been extensively revised during our 10th semester.

Evolving and Changing Requirements: A Literature Survey

Abstract

This paper is a survey of the literature concerning evolving and changing requirements during system development. The main interest is to perceive an overview of how changes are handled in a system development process. The contributions in the field of evolving and changing requirements are divided in three different categories according to their focus: rationale, strategy, and operations. Together these categories provide a holistic view of the phenomenon. The distribution of the articles across the categories gives a picture of the tendencies in the area of evolving and changing requirements. The categorization of the articles is represented in a tree structure, which not only gives an overview, but also illustrates the relation in between the different categories. Our results show that contributions in the field of evolving and changing requirements are predominantly on the strategy and operations levels. The majority of the contributions focus on development and use of tools to simplify the process of development.

Keywords

Evolving and changing requirements, requirements process, plan-driven development, agile development, requirements management.

1. Introduction

Evolving and changing requirements are an essential condition that complicates the process of software development. In this survey the SWEBOK [17] has been chosen as a starting point for gaining insight into what is written about evolving and changing requirements during a system development process.

The SWEBOK is a joint IEEE and ACM project with the aim to give a snapshot of the state in the field of software engineering. The guide is organized into several knowledge areas, which represents a broad consensus regarding what a software engineering professional should know. The SWEBOK sets boundaries of the software engineering discipline and identifies related disciplines, whose material software engineers should also be familiar with [17].

SWEBOK is sponsored by IEEE and constitutes an authoritative source regarding the state in the field of software engineering. SWEBOK's perception and description of evolving and changing requirements is prevailing and widely accepted within the field. As the SWEBOK distinguishes between the software engineering discipline and related disciplines [17], so do we. We focus on the knowledge areas, which a professional software engineer is expected to have.

SWEBOK's view on requirements is that requirements in a particular part of software are a complex combination of requirements from many different sources. The SWEBOK sees requirements as dynamic and conflicting; and the understanding of requirements is never complete nor perfect. It is stated, that the most crucial thing to understand is, that requirements keep changing throughout the process, and the understanding of the requirements also keeps evolving during the process [17].

The following statement expresses the overall view upon change and how to handle change according to the SWEBOK:

> *"Whatever the cause, it is important to recognize the inevitability of change and take steps to mitigate its effects. Change has to be managed by ensuring that proposed changes go through a defined review and approval process, and, by applying careful requirements tracing, impact analysis, and software configuration management."*
>
> ***- SWEBOK [17 ch 2 p 10]***

In the SWEBOK view, changes are inevitable, but there seems to be no interest in why changes occur, and handling changes seems to be a matter of applying techniques and tools for evaluating the consequences of a change request. The human and social issues behind the changing and conflicting requirements are paid very little attention.

The SWEBOK states that requirements elicitation is fundamentally a human activity and not mainly a technical one [17]. It may seem that requirements elicitation is somehow distinguished from e.g. managing requirements, as management of requirements is not mentioned as an important activity and is predominantly addressed as an activity which needs tools and strict techniques. This approach for managing requirements is focused on the request for change itself, and not on the reasons nor conflicts regarding change requests.

The approaches for handling changes during a development process suggest a software world view, which does not consider human and social processes, and interests inherent in the organization as the primary interest. This is thought provoking, as the success of an information system depends upon the humans that make up the organization and the social and political processes present in the organization [84].

The above mentioned has motivated us to survey the literature on evolving and changing requirements. The aim of this survey is to characterize the field and provide a basis for analyzing what the view upon change is, and how change is generally handled.

The remainder of this paper is structured as follows: **Section 2** describes the research method applied for the study. **Section 3** describes a framework for categorizing the field. **Section 4** describes the distribution of the articles found within the categories; rationale, strategy and operations. **Section 5** presents and explains the tree structure for the distribution of articles. **Section 6** presents an analysis of the results, **section 7** presents a short discussion, and **section 8** concludes on the study conducted.

2. Research method

In order to explore the prevailing perception of change as well as methods and tools applied for handling evolving and changing requirements, a search in ACM's and IEEE's article databases has been performed.

ACM's and IEEE's article databases contain contributions primarily written by academic researchers to practitioners and researchers and are well-reputed within this area. The articles in these two databases can give an overview of the most recent case studies of how changing requirements are dealt with, and recommendations on how better to handle changing requirements in practice. These two particular databases have been chosen due to the fact that they contain contributions both aimed at practitioners and researchers, as our objective with this literature survey is to track down, which issues are currently given much attention and which are ignored or less discussed in the literature.

To get inspiration for relevant and up-to-date search phrases and keywords, we have selected phrases and keywords from the SWEBOK. The criteria for search words and phrases has been that they connote a dynamic view of requirements. The search has been performed only on abstracts, and has been limited to searching for articles published during the last ten years (1995-2005) in order to get a feel for the development in and the current state of the area of how to handle changing requirements during a system development process.

The search resulted in a total amount of 268 articles and 145 of them turned out to be irrelevant for this study, as they did not address evolving and changing requirements in any way. This left us with 122 relevant articles, which somehow dealt with evolving and changing requirements.

3. Categorization

In this section we present a categorization framework for characterizing the tendencies in the perception and handling of evolving and changing requirements during the process of software engineering. In order to draw a map of the field we categorize the articles according to whether their primary focus is on rationale, strategy, or operational level. Further, we identify which development paradigm each article belongs to. The categories are all described below.

3.1 Rationale, Strategy, and Operations

Together the three categories; rationale, strategy, and operations, provide a holistic view on some phenomenon. One could think of the three categories in e.g. a military context. **Rationale** is the motivation, reason and context for developing or following a particular **strategy** for actual military **operations**. These three categories of focus and the interaction among the categories can give a nuanced and complex description of some phenomenon. Therefore, the distribution of the articles across the categories can give a hint as to what level of consideration is given more attention in handling evolving and changing requirements.

The three categories in this paper are used as characterizing what the authors' objectives with the contributions are.

- **Rationale:** Contributions put in this category concern the motivation for handling changes during the requirements process. These articles are concerned with the context and reasons for why changing and evolving requirements occur and why they should or should not be considered.
- **Strategy:** Contributions in this category focus on strategies for handling evolving and changing requirements. These contributions outline, who should do what, when and how often in order to conduct particular activities.
- **Operations:** Contributions in this category describe concrete and specific tools or techniques for actually handling evolving and changing requirements.

3.2 Development Paradigms

In addition to the three categories above - rationale, strategy, and operations - we classify the articles according to, which development paradigm they belong to. This is done in order to determine how changing requirements are perceived and how requirements management and change management is handled in each paradigm. This can give a broader view of the field of changing requirements, in the sense that paradigm-specific ways of handling and perceiving change are identified. The three paradigms are classified into: *The plan-driven*, *the incremental*, and *the agile*. A fourth category, paradigm *not-definable*, is used for the articles that cannot clearly be placed in one of the other categories.

3.2.1 The Plan-driven Paradigm

Following the plan-driven development paradigm the development process flows from requirements elicitation through delivery and maintenance in a reasonably linear fashion. It is a systematic and sequential approach [103]. The development process is separated in distinct phases, where each phase

Table 1: Boehm and Turner's polar chart describing the agile and plan-driven home grounds and environmental dimensions [16].

	Agile home ground	Plan-driven home ground
Application		
Primary goals	Rapid value, responding to change	Predictability, stability, high assurance
Size	Smaller teams and projects	Larger teams and projects
Environment	Turbulent, high change, project focused	Stable, low change, project and organization
Management		
Customer relations	Dedicated onsite customers, focused on prioritized requirements	As-needed customer interactions, focused on contract provisions
Planning and control	Internalized plans, qualitative control	Documented plans, quantitative control
Communications	Tacit interpersonal knowledge	Explicit documented knowledge
Technical		
Requirements	Prioritized informal stories and test cases, undergoing unforeseeable change	Formalized project, capability, interface, quality, foreseeable evolution requirements
Development	Simple design, short increments, refactoring assumed inexpensive	Extensive design, longer increments, refactoring assumed expensive
Test	Executable test cases define requirements, testing	Documented test plans and procedures

is ended before a new is started. Requirements are captured in the beginning of the process and are then partly disclosed. Within this paradigm requirements are understood as a means of controlling the product result. The requirements are seen as a product in the development process.

Compared to other development paradigms, we understand the characteristics of the plan-driven development as presented by Boehm and Turner [16] in table 1.

3.2.2 The Incremental Paradigm

The incremental development paradigm combines the phases of the plan-driven model in an iterative fashion [103]. Within this paradigm, requirements are understood as an recurring event with the aim of improving every iteration that occurs. The characteristics of this paradigm are:

- The requirements are improved in increments.
- The process for each increment goes through all the phases as in the plan-driven paradigm, but not necessarily as sequentially.
- Again documentation is central, although not all requirements are defined from the beginning.
- A limited set of functionality is provided to the users rather quickly through prototypes.
- The set of functionality is refined and expanded in each software release, so that each increment provides additional functionality [103].

3.2.3 The Agile Paradigm

The agile paradigm emerged as an alternative way to do software engineering. It differs from the conventional plan-driven paradigm in more ways. In the manifesto for agile software development, it is stated that agile development values

> *"individuals and interactions over processes and tools, working software over comprehensive documentation, customers collaboration over contract negotiation, and responding to change over following a plan"*
>
> ***- Pressman [103 p 103]***

Requirements are understood as an integrated part of the process, which cannot be distinguished from the rest of the process. In comparison with the other paradigms, we understand the characteristics of the agile paradigm as presented by Boehm and Turner [16] in table 1.

3.2.4 Paradigm Not-definable

This category is used when it is not possible to determine, which of the development paradigms a contribution should be placed in, and the contribution is still relevant in the context of evolving and changing requirements. An example of such a contribution is an article describing a tool that could be used in more than one of the development paradigms.

4. Distribution of Articles

In table 1 the distribution of the articles across the categories is illustrated. The articles, which are in parentheses are secondary, and therefore primary in another of the categories rationale, strategy, or operations.

Looking at the distribution of the articles across the table, it is striking, that the majority of the contributions have either strategy or operations as focus under the plan-driven development paradigm. Out of the total 122 articles, there are 77 - over 60 percent - articles primary in these to two fields of the table. Also noteworthy, is the absence of contributions at the rationale level; there are a total of only six primary articles at the rationale level - less than 10 percent.

Table 2: Distribution of papers on levels of focus and development paradigms, respectively.

	Plan-driven	Incremental	Agile	Not-definable
Rationale	5, 24, (67), (77), 84, 89, 114,115	(27), (83)		
Strategy	1, 7, 22, 26, 28, 29, 30, 31, 41, 42, 46, 49, 61, 62, 65, 67, 73, (74), 75, 76, 81, 87, (88), (89), 95, 101, 105, 106, 107, 108, (111), 113, (114), 117, 119, 121	11, 12, 18, 27, 78, 82, 83, 120	35, 36, 37, 47, 93, 125	21, 59, 72, 85, 90
Operations	2, 3, 4, 6, (7), 8, 9, 10, 13, 25, (26), 33, 34, 38, 39, 52, 55, 56, 57, 60, 63, 64, 68, 69, 70, 71, 74, 77, 80, 88, 92, 94, 96, 97, 98, 100, 104, 109, 110, 111, 112, 116, 118, 122, 123, 124, 126, 127	(11), (12), (18), 19, 23, 43, 44, 45, 86, 99	53, 66, 91	14, 32, 40, 48, 50, 51, 54, 58, 79, 102

In summary, we found six papers in the rationale category; 50 papers in the strategy category; and 66 papers in the operations category. In relation to the categories of development paradigms, we found 83 papers under the plan-driven paradigm, 15 under the incremental paradigm, 9 under the agile paradigm, and then we were unable to place 15 papers under a specific paradigm.

To further elaborate on the contents of the articles found from ACM and IEEE, we develop a tree in the next section to illustrate, which topics the articles address, and the relation between the topics at the rationale, strategy, and operations levels.

5. Topic Classification and Distribution

In table 2 it is shown how the articles distribute across the categories. The next step is to elaborate on the contents of the articles. This is done by using a tree structure for each of the development paradigms. Under each of the development paradigms, the articles found are divided into topics of requirements issues. All development paradigms are further divided into levels of focus: Rationale, strategy, and operations. The tree structure is used for visualizing the relation between the topics in each of the categories.

The root of each of the four trees states the development paradigm - plan-driven, incremental, agile, or not-definable, which furthermore, are divided into levels. At the first level, topics concerning rationales, are placed. Topics in the strategy category are at the second level, while topics in the operations category are at the third level. Each paradigm tree is described and illustrated separately below. Some articles span over more than one topic, where we have chosen to place them in the topic, which seems to be most supported.

The description of the trees is divided according to the three foucs levels. Each of the three sections start by giving a general description of the aim with the contributions at the specific level. Then a summary of the contributions under each of the topics is given.

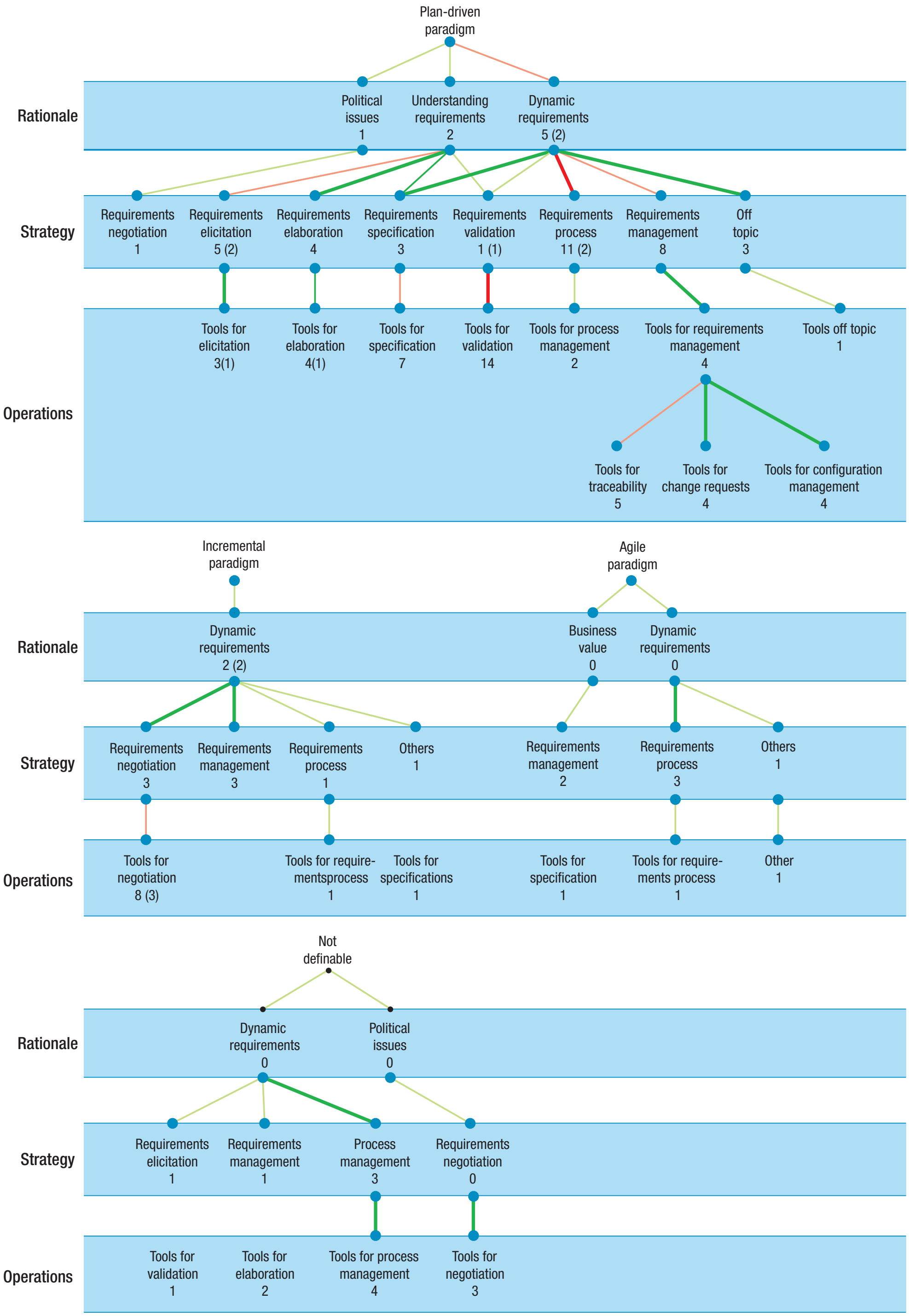

Plan-driven paradigm
Rationale
Political issues 1
Understanding requirements 2
Dynamic requirements 5 (2)
Strategy
Requirements negotiation 1
Requirements elicitation 5 (2)
Requirements elaboration 4
Requirements specification 3
Requirements validation 1 (1)
Requirements process 11 (2)
Requirements management 8
Off topic 3
Operations
Tools for elicitation 3(1)
Tools for elaboration 4(1)
Tools for specification 7
Tools for validation 14
Tools for process management 2
Tools for requirements management 4
Tools off topic 1
Tools for traceability 5
Tools for change requests 4
Tools for configuration management 4
Incremental paradigm
Agile paradigm
Rationale
Dynamic requirements 2 (2)
Business value 0
Dynamic requirements 0
Strategy
Requirements negotiation 3
Requirements management 3
Requirements process 1
Others 1
Requirements management 2
Requirements process 3
Others 1
Operations
Tools for negotiation 8 (3)
Tools for require-mentsprocess 1
Tools for specifications 1
Tools for specification 1
Tools for require-ments process 1
Other 1
Not definable
Rationale
Dynamic requirements 0
Political issues 0
Strategy
Requirements elicitation 1
Requirements management 1
Process management 3
Requirements negotiation 0
Operations
Tools for validation 1
Tools for elaboration 2
Tools for process management 4
Tools for negotiation 3

Figure 1: An illustration of the topic classification and distribution of the articles within the three levels.

0 - 2 papers in node below

3 - 4 papers in node below

5 - 9 papers in node below

10+ papers in node below

If a node is not connected to the upper level, no articles were found concerning its rationale or strategy.

5.1 Rationale Topics

We have identified three different rationale topics from the articles in the rationale categories. The first topic is political issues, the second is understanding requirements, and the third is dynamic requirements.

We only found primary articles at the rationale level under the plan-driven paradigm. And two secondary articles ([83]), ([27]) under the incremental paradigm.

5.1.1 Political Issues

Contributions in this node in the tree concern motivations for and reasons why requirements management should deal with political issues and why political issues are present in systems development.

We have found one article [5] in this node, and it discusses the influence of politics on requirements management and system development. It states that requirements management is a political process, not a technical one. The article also suggests that many projects are poorly thought out and should never have been built in the first place. The reasons why these systems are built, even so, are political. The article proposes questions that should be possible to answer, if the motivation for building a system is valid.

5.1.2 Understanding Requirements

This rationale concerns why it is important to make an extra effort to understand requirements, and make sure that customers/users and developers have a mutual understanding of requirements during the requirements process.

We found two articles ([67]), [24] in this node, concerning why it is beneficial that developers and customers/users achieve better mutual understanding of requirements, and why users should be more involved in the requirements management process.

5.1.3 Dynamic Requirements

This rationale concerns the motivation for why requirements management should deal with dynamic requirements, and why requirements are dynamic.

We have found five articles in this node [89], [114], ([77]), [84], [115]. New requirements and changes in requirements are understood as an essential prerequisite for system engineering in the articles. One article examines how the maturity of a requirements management process influences the requirements process in terms of experienced problems. Another article aims to gain a more holistic understanding of the requirements process, in order to make the companies organize and manage requirements more effectively. One article examines reasons for requirements volatility in an ISO 9001 certified company.

The last article [115] studies how requirements instability (more specifically pre and post release changes) influence on defects in the software.

5.2 Strategy Topics

At the strategy level the articles have been grouped according to the following topics: Requirements negotiation, requirements elicitation, requirements elaboration, requirements specification, requirements validation, requirements process, requirements management and off topic. Our use of these topics and their description are inspired by Pressman's description of the tasks in requirements engineering [103].

5.2.1 Requirements Negotiation

Requirements negotiation is needed when different stakeholders express conflicting requirements. It is basically about resolving conflicting requirements, which typically arise when stakeholders require mutually incompatible features. This can be during requirements elicitation, elaboration and during the process of handling change requests.

Not many articles concern requirements negotiation; [18], [12], [11], and [30]. The first three fall under the incremental paradigm, and the last contribution belongs under the plan-driven paradigm. The papers concern distributed requirements negotiation and experiences with systems like WinWin and GSS (Group Support Systems).

5.2.2 Requirements Elicitation

Requirements elicitation is the process of discovering the requirements for a particular piece of software. In the process of elicitation the origin of the requirements are denoted. The actors involved in the process of elicitation and their understanding are considered. The elicitation part is closely related to

the analytical part of requirements management.
The articles all fall under the plan-driven tree, except the last one that are in the not-definable tree; ([74]), ([88]), [95], [121], [7], and [90]. One of the papers' main concerns is to integrate creativity, using scenarios and aiming to automate elicitation of functional requirements.

5.2.3 Requirements Elaboration

Requirements elaboration focuses on developing a refined technical model of software functions, features, and constraints. It has the purpose of making the transition from user requirements to system requirements as smooth as possible [103].
All contributions concerning requirements elaboration are placed under the plan-driven paradigm; [67], [22], [26], and [106]. One addresses how to reduce complexity in the translation of natural language requirements to formal language; the two next concern how to incorporate business strategies into the requirements specification; the last paper concerns a suggestion to change the actors assignments. For instance, the developer should have the opportunity to write the requirements down as it gives a better understanding.

5.2.4 Requirements Specification

The requirements specification is the documentation of the requirements that the software engineers and stakeholders have agreed upon. A specification can be written in documents, sets of graphical models, formal mathematical models, scenarios etc. The specification establishes the basis for agreement between customers and contractors and can provide a realistic basis for estimating product costs, risks and schedules. The specifications are often written in natural language but are typically supplemented with formal and semi-formal descriptions [103].
Articles regarding specifications are only found under the plan-driven tree [1], [49], [113]. Suggestions to replace documents with databases are given as well as suggestions to define requirements at different levels of abstractions.

5.2.5 Requirements Validation

Requirements validation is an activity performed to be able to confirm that there are not any omissions, conflicts and ambiguities in the requirements [103].

Most of the articles concerning validation are placed at the operations level as there are concrete suggestions on how to validate requirements. There is only one secondary paper at the strategy level, which is in the plan-driven tree. It proposes a mathematical approach for finding conflicts in requirements. It is about Viewpoint-Based Requirements Engineering.

5.2.6 Requirements Process

The requirements process includes all the activities connected to developing and handling requirements: Negotiation, elicitation, elaboration, specification, validation etc. It is the overall view of how all the above mentioned parts are connected. This category is included as some articles refer to the entire requirements process.

Most of the articles in this node are placed in the plan-driven tree ([89]), ([114]), [108], [87], [29], [101], [107], [117], [28], [41], [42]. The articles within this tree concerns improvements regarding SPI, SMM and SPICE and improvements in sense of integrating creativity. The three articles in the agile tree are mostly about how XP should be integrated and what experiences one has [36], [35], [125]. The one paper in the incremental tree, [27], presents a strategy for integrating an agile development lifecycle instead of the plan-driven development lifecycle at ABB. The rest of the articles, which are in the paradigm not-definable tree, are case studies to clarify what problems are occurring in the requirements process [59], [21], [85].

5.2.7 Requirements Management

Requirements management is a set of activities, which helps the project team to identify, control and track requirements and changes to requirements during the development process. The steps within management are identification of requirements and development of traceability tables. Traceability tables relate requirements to one or more aspects of the system or its environment. We have chosen to also place articles about software configuration management and management of change requests in this topic, as these activities also concern management of requirements.

The majority of the articles within this topic are placed in the plan-driven tree, namely eight of the total 14; [76], [65], [75], [105], [81], [46], [61], [31]. Then there are two in the agile tree, [37], [93]; three in the incremental tree,

[83], [120], [78]; and finally one paper in the paradigm not-defineable tree; [72]. The papers within all the trees somehow concern difficulties managing changes and experiences regarding requirements management.

5.2.8 Off Topic and others

Off topic contains articles that are somehow relevant to the topic of evolving and changing requirements within the plan-driven development paradigm, but are not possible to place under one of the other strategy topics.
Examples of topics of the articles, that do not fall under any of the other topics, are software maintenance, evolving software etc. with no direct connection to evolving and changing requirements are placed here; [62], [119], [73], [47], [82].

5.3 Operations Topics

At the operations level articles are grouped according to the following topics: tools for elicitation, tools for elaboration, tools for specification, tools for validation, tools for requirements process (management), and tools for requirements management, which are further specified into tools for traceability, tools for change requests and tools for configuration management. Finally, there is a node for tools that belong to the plan-driven development paradigm, but cannot be placed in any of the other operations topics.

5.3.1 Tools for Elicitation

Tools for elicitation are tools for discovering and capturing the stakeholders' requirements. The tools, which are described in the three articles; [74], [88], ([7]), are for elicitating system and functional requirements. No articles within tools for elicitation are placed in the agile, incremental or paradigm not-definable tree.

5.3.2 Tools for Elaboration

Tools for elaboration are tools used to help transform user requirements to system requirements. There are four primary and one secondary paper in this topic. The three primary and the secondary are in the plan-driven tree; [98], [80], ([21]), [124]; the last two articles are located in the paradigm not-definable tree; [58], [54]. Most of the articles under this topic concentrate on simplifying requirements or transforming the requirements from natural language to formal language.

5.3.3 Tools for Negotiation

Tools for negotiation are tools that help to resolve conflicting requirements, which typically arise when stakeholders require mutually incompatible features. Nine of the articles in tools for negotiation have been placed in the incremental tree; [15], [44], [43], [99], [19], [45], ([18]), ([12]). Three articles were placed in the paradigm not-definable tree; [48], [50], [51]. The majority of the contributions concern the use of WinWin in different versions. The articles are more of the descriptive form, where the software functionality is in focus rather than the social issues regarding negotiation.

5.3.4 Tools for Specification

Tools for specification are tools to simplify the process of writing. Some of the tools described in the articles present variations of languages to formulate the requirements. For instance the process of converting requirements written in natural language to a more formal language can be challenging as some of the requirements are ambiguous and cannot be translated directly and correctly. There are nine articles in this topic. Seven are in the plan-driven tree; [55], [34], [13], [122], [96], [94], [3].

Solutions on how to handle the transformation from natural language to a more formal language is a popular subject within the articles. However one article is dedicated to the subject of inconsistent requirements and how to handle them. As earlier mentioned the agile method usually do not use specifications, however, there is one article on how to link project assets to and interactive specification in the agile environment [53]. The last article of this topic is in the incremental tree; [23], and concerns a specification language which is used for writing down user requirements.

5.3.5 Tools for Validation

Tools for validation are tools which can assure that the dependencies between requirements are correct. The process of validation can be of the technical kind as well as the kind where users are involved. Once again the majority of the articles are placed in the plan-driven tree, 14 out of 15; [116], [111], [33],

[4], [68], [71], [69], [38], [8], [6], [60], [97], [2], [39]. Changes in requirements are generally percieved as small disruptions, which can be handled either by calculating conflicts beforehand or in general by using formal methods.
The last contribution is in the paradigm not-defineable tree; [79]. It discusses a concept for a groupware supported requirements inspection process.

5.3.6 Tools for Process Management

Tools for process management are tools which can help simplify and organize the management of the entire requirements process. We found eight articles under this topic. We placed two articles under the plan-driven development paradigm; [77], [127], one under the agile paradigm; [91],one under the incremental paradigm, [86]; and four articles were placed in the paradigm not defineable tree; [102], [14], [32], [40]. Both in the plan-driven and paradigm not-definable tree we find articles, which present questions for the organization to answer in order to evaluate and improve the process management. Only few of the articles give concrete model solutions to managing the process, the rest of the articles concern what experiences have been gained and how to integrate the lessons learned into a development process.

5.3.7 Tools for Requirements Management

Tools for requirements management are tools that can manage to identify, control and track requirements and changes to requirements. The articles found are placed in the plan-driven tree and have a technical view upon requirements management; [9], [10], [70], [104]. Descriptions and lessons learned from PARSNIP, CARET and CASAPS (tools) are the topic within requirements management.

5.3.8 Tools for Traceability

Tools for traceability can help to identify the relation between the requirements and its environments. Traceability is only a topic in the plan-driven tree. The content of the articles are concerned with concrete techniques to support changes in software systems. More specifically they concern how to find equality between what has been acquired in natural language and transformation to object models; [56], [118], [57], [100], [112].

5.3.9 Tools for Change Requests

Tools for change requests are tools, which handles the requests for changes in different stages of the development. Change requests are changes in requirements as well as changes in the developed code. We found four articles concerning change requests under the plan-driven paradigm; [63], [92], [123], [52]. The articles each concern a different method for handling change requests.

5.3.10 Tools for Configuration Management

Tools for configuration management are tools, which can manage the different configurations of a system during its life cycle. It can systematically control changes to the configuration and maintenance of the integrity and traceability through the software life cycle. We found four articles concerning configuration management and placed them all under the plan-driven paradigm; [64], [109], [110], [25].

6. Analysis

The tree structure, figure 1, together with the descriptions of the topics addressed by the authors indicate that the reasons for evolving and changing requirements, and the need for handling evolving and changing requirements is not an area, which has been given much attention. From the articles acquired, it is difficult to identify a view upon change, as the view upon change is only mentioned in less than ten percent of the articles placed under the rationale category. Instead, the authors generally focus on solutions, in terms of guidelines, methods, tools and techniques, for handling evolving and changing requirements.

The SWEBOK does not seem to have any interest in, why changes occur, and pays very little attention to social issues related to change. The lack of interest in why changes occur is not quite as distinct in the articles, as a few of the rationale articles discuss reasons for requirements volatility. Also some of the rationale articles put focus on the social aspect of system development. As figure 1 and table 2 show, the plan-driven tree is the heaviest. The fact that we found most articles under the plan-driven development paradigm suggests a traditional development philosophy, that considers documentation as fundamental.

The plan-driven tree reveals that authors place special attention on tools for requirements validation, the requirements process, and requirements management in general. These topics are critical in controlling requirements during a system development process, and perhaps for that reason, especially the contributions in the requirements process and requirements management nodes are concerned with structuring and standardizing the requirements process. Activities concerning organizational or social issues, for instance elicitation, negotiation, and user validation, are either ignored, discussed briefly, or addressed with concrete tools or techniques.

We only placed a small number of articles in the agile development tree; and none were located on the rationale level. The small number of contributions in this tree, may be due to the fact that, in agile development changes in requirements and priorities are described as a natural part of the process. The process is not divided into smaller activities like requirements negotiation, elaboration, validation, etc. These activities are embedded in the process. As changes are considered natural and always occurring in agile development, there is no need to discuss them, as they do not cause the same problems as they do in a traditional development process.

The incremental tree is similar to the agile, size-wise. It is noticeable, however, that there is an overweight at the operational level in the negotiation branch. Most of the articles in this node concern WinWin, an approach for stakeholders to reason about inconsistency in requirements.

To comment on the general picture that table 2 and the figure 1 draw of changing requirements, it seems that focus is on addressing the changing requirements with different methods, techniques, or tools depending on the stage in the development process. The focus is on the information system itself and not the process of developing the system.

7. Discussion

The description of the contents of the articles according to the topics identified revealed that very few of the articles had their primary focus on requirements. Many articles contained a description of new software to support work internal to the developer organization. Generally, there seems to be a conviction, that organizational and social issues should be solved with technology.

Handling changes can be understood on many levels. We could be talking about handling organizational changes, changes from the customer or changes within the software. We were surprised to find more articles on how to manage changes in software than in the organization or changes from the customer. It seemed that developing software and the organization, in which the software is to be used, are seen as two separated parts, which concern us.

With the categorization and the tree in mind, we suggest that more attention should be paid on the underlying reasons for why changes are handled as they are, and what consequences these approaches have. We believe it is important to be aware why particular methods are applied and what consequences they have. What concerns us, is the lack of attention towards social and organizational issues during the development process, as these can have critical impact on the process and therefore also the product.

Apparently Brooks' [20] article *No Silver Bullet* confirms our concern on the view upon software. Our results showed that changes are still seen as occurring accidents and not as a natural part of the development. Brooks emphasizes the need for seeing software as complex and not as accidents.

8. Conclusion

The categories rationale, strategy, and operations have been used for characterizing the authors' focus and main objectives with their contributions. As it turns out, we found seven articles in the rationale category, indicating that the reasons for evolving and changing requirements, and the need for handling the requirements is not an area, which is given much attention. Instead the authors focus on solutions, in terms of guidelines, methods, tools, and techniques, for handling evolving and changing requirements. This can be seen from the fact, that we found 50 articles which are concerned with strategies, that are somehow related to how requirements could be handled during a development process. Further, we found that the focus of 66 articles is specific tools and techniques for handling changes.

When looking at the distribution of the articles according to the development paradigms, we found that the majority of the articles fall under the plan-driven development paradigm - 83 out of 122 articles. We found notably fewer articles under both the incremental and the agile development paradigms - 15 under the incremental paradigm, and 9 under the agile development paradigm. 15 articles are placed under the paradigm not-definable, as we were unable to decide upon a specific development paradigm; some tools and methods could be used under more than one development paradigm.

To sum op, the categorization illustrates, that the majority of the articles fall under the plan-driven development paradigm and are concerned with solutions - strategic or operational - to the problem of managing evolving and changing requirements. Few are concerned with organizational and social issues related to changing requirements, and we call for more research in this area, as social and organizational issues should be seen as an integrated part of system development and not something, which is separable from each other.

References

[1] HIGGINS, A., DE LAAT, M., GIELES, P. M. C., and GEURTS, E. M., "Managing product requirements for medical IT products," *Proceedings of the IEEE Joint International Conference on Requirements Engineering* (RE'02), IEEE Software, vol. 20, no. 1, 2003, pp. 26-33.

[2] D'AVILA GARCEZ, A.S., RUSSO, A., NUSEIBEH, B., KRAMER, J., "Combining abductive reasoning and inductive learning to evolve requirements specifications," *IEE Proceedings Software*, vol. 150, no. 1, 2003, pp. 25-38.

[3] AL-ANI, B., and EDWARDS, K., "An empirical study of a qualitative systematic approach to requirements analysis (QSARA)," *Proceedings of the 2004 International Symposium on Empirical Software Engineering* (ISESE'04), IEEE, 2004, pp. 177-186.

[4] DURAN, A., RUIZ-CORTEZ, A., CORCHUELO, R., TORO, M., "Supporting requirements verification using XSLT," *Proceedings of the IEEE Joint International Conference in Requirements Engineering* (ICRE'02), IEEE, 2002, pp. 165-172.

[5] ANDRIOLE, S., "The politics of requirements management," *IEEE Software*, vol. 15, no. 6, 1998, pp. 82-84.

[6] POS, A., Akkermans, H., and TOP, J., "Automatic revision of simulation models," *IEEE*, vol. 13, no. 2, 1998, pp. 75-81.

[7] ANKORI, R., "Automatic requirements elicitation in agile processes," *Proceedings of the IEEE International Conference on Software - Science, Technology and Engineering* (SWSTE'05), IEEE, 2005, pp. 101-109.

[8] ANTONIOU, G., and WILLIAMS M.-A., "Revising default theories," *Proceedings of the 10th IEEE International Conference on Tools with Artificial Intelligence* (TAI'98), IEEE, 1998, pp. 423-430.

[9] BABAR, M. A., "An experience of applying architecture-based approach to build a component-based requirements management toolset," *Proceedings of the 27th Annual International Computer Software and Applications Conference* (COMPSAC'03), IEEE, 2003, pp. 550-556.

[10] BABAR, M. A., and ZOWGHI, D., "Developing a requirements management toolset: Lessons learned," *Proceedings of the 2004 Australian Software Engineering Conference* (ASWEC'04), IEEE, 2004, pp. 10-19.

[11] BOEHM, B., ABI-ANTOUN, M., PORT, D., KWAN, J., and Lynch, A., "Requirements engineering, expectations management, and the Two Cultures," *Proceedings of the IEEE International Symposium on Requirements Engineering* (ISRE'99), IEEE, 1999, pp. 14-22.

[12] BOEHM, B., GRUNBACHER, P., and BRIGGS, R.O., "Developing groupware for requirements negotiation: Lessons learned," *IEEE Software*, vol. 18, no. 3, 2001, pp. 46-55.

[13] NUSEIBEH, B., EASTERBROOK, S., and RUSSO, A., "Leveraging inconsistency in software development," *IEEE Computer Society*, vol. 33, no. 4, 2000, pp. 24-29.

[14] WESTFECHTEL, B., MUNCH, B.P., and CONRADI, R., "A layered architecture for uniform version management," *IEEE Transactions on Software Engineering* (TSE'01), IEEE Computer Society, vol. 27, no. 12, 2001, pp. 1111-1133.

[15] BOEHM, B., and EGYED, A., "Software requirements negotiation: Some lessons Learned," *Proceedings of the 20th International Conference on Software Engineering* (ICSE'98), IEEE, 1998, pp. 503-506.

[16] BOEHM, B., and TURNER, R., "Using risk to balance agile and plan- driven methods," *IEEE Computer Society*, vol. 36, no. 6, 2003, pp. 57-66.

[17] BOURQUE, P., DUPUIS, R., ABRAN, A., MOORE, J. W., and TRIPP, L., "The Guide to the Software Engineering Body of Knowledge," *IEEE Software*, vol. 16, no. 6, pp. 34-44.

[18] BRIGGS, R. O., and GRUENBACHER, P., "Easywinwin: managing complexity in requirements negotiation with GSS," *Proceedings of the 35th Hawaii International Conference in System Sciences* (HICSS'02), IEEE, 2002, pp. 10.

[19] BRIGGS, R.O., ZHAO, J.L., and NUNAMAKER, J.F., "Intelligent workflow techniques for distributed group facilitation," *Proceedings of the 35th Annual Hawaii International Conference* (HICSS'02), IEEE, 2002, pp. 597-605.

[20] BROOKS, F. P., "No silver bullet: Essence and accidents of software engineering," *Computer Magazine*, April 1987, pp.10-19.

[21] CARLSHAMRE, P., and REGNELL, B., "Requirements lifecycle management and release planning in market-driven requirements engineering processes," *Proceedings on the 11th International Workshop on Database and Expert Systems Applications* (DEXA'00), IEEE, 2000, pp. 961–965.

[22] CHAMPION, R. E. M., and MOORES, T. T., "Exploiting an enterprise model during systems' requirements capture and analysis," *Proceedings of the 2nd International Conference on Requirements Engineering* (ICRE'96), IEEE, 1996, pp. 208–215.

[23] CHIANG, C.-C., and URBAN, J. E., "Incremental elicitation and formalization of user requirements through rapid prototyping via software transformations," *Proceedings of the 20th International Computer Software and Applications Conference* (COMPSAC'96), IEEE, 1996, pp. 240–245.

[24] CLAVADETSCHER, C., "User involvement key to success," *IEEE Software*, vol. 15, no: 2, 1998, pp. 30-32.

[25] CRNKOVIC, I., "A change process model in an SCM tool," *Proceedings of the 24th Euromicro Conference* (EUROMICRO'97), IEEE, vol. 2, 1998, pp. 794-799.

[26] CYBULSKI, J. L. and REED, K., “Computer-assisted analysis and refinement of informal software requirements documents,” *Proceedings of the Asia Pacific Software Engineering Conference* (APSEC), IEEE, 1998, pp. 128-135.

[27] DAGNINO, A., “An evolutionary lifecycle model with agile practices for software development at ABB.,“ *Proceedings of the 8th IEEE International Conference on Engineering of Complex Computer Systems* (CECCS’20), IEEE, 2002, pp. 215-223.

[28] DAMIAN, D., ZOWGHI, D., VAIDYANATHASAMY, L., and Pal, Y., “An industrial experience in process improvement: An early assessment at the australian center for Unisys software,” *Proceedings of the 2002 International Symposium on Empirical Software Engineering* (ISESE’02), IEEE, 2002, pp. 111-123.

[29] DAMIAN, D., CHISAN, J., VAIDYANATHASAMY, L., and Pal, Y., “An industrial case study of the impact of requirements engineering on downstream development,” *Proceedings of the 2002 International Symposium on Empirical Software Engineering* (ISESE’02), IEEE, 2003, pp. 40-49.

[30] HERLEA DAMIAN, D.E., EBERLEIN, A., SHAW, M.L.G., and GAINES, B.R., “The effects of communication media on group performance in requirements engineering,” *Proceedings of the 4th International Conference on Requirements Engineering* (ICRE’00), IEEE, 2000, p. 191.

[31] DIETEL, K., “Mastering it change management step two: moving from ignorant anarchy to informed anarchy,” *Proceedings of the 32nd annual ACM SIGUCCS conference on User services* (SIGUCCS’04), ACM Press, 2004, pp. 188-190.

[32] DOERNEMANN, H., “Tool-based risk management made practical,” *Proceedings of the IEEE Joint International Conference on Requirements Engineering* (ICRE’02), IEEE, 2002, p.192.

[33] EASTERBROOK, S., and CHECHIK, M., “A framework for multi-valued reasoning over inconsistent viewpoints,” *Proceedings of the 23rd International Conference on Software Engineering* (ICSE’01), IEEE, 2001, pp. 411–420.

[34] PEREZ-MINANA, E., KRUSE, P., AMERICA, P., “Empowering requirements for a product family,” Proceedings of the 5th IEEE International Symposium on Requirements Engineering (SRE’01), IEEE, 2001, pp. 302-303.

[35] English, A., “Extreme programming, it’s worth a look,” *IT Professional*, vol. 4, no. 3, 2002, pp. 48-50.

[36] GROSSMAN, F., BERGIN, J., LEIP, D., MERRIT, S., and Gotel, O., “One XP experience: Introducing agile (XP) software development into a culture that is willing but not ready,” *Proceedings of the 2004 conference on the Centre for Advanced Studies on Collaborative research* (CAS’04), IBM Press, 2004, pp. 242-254.

[37] FAVARO, J., “Managing requirements for business value,” *IEEE Software*, vol. 19, no. 2, 2002, pp. 15-17.

[38] GARCIA-DUGUE, J., PAROZ-ARIAS, J., and BARRAGANS-MARTINEZ, B., “An analysis-revision cycle to evolve requirements specifications by using the SCTL-MUS methodology,” *10th Anniversary Joint IEEE International Requirements Engineering Conference* (RE’02), IEEE, 2002, p. 282

[39] GHOSE, A. K., “A formal basis for consistency, evolution and rationale management in requirements engineering,” *Proceedings of the 11th IEEE International Conference on Tools with Artificial Intelligence* (ICTAI’99), IEEE, 1999, pp. 77-84.

[40] Cuevas, G., SERRANO, A., and SERRANO, A., “Assessment of the requirements management process using a two-stage questionnaire,” *Proceedings of the Fourth International Conference on Quality Software* (QSIC’04), IEEE, 2004, pp. 110-116.

[41] GRIES, M. J., “Systems engineering for the 777 autopilot flight director system,” *14th Digital Avionics Systems Conference* (DASC’95), IEEE, 1995, pp. 403-409.

[42] GRIES, M. J., “System engineering for the 777 autopilot system,” *IEEE Transactions on Aerospace and Electronic Systems* (AES’97), IEEE, vol 33, no. 2, 1997, pp. 649–655.

[43] GRÜNBACHER, P., “Integrating groupware and case capabilities for improving stakeholder involvement in requirements engineering,” *Proceedings of the 26th Euromicro Conference* (EUROMICRO’00), IEEE, 2000, pp. 232-239.

[44] GRÜNBACHER, P., and BRIGGS, R. O. “Surfacing tacit knowledge in requirements negotiation: experiences using EasyWinWin,” *Proceedings of the 34th Annual Hawaii International Conference on System Sciences*, IEEE, 2001, pp 1-8.

[45] RUHE, G., EBERLEIN, A., and PFAHL, D. “Quantitative WinWin - a new method for decision support in requirements negotiation. *Proceedings of the 14th international conference on Software engineering and knowledge engineering* (SEKE’02) (2002), ACM Press, pp. 159-166.

[46] Heller, G., and Vollmer, P., “Requirements based testing at HP OpenView, ”*Proceedings of the 11th IEEE International Requirements Engineering Conference*, IEEE, 2003, p. 299.

[47] HIRSCH, M., “Moving from a plan driven culture to agile development,” Proceedings of the 27th international conference on Software engineering (ICSE’05), ACM Press, 2005, p. 38.

[48] HOH, I., OLSON, D., and RODGERS, T., “A requirements negotiation model based on multi - criteria analysis,” *Proceedings of the 5th IEEE International Symposium on Requirements Engineering* (SRE’01), IEEE, 2001, pp. 312-313.

[49] HUNT, L. B., “Getting the requirements right - a professional approach,” *Proceedings of the 8th International Workshop on Software Technology and Engineering Practice* (STEP’97) (including CASE ‘97), IEEE, 1997, pp. 464-472.

[50] HOH, I., and SIDDHARTHA, R., “Issues of visualized conflict resolution,” *Proceedings of the 5th IEEE International Symposium on Requirements Engineering* (COMPSAC’01), IEEE, 2001, pp. 314-315.

[51] HOH, I., and SIDDHARTHA, R. “Visualization issues for software requirements negotiation,” *Proceedings of the 25th Annual International Computer Software and Applications Conference* (COMPSAC’01), IEEE, 2001, pp. 10-15.

[52] CRNKOVIC, I. FUNK, P., and LARSSON, M., “Processing requirements by software configuration management,” *Proceedings of the 25th Euromicro Conference* (EUROMICRO’99), IEEE, 1999, pp. 260-265.

[53] HAKIM, J., SPITZER, T., and ARMITAGE, J., “Sprint: Agile specifications in shockwave and flash,” *Proceedings of the 2003 conference on Designing for user experiences* (DUX’03), ACM Press, 2003, pp. 1-14.

[54] CASTRO, J. F., MYLOPOULOS J, ALENCAR, F. M. R., GILBERTO, A., AND CYSNEIROS, F., “Integrating organizational requirements and object oriented modelling,” *Proceedings of the 5th IEEE International Symposium on Requirements Engineering* (SRE’01), IEEE, 2001, pp. 146–153.

[55] JAMES, L., “Automatic requirements specification update processing from a requirements management tool perspective,” *Proceedings of the International Conference and Workshop on Engineering of Computer-Based Systems* (ECBS ‘97), IEEE, 1997, pp. 2–9.

[56] CLELAND-HUANG, J., CHANG, C. K., and Ge, Y., “Supporting event based traceability through high - level recognition of change events,” *Proceedings of the 26th Annual International Computer Software and Applications Conference* (COMSAC’02), IEEE, 2002, pp. 595–600.

[57] CLELAND-HUANG, J., ZEMONT, G., and LUKASIK, W., "A heterogeneous solution for improving the return on investment of requirements traceability," *Proceedings of the 12th IEEE International Requirements Engineering Conference* (RE'04), IEEE, 2004, pp. 230–239.

[58] SWAN, J., KUTAR, M., BARKER, T., and BRITTON, C., "User preference and performance with UML interaction diagrams," *IEEE Symposium on Visual Languages and Human Centric Computing* (VLHCC'04), IEEE, 2004, pp. 243-250.

[59] EDWARDS, J., MILLEA, T., MCLEOD, S., and Coutts, I., "Agile system design and build," IEE Colloguium on Managing Requirements Change: A Business Process Re-Engineering Perspective, *IEE*, 1998, pp. 1–7.

[60] GARCIA-DUQUE, J., PAZOS-ARIAS, J.J., and BARRAGANS-MARTINEZ, B., "An analysis-revision cycle to evolve requirements specifications by using the SCTL-MUS methodology," *Proceedings of the IEEE Joint International Conference on Requirements Engineering* (RE'02), IEEE, 2002, pp. 282–288.

[61] KAUTZ, K., "Making sense of measurement for small organizations," *IEEE Software*, vol. 16, no. 2, 1999, pp. 14–20.

[62] KILPI, T., "Improving software product management process: Implementation of a product support system," *Proceedings of the 31st Annual Hawaii International Conference on System Sciences* (HICSS'98), IEEE, 1998, pp. 3–12.

[63] KOBAYASHI, A., and MEAKAWA, M., "Need-based requirements change management," *Proceedings of the 8th Annual IEEE International Conference and Workshop on the Engineering of Computer Based Systems* (ECBS '01), IEEE, 2001, pp. 171–178.

[64] Choi K. J., Kim, K. B. and Jin, S. I., "A Modeling Method of Software Configuration Change Control," *IEEE Pacific Rim Conference on Communications, Computers and Signal Processing* (PACRIM'97), IEEE, vol.1, 1997, pp. 430-433.

[65] LAVAZZA, L., and VALETTO, G., "Enhancing requirements and change management through process modelling and measurement," *Proceedings of the 4th International Conference on Requirements Engineering* (ICRE'00), IEEE, 2000, pp. 106–115.

[66] LAW, A., and CHARRON, R., "Effects of agile practices on social factors," *Proceedings of the 2005 workshop on Human and Social factors of software engineering*, ACM PRESS, 2005, pp. 1–5.

[67] LAWRENCE, B., "Designers must do the modelling," *IEEE Software*, vol. 15, no. 2, 1998, p. 31, p. 33.

[68] LEE, S. W., "Proxy viewpoints model-based requirements engineering," *Proceedings of the 2002 ACM symposium on Applied computing* (SAC'02), ACM Press, 2002, pp. 1004-1008.

[69] LEE, S. W., and RINE, D. C., "Missing requirements and relationship discovery through proxy viewpoints model," *ACM Symposium on Applied Computing* (SAC'04), ACM Press, 2004, pp. 1513–1518.

[70] LEVER, J. A., and ELKINS, M., "Requirements management for the oceanographic information system at the naval oceanographic office," *IEEE*, vol. 1, 2003, pp. 141–144.

[71] LIU, X. F., and YEN, J., "An analytic framework for specifying and analyzing imprecise requirements," *Proceedings of the 18th International Conference on Software Engineering* (ICSE'96), IEEE, 1996, pp. 60–69.

[72] LOCONSOLE, A., "Empirical studies on requirement management measures," *Proceedings of the 26th International Conference on Software Engineering* (ICSE'04), IEEE, 2004, pp. 42–44.

[73] LÖWE, M., "Formal methods," *1st Euromicro Conference on Software maintenance and Reengineering* (EUROMICRO'97), IEEE, 1997, p. 43.

[74] MAIDEN, N., and ROBERTSON, S., "Developing use cases and scenarios in the requirements process," *Proceedings of the 27th international conference on Software engineering* (ICSE'05), IEEE, 2005, pp. 561–570.

[75] LORMANS, M., VAN DIJK, H., and VAN DEURSEN, A., "Managing evolving requirements in an outsourcing context: An industrial experience report," *Proceedings of the 7th International Workshop on Principles of Software Evolution* (IWPSE'04), IEEE, 2004, pp. 149–158.

[76] HOFFMANN, M., KÜHN, N., and BITTNER, M., "Requirements for requirements management tools," *Proceedings of the 12th IEEE International Requirements Engineering Conference* (RE'04), IEEE, 2004, pp. 301–308.

[77] MAY, G., and OULD, M., "Software project casualty," *Engineering Management Journal*, vol.12, no.2, 2002, pp. 83–90.

[78] MCGOVERN, F., "Managing software projects with business-based requirements," *IT Professional*, vol. 4, no. 5. 2002, pp. 18–23.

[79] HALLING, M., GRÜNBARCHER, P., and BIFFL, S., "Tailoring a cots group support system for software requirements inspection," *16th Annual International Conference on Automated Software Engineering* (ASE'01), IEEE, 2001, pp. 201–208.

[80] Beeck, V. D. M., Braun, P., RAPPL, M., and SCHRÖDER, C., "Model based requirements engineering for embedded software," *Proceedings of the IEEE Joint International Conference on Requirements Engineering* (RE'02), IEEE, 2002, p. 92.

[81] WINOKUR, M., GRINMAN, A., YOSHA, I., and GALLANT, R., "Measuring the effectiveness of introducing new methods in the software development process," *Proceedings of the 24th Euromicro Conference* (EURMIC'98), IEEE, vol. 2, 1998, pp. 800–807.

[82] MIKKONEN, T., and PRUUDEN, P., "Practical perspectives on software architectures, high-level design, and evolution," *Proceedings of the 4th International Workshop on Principles of Software Evolution* (IWPSE '01), ACM Press, 2001, pp. 122–125.

[83] MOHAGHEGHI, P., and CONRADI, R., "An empirical study of software change: Origin, acceptance rate, and functionality vs. quality attributes," *Proceedings of the 2004 International Symposium on Empirical Software Engineering* (ISESE'04), IEEE, 2004, pp. 7–16.

[84] NURMULIANI, N., ZOWGI, D., and FOWELL, S., "Analysis of requirements volatility during software development life cycle," *Proceedings of 2004 Australian Software Engineering Conference* (ASWEC'04), 2004, pp. 28–37.

[85] JURISTO, N., MORENO, A. M., and SILVA, A., "Is the european industry moving toward solving requirements engineering problems," *IEEE Software,* vol. 19, no. 6, 2002, pp. 70–77.

[86] NATH, I., and KUNDU, S. "Is software requirements a corporate asset?," *Proceedings of the 2004 IEEE International Engineering Management Conference* (ICMC'04), IEEE, vol. 1, 2004, pp. 244–247.

[87] MAIDEN, N., GIZIKIS, A., and ROBERTSON, S., "Provoking creativity: Imagine what your requirements could be like," *IEEE Software*, vol. 21, no. 5, 2004, pp. 68–75.

[88] MAIDEN, N., MANNING, S., ROBERTSON, S., and GREENWOOD, J., "Integrating creativity workshops into structured requirements processes," *Proceedings of the 2004 conference on Designing interactive systems: processes, practices, methods, and techniques* (DIS'04), ACM Press, 2004, pp. 113–122.

[89] NIAZI, M., and SHASTRY, S., "Role of requirements engineering in software development process: An empirical study," *Proceedings of the 7th International Multi Topic Conference* (INMIC'03), IEEE, 2003, pp. 402–407.

[90] MILLARD, LYNCH, P., and TRACEY, K., “Child’s play: using techniques developed to elicit requirements from children with adults,” *Proceedings of the 3rd International Conference on Requirements Engineering* (ICRE’98), IEEE , 1998, pp. 66-73.

[91] NISAR, M. F., and HAMEED, T., “Agile methods handling off shore software development Issues,” *Proceedings of the 8th International Multitopic Conference* (INMIC’04), IEEE, 2004, pp.417–422.

[92] NURMULIANI, N., ZOWGHI, D., and WILLIAMS, S. P., “Using card sorting technique to classify requirements change,” *Proceedings of the 12th IEEE International Requirements Engineering Conference (RE’04)*, IEEE, 2004, pp. 240–248.

[93] O’REILLY, C., “A weakly constrained approach to software change coordination,” *Proceedings of the 26th International Conference on Software Engineering* (ICSE’04), IEEE, 2004, pp. 66–68.

[94] OSBORNE, M., and MACNISH, C., “Processing natural language software requirement specifications,” *Proceedings of the Second International Conference on Requirements Engineering* (ICRE’96), IEEE, 1996, pp. 229–236.

[95] GOUGH, P., FODEMSKI, F. T., HIGGINS, S. A., and RAY, S. J., “Scenarios - an industrial case study and hypermedia enhancements,” *Proceedings of the 2nd IEEE International Symposium on Requirements Engineering* (SRE’95), IEEE, 1995, pp. 10–17.

[96] PANYUNHE, Z. Y., “Agent-oriented analysis and modelling,” *ACM SIGSOFT Software Engineering Notes*, vol. 25, no. 3, 2000, pp. 36–40.

[97] FENKAM, P., GALL, H., and JAZAYERI, M., “Visual requirements validation: Case study in a Corba-supported environment,” *Proceedings of the IEEE Joint International Conference on Requirements Engineering* (RE’02), IEEE 2002, pp. 81–88.

[98] GRÜNBACHER, P., EGYED, A., and MEDVIDOVIC, N., “Reconciling software requirements and architectures: The CBSP approach,” *Proceedings of the 5th IEEE International Symposium on Requirements Engineering* (RE’01), IEEE, 2001, pp. 202–211.

[99] GRÜNBACHER, P., HALLING, M., BIFFL, S., KITAPCI, H., and BOEHM, B. W., “Repeatable quality assurance techniques for requirements negotiations,” *Proceedings of the 36th Hawaii International Conference on System Sciences* (HICSS’03), IEEE, 2002, p. 9.

[100] MASON, P., COSH, K., and VIHAKAPIROM, P., “On structuring formal, semi-formal and informal data to support traceability in systems engineering environments,” *Proceedings of the 13th ACM conference on Information and knowledge management* (CIKM’04), ACM Press, 2004, pp. 642–651.

[101] SAWYER, P., SOMMERVILLE, I., and VILLER, S., “Capturing the benefit of requirements engineering,” *IEEE Software*, vol. 16, no. 2, 1999, pp. 78–85.

[102] LINDSAY, P., MACDONALD, A., STAPLES, M., and STROPPER, P., “A framework for subsystem based configuration management,” *Proceedings of the 2001 Australian Software Engineering Conference* (ASWEC’01), IEEE, 2001, pp. 275–284.

[103] PRESSMAN, R. S., *Software Engineering - A Practitioner’s Approach*, 6th ed., McGraw-Hill, 2004, 0-07-123840-9.

[104] RAHIKKALA, T., and SEPPANEN, V., “From VSC attributes and characteristics to SCM challenges. Proceedings of the 26th Euromicro Conference (EURMIC’00), IEEE, vol. 2, 2000, pp. 308–316.

[105] GALLANT, R., WINOKUR, M., and KUDISH, J., “Tool and Method Reciprocity: A Case Study in Requirements Management,” *Proceedings of the 7th Israeli Conference on Computer Systems and Software Engineering* (ICCSSE’96), IEEE, 1996, pp. 109–118.

[106] SAGHEB-TEHRANI, M., and GHAZARIAN, A., “Software Development Process: Strategies for Handling Business Rules and Requirements,” *ACM SIGSOFT Software*

EngineeringNotes, vol. 27, no. 2, 2002, pp. 58–62.

[107] SALO, A., and KAKOLA, T., "Groupware Architecture for Requirements Processes in New Product Development," *Proceedings of the 32nd Annual Hawaii International Conference on System Sciences* (HICSS'99), IEEE, 1999, pp. 9.

[108] SALVATORE, F. J., and ALAMEDA, T., "Daily challenges in requirements engineering," *Proceedings of the 11th International Requirements Engineering Conference* (RE'03), IEEE Computer Society, 2003, pp. 297.

[109] SCHWILLE, J., "Modeling Product and Process Characteristics in Software Configuration Management," *Proceedings of the 1st Euromicro Conference on Software Maintenance and Reengineering* (EUROMICRO'97), IEEE, 1997, pp. 25–32.

[110] SEAWLHO, P., and SUWANNASART, T., "A SCM Workflow Model for CMM Organizations," *10th Asia-Pacific Software Engineering Conference* (APSEC'03), IEEE Computer Society, 2003, pp. 253–260.

[111] SILVA, A., "Requirements, Domain and Specifications: A Viewpoint-Based Approach to Requirements Engineering," *Proceedings of the 24th International Conference on Software Engineering* (ICSE'02), IEEE, 2002, pp. 94–104.

[112] SPANOUDAKIS, G., "Plausible and Adaptive Requirement Traceability Structures," *Proceedings of the 14th International Conference on Software engineering and knowledge engineering* (SEKE'02), ACM Press, 2002, pp. 135–142.

[113] BUHNE, S., HALMANS, G., POHL, K., WEBER, M., KLEINWECHTER, H., and WIERCZOCH, T., "Defining Requirements at Different Levels of Abstraction," *Proceedings of the 12th IEEE International Requirements Engineering Conference* (RE'04), IEEE Computer Society, 2004, pp. 346–347.

[114] HALL, T., BEECHAM, S., and RAINER, A., "Requirements Problems in Twelve Software Companies: an Empirical Analysis," *Proceedings of IEE Software*, vol. 149, no.5, 2002, pp. 153–160.

[115] JAVED, T., MAQSOOD, M., and DURRANI, Q. S., "A Study to Investigate the Impact of Requirements Instability on Software Defects," *ACM SIGSOFT Software Engineering Notes*, vol. 29, no. 3, 2004, pp. 1–7.

[116] MENZIES, T., EASTERBROOK, S., NUSEIBEH, B., and WAUGH, S., "An Empirical Investigation of Multiple Viewpoint Reasoning in Requirements Engineering," *Proceedings of the IEEE International Symposium Requirements Engineering* (ISRE'99), IEEE, 1999, pp. 100–109.

[117] VARKOI, T. K., and MAKINEN, T. K., "Case Study of CMM and Spice Comparison in Software Process Assessment," *Proceedings of the International Conference on Engineering and Technology Management* (IEMC'98), IEEE, 1998, pp. 477–482.

[118] VON KNETHEN, A., and GRUND, M., "Quatrace: A Tool Environment for (Semi-) Automatic Impact Analysis Based on Traces," *Proceedings of the International Conference on Software Maintenance* (ICSM'03), IEEE, 2003, pp. 246–255.

[119] WALKER, A., "Quality Management Applied to the Development of a National Checklist for ISO 9001 Audits for Software," *Proceedings of the 3rd IEEE International Software Engineering Standards Symposium and Forum*, (ISESS'97), IEEE, 1997, pp. 6–14.

[120] WANG, Q., and LAI, X., "Requirements management for the incremental development Model," *Proceedings of the 2nd Asia Pacific Conference on Quality Software*, (APAQS'01), IEEE, 2001, pp. 295–301.

[121] WESLEY J. L., ROSSON, B. R., and ARTHUR, J. D., "Effectiveness of Elicitation Techniques in Distributed Requirements Engineering," *Proceedings of the IEEE Joint International Conference on Requirements Engineering* (RE'02), IEEE, 2002, pp. 311–318.

[122] WIELS, V., and EASTERBROOK, S., "Management of Evolving Specifications using Category Theory," *Proceedings of the 13th IEEE International Conference on Automated Software Engineering* (ASE'98), IEEE Computer Society, 1998, pp. 12–21.

[123] WIELS, V., and EASTERBROOK, S., "Formal Modeling of Space Shuttle Software Change Requests using SCR," *Proceedings of the IEEE International Symposium on Requirements Engineering* (RE'99), IEEE Computer Society, 1999, pp. 114–122.

[124] STUFFLEBEAM, W., ANTON, A. I., and ALSPAUGH, T. A., "Smart - Scenario Management and Requirements Tool," *Proceedings of the 11th IEEE International Requirements Engineering Conference* (RE'03), IEEE Computer Society, 2003, pp. 351.

[125] WOIT, D. M., "Requirements Interaction Management in an Extreme Programming Environment: A Case Study," *Proceedings of the 27th international conference on Software engineering* (ICSE'05), ACM, 2005, pp. 489–494.

[126] CHEN, Y., SHIH, W., and SHEN, C., "Distributed Engineering Change Management for Allied Concurrent Engineering," *International Journal of Computer Integrated* Manufacturing, vol. 15, no. 2, 2002, pp.127, 25.

[127] Zowghi, D. A., "Requirements Engineering Process Model Based on Defaults and Revisions," *Proceedings of the 11th International Workshop on Database and Expert Systems Applications* (DEXA'00), IEEE, 2000, pp. 966–970.

PART TWO

Setting the Course for a Complex Process

The literature survey presented an overview of what requirements issues are given most attention in the academic literature. A heavy overweight of articles on requirements management in the plan-driven paradigm was found and has triggered us to further explore the area of requirements management. The fact that the plan-driven approach still dominates the research literature is a concern, as the plan-driven thought emphasizes a static view of the world, even though the environment, in virtually every project, is continuously changing.

> *"In short, the software product is embedded in a cultural matrix of applications, users, laws, and machine vehicles. These all change continually, and their changes inexorably force change upon the software product"*
>
> - ***Brooks [p 13]***

BROOKS, F. P., "No silver bullet: Essence and accidents of software engineering," IEEE Computer Society Press., vol. 20, no. 4, 1987, pp. 10–19.

This part still concerns requirements management, but specifically the agile paradigm in comparison to the plan-driven. To exemplify the two development approaches, we have chosen two contemporary development methods: Scrum and OOA&D.

In order to analyze these approaches thoroughly we adopt four different philosophical world views. These views, along with the methods, are then discussed in relation to a fictitious development project. As a result we present a strategy for developing the specific project. ■

1. Introduction

"Agile and plan-driven development approaches have generally been seen as opposing viewpoints, and the rhetoric on both sides still remains essentially confrontational. The claims and counter-claims, misrepresentations and salesmanship create a sense of perplexity in those of us who simply want to successfully complete our projects and please our customers."

- Boehm and Turner [6 p 1]

When developing information systems it is necessary at some point to decide, which requirements to implement. The question is, how and when it should be decided. Different philosophical groundings motivate different answers to these questions.

Kienholz's presentation of Churchman's inquiry modes are used as reference.

In his book *The Design of Inquiring Systems* from 1971, C. West Churchman [17] proposed five traditions of inquiry which are based upon five different philosophical directions. These five philosophically based inquiring modes are a frame of reference for thinking about how we gather data, how we ask questions, how we solve problems, and make decisions. According to Kienholz [17], mental models can explain why two people can observe the same event and describe it differently; namely because different mental models pay attention to different details. Likewise, different inquiring modes attract attention to different details in an event. Due to the tacit nature of mental models, they are generally invisible. They are ingrained assumptions and generalizations which influence how we understand and behave in the world. The purpose of reasoning about different inquiring modes is to bring different mental models to the surface, in order to gain more insight into the maps used for navigating through, for instance *wicked problems* [17].

According to Churchman, every survey and research study is guided by implicit assumptions for gaining knowledge and insight into some field. Burrell and Morgan follow this line of work in their description of an analytical framework for understanding organizations. They operate with four different philosophical directions; the functionalist, the social relativist, the radical structuralist, and the radical humanist. These underlie the way we conceive the nature of the social world [9]. Building on Burrell and Morgan's analytical framework, Hirschheim and Klein identifies four similar philosophical lines of assumptions underlying information system development. Their main argument is, that different philosophical assumptions lead to different approaches for information system development and in the end it results in different constructions of information systems [14].

For instance, the decision of when and which requirements are to be implemented will depend on which philosophical line of inquiry one adheres to. Each decision during the process will be made according to some mental map. The question is how these implicit assumptions represent themselves during the system development process, and how they will affect the management of requirements. Especially, opportunities and consequences of a non-functionalist approach seem to be a relatively unexplored area, as the majority of development methods are predominantly functionalist.

The **functionalist** perceives the social world as an objective real and as an observable facticity, see section 3.1.

1.1 Plan-driven or Agile Development?

The plan-driven and agile development have different assumptions and priorities. The plan-driven development methods are substantially positivistic in nature, disregarding issues which another philosophical foundation would potentially see as critical issues. Hence, continuous user interaction is critical in agile development, but in the extreme case it is regarded as unnecessary disturbance in plan-driven development.

Agile methods are organized to accommodate changing and evolving requirements better than plan-driven development methods. Agile development methods do not value a great amount of documentation, as it hinders re-prioritizing requirements when necessary. Development typically proceeds according to which requirements have highest priority, therefore planning is not a time-consuming activity in agile development methods. Actually, planning is regarded impossible, as the current state of affairs is bound to change somehow; that is why the highest priority requirements are implemented first, and other requirements are implemented if required.

Agile development is based upon user stories as representations of requirements to be implemented; these requirements are then typically validated on a continuous basis by users. Plan-driven development methods base implementation on extensive analysis and modelling documents and do typically not let users validate the system during the implementation phase.

1.2 Problem Statement

Through the introduction we have taken a critical stance towards the predominantly functionalist approach for system development, that naturally entails a mainly functionalist approach for requirements management. We also remarked that all knowledge and data gathering necessarily will be done through a set of glasses, that implies a certain world view and certain implicit assumptions. In other words, every inquiry will be guided by the implicit assumptions inherent in a specific world view; and it is not possible to have a neutral or objective world view. This means, that a person being functionalist in nature will notice certain aspects of some event, while a person adhering to a different philosophy will probably notice other aspects of the same event. Therefore, a convinced functionalist, who rejects inputs from persons with other world views, will fail to notice some aspects of an event.

Hence, we want to examine how the predominantly functionalist approach for requirements management can be nuanced. First, we wish to be more specific about how different philosophical assumptions express themselves in the context of system development and requirements management. Next, we wish to examine if the functionalist approach for system development could incorporate aspects from other philosophical directions, in order to reduce the "one-eyed-ness" of the functionalist approach. That way, perhaps it would be possible to obtain benefits from several philosophical perspectives and analyze a complex scenario as thorough as possible. To obtain the philosophical perspectives we make use of Burrell and Morgan's [9] as well as Hirschheim and Klein's [14] four different philosophical paradigms.

In order to relate the four theoretical philosophical paradigms to a systems development context, we wish to examine how a convinced project manager of each philosophical paradigm will comment on two widely applied development methods: Scrum and OOA&D, and the requirements management process inherent in these two methods. We have chosen these two particular methods to be evaluated and judged, as they each represent a widespread and well-known method from the agile and the plan-driven development paradigm.

To further relate the philosophical paradigms and their theoretical assumptions to practice, we set up a fictitious project proposal, that acts as an example of a *wicked problem*. This project will be used for reasoning about how the different philosophical assumptions express themselves when a system development process is to be planned.

Thus, the problem statement goes:

- **Which strengths and weaknesses of Scrum and OOA&D do each of the four philosophical paradigms observe?**
- **Which main risks do each of the four philosophical paradigms notice in the fictitious project?**
- **Which method would a functionalist project manager choose for the fictitious project, when taking the views of the three alternative paradigms into consideration?**
- **With the chosen development method in mind, how can a possible development strategy look like?**

2. Method

In this section our approach for providing an answer to the problem statement is specified. The essence of this project will be the four philosophical paradigms, which we use for evaluating requirements management in Scrum and OOA&D, and for carrying out a thorough analysis of a fictitious project. In the forthcoming sections our choice of theoretical foundation, analysis framework, our narrative technique, and tools for our analysis will be discussed.

2.1 Choice of Theoretical Foundation

Our approach for providing an answer to the problem statement is inspired by Churchman's presentation of five different traditions of inquiry, which is based upon five different philosophical directions. The five traditions each entail particular strategies for understanding the world and behaving in it [17]. As system development and requirements management tend to be very complex showing symptoms of *wicked problems* [23], awareness of implicit assumptions underlying specific preferences may provide further insight into the consequences of different choices.

In their article *Four Paradigms of Information Systems Development* Rudy Hirschheim and Heinz K. Klein [14] argue that there is a relation between the approach applied for system development and the design, and configuration of the end system. This article builds the characteristics of the four different development paradigms on Gibson Burrell and Gareth Morgan's book *Socio-*

logical Paradigms and Organisational Analysis [9]. The book presents a coherent and complete map of social and organizational theories, that are still relevant. The cohesion and completeness of their analytical framework is a result of using two continuums that incorporate a wide spectrum of opposing theories. The ends that define each of the continuums are still relevant, and this is why we believe that Burrell and Morgan's analysis framework is suitable for analyzing implicit assumptions in an information system development context.

2.1.1 The Four Philosophical Paradigms

Burrell and Morgan have identified four distinct sociological paradigms, that define fundamentally different perspectives on analysis of the social world. These four perspectives are the *functionalist* paradigm, the *social relativist* paradigm, the *radical structuralist* paradigm, and the *radical humanist* paradigm. Through the identification of the four paradigms, Burrell and Morgan suggest, that it will make sense to examine social phenomena in terms of four sets of basic assumptions. Each paradigm views the world in a particular way. Hence, the four paradigms define four views on the social world in terms of meta-theoretical assumptions with regard to the nature of science and society. Under each paradigm there is room for much variation, but still there is an underlying unity in terms of implicit assumptions. The four paradigms should be considered mutually exclusive:

Burrell and Morgan adapts Thomas Samuel Kuhns notion of a paradigm, as a set of practices that define a scientific discipline during a particular period of time.

The social relativist perspective is similar to Burrell and Morgan's interpretive perspective. The notion social relativist stems from Hirschheim and Klein [14].

> *"They offer alternative views of social reality, and to understand the nature of all four is to understand four different views of society."*
>
> ***- Burrell and Morgan [9 p 25]***

The point to note here, is that a synthesis of the four paradigms is not possible, as they are contradictory in terms of meta-theoretical assumptions. They are alternatives, as the acceptance of one paradigm will reject the others more or less. Over time it is possible to adhere to different perspectives, but at any given point in time, it is only possible to accept the assumptions of one paradigm. The four paradigms together, provide a map for negotiating some social subject. Further, they can be used for identifying a frame of reference for various theorists.

The Analysis Framework - Two Continuous Axes

Central to Burrell and Morgan's four paradigm thesis is that all organization theory is "*based upon a philosophy of science and a theory of society*" [9 p 2]. The first dimension of Burrell and Morgans analysis framework is the subjective-objective continuum, which refers to the philosophy of science. The second dimension of the framework is the radical change-regulation continuum, which relates to the theory of society. The analytical framework as well as the positions of the four paradigms can be seen in figure 2.1. The next two sections are devoted to explaining the continuums of the framework. The actual description of the paradigms can be found in chapter 3.

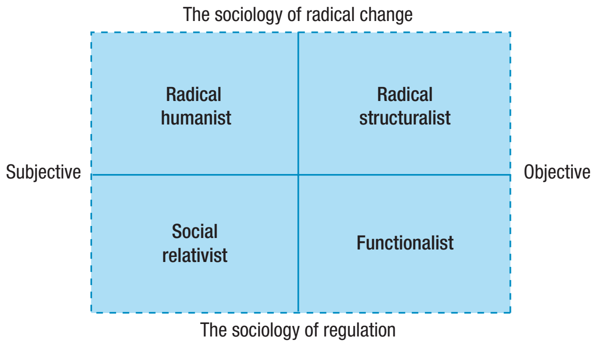


Figure 2.1: The placing of the four paradigms in the analysis framework [9].

Subjectivity and Objectivity in Philosophies of Science

The philosophy of science is related to assumptions concerning ontology, epistemology, human nature, and methodology. These four sets of assumptions have a subjective and an objective side.

Ontology refers to assumptions concerning the essence of the examined phenomena. The basic ontological question is whether reality can be conceived as something external to the individual, or if it is a product of the individual's mind. The more objective assumptions fall under the realism branch; whereas the subjective assumptions can be characterized as nominalistic regarding every term only as a name and not relating to anything in real. For instance, for a nominalist a dog would only be a concept and not an entity in the *real world*.

Epistemology concerns the grounds of knowledge; how the world can be understood, as well as how knowledge can be communicated from one individual to another. The two extreme epistemological positions in relation to objectivity and subjectivity are positivism and anti-positivism. The positivist sees knowledge as something that can be acquired, while the anti-positivist believes that in order to gain knowledge on some phenomenon, it has to be experienced.

The third set of assumptions regarding the philosophy of science has to do with human nature. Assumptions about human nature concerns "*the relationship between humans and their environment*" [9 p 2]. In the extreme objective case man responds to the environment in a mechanistic way; he is regarded as a product of the environment. At the other end of the spectrum, human beings have a free will and are regarded as the creator of the environment; "*the controller as opposed to the controlled*" [9 p 2]. Hence, the two ends of the continuum for assumptions concerning human nature are determinism and voluntarism.

Ontological, epistemological, and human nature assumptions have a direct impact on the last set of assumptions. Methodological assumptions refer to the approach for examining and gaining knowledge about the social world. At the objective end on the continuum of methodological assumptions lies the pure nomothetic approaches. The idiographic approaches are situated at the other end of the continuum. Nomothetic approaches treat the social world as an external objective reality, whereas the idiographic approaches focus upon how the individual creates and interprets the world she lives in.

Theories of Society

The second dimension of Burrell and Morgan's analysis framework concerns theories of society. At one end of this continuum is the sociology of regulation; at the other end of this continuum is the sociology of radical change dimension. The debate between the two opposing models of society is referred to as the order-conflict debate.

The two ends of the continuum stand for two opposing models of society, which draw on two competing sets of assumptions. The regulation end of the continuum is concerned with explaining the nature of social order and equilibrium in the society. This model of society attempts to explain why society holds together rather than falling apart, and it stresses concepts such as commitment, cohesion, solidarity, consensus, reciprocity, co-operation, integration, stability, and persistence.

The other model of society is in stark contrast to the sociology of regulation. The sociology of radical change is based upon the concern for explaining deep-seated structural conflicts and modes of domination. The model stresses concepts as coercion, division, hostility, dissent, conflict, integration, and change. This sociology is:

> *"essentially concerned with man's emancipation from the structures which limit and stunt his potential for development [...] both material and psychic."*
>
> ***- Burrell and Morgan [9 p 17]***

2.1.2 The Four Paradigms in Information System Context

Hirschheim and Klein's [14] article is used as a supplement to Burrell and Morgan due to their specific use of the four philosophical paradigms in an information system development context. Hirschheim and Klein have described main objectives, considerations, threats, important stakeholders, and practical considerations in system development depending on which paradigm one adheres to. Their main objective is to attract attention towards how system developers' world view can influence the development process and the implemented system. We use Hirschheim and Klein as the stepping stone from Burrell and Morgan's four theoretical paradigms to describe objectives, considerations, threats, important stakeholders, and practical considerations for requirements management under each of the four paradigms.

2.2 A Narrative Approach

For providing an answer to the problem statement we will apply a narrative style from chapter 5 and onwards. In order to thoroughly explore each of the four philosophical paradigms, each group member will take the role of either a functionalist project manager, a social relativist project manager, a radical structuralist project manager, and a radical humanist project manager. Letting each of us explore one of the roles and specify the actual project managers opinion on requirements management in Scrum and OOA&D as well as give an opinion on how to plan a development process that facilitates the best possible requirements management, will encourage a discussion of the differences and conflicts between the perspectives.

The project proposal is given to the functionalist project manager. He alone is to decide which development method to use and how to plan the requirements management process. Our literature survey revealed an emphasis on plan-driven requirements management in the academic literature. Burrell And Morgen further states:

> *"Positivist epistemology is in essence based upon the traditional approaches which dominate the natural sciences"*
>
> ***- Burrell and Morgan [9 p 5]***

We therefore give the project proposal to the functionalist project manager, and advise him to incorporate the other three project managers considerations into his decisions. This way, we believe we can nuance the functionalist approach and still propose a realistic solution to the project proposal.

2.3 Choice of Development Methods and Project Proposal

We have chosen to work with Scrum and OOA&D as they are both well-known and widely used by practitioners. Further, Scrum has been chosen among other agile development methods, because it facilitates some degree of project management, and it is claimed to be usable for large projects [24]. OOA&D is chosen as a representative of a plan-driven development method because in more ways it is opposite to Scrum. OOA&D puts much emphasis on documentation; it does not, however, provide explicit guidelines for planning a development process. Scrum and OOA&D each focus on different aspects of system development and therefore we believe they will be suitable methods to be evaluated by the four philosophical paradigms.

As mentioned in the introduction, to be able to relate the four philosophical paradigms to practice, we set up a fictitious project proposal where it has to be chosen if either Scrum or OOA&D should be used as the development method. The project proposal has the characteristics of a *wicked problem*, as it is very complex, providing possibly many different interpretations depending on one's philosophical standpoint. It does not clearly state the system to be developed, it merely describes in general terms the system's main objective. However, setting up this kind of generally formulated task, we believe is realistic for evaluating risks and doing the initial planning of the project.

2.3.1 Risk Analysis for Evaluating Scrum and OOA&D

We have chosen to use risk analysis in order to decide on which method to use for the given project. We employ a method for evaluating the severity and impact of the risks that follows the functionalist way of reasoning well. This is done, as it is the functionalist that has to make the decision of which method to use.

We have based our analysis on Boehm's [5] and Baskerville and Stage's [2] method for risk analysis. The overall purpose with risk analysis is to identify the high risk elements of the project, provide an overview of consequences which lies in the risks and the severity of each risk. We see risk analysis as a useful framework for determining priorities, obtain overview of the current situation, and measure risks. Baskerville and Stage put it in the following way:

> *"Once the risk factors received their compound rank, the analysis was simplified, and resolution strategies for the risk factors with the highest product were discussed."*
>
> ***- Baskerville and Stage [2 p 494]***

We have decided to define the risks from a set of activities, specifying what consequences the risk might have on the project, and the total probability. The activities are divided into the five requirement issues described later in the chapter. With the above activities in mind, risk will be evaluated and the consequence of following the activity. A simple mathematical measurement will then be made in which severity (**S**), probability (**P**) and a total (**T**) will be variables.

- Severity: **S**, the economic cost of loss attributed to such an exposure

- Risk probability: **P**, is the probability of an exposure's occurrence
- Risk: **R**, the risk related to this exposure is calculated as the product of the two elements; **R = P * S**.

In the next section we describe generic critical issues of requirements management, that the functionalist can use in his risk analysis.

2.4 Issues in Requirements Management

The requirements management activities are essential in the sense that they are all critical to the success of the information system development process. Our classification is primarily based on the SWEBOK [8], as this is considered the general guide to software engineering. The classification of the different activities will be used to structure and evaluate the risk analysis of Scrum and OOA&D.

2.4.1 Requirements elicitation

Requirements elicitation is the activity, which has the purpose of gathering user needs. The purpose of the process is not to obtain an agreement on which requirements should be implemented, but rather to collect a pool of all the requirements, which could be needed. The SWEBOK describes requirements elicitation as the first stage in building an understanding of the problem the software is required to solve. Further, requirements elicitation is fundamentally a human activity, and is the activity in which the stakeholders are identified and relationships are established between the development team and customers [8]. Ian Sommerville [26] describes the process as very difficult as many different stakeholders could be involved. Stakeholders' inability to know what they need, political factors which have an influence on the requirements, economics, and the business environment has great influence on the process of requirements elicitation.

2.4.2 Requirements negotiation

Requirements negotiation concerns a negotiation of which requirements should be implemented. The result of the activity is an agreement, by all the stakeholders, on the set of requirements which should be implemented. It can be described as a *conflict resolution* activity, which concerns resolving problems with requirements, where conflicts occur between two stakeholders requiring incompatible features [8, 22]. The process could also be classified as a sub-topic within requirements validation, but often problems emerge as a result of some analysis, and this makes us classify requirements negotiation as a fundamental sole requirements activity.

2.4.3 Requirements documentation

Requirements documentation is concerned with documenting and describing requirements. Documentation of requirements is often considered as a way to analyze and model requirements, and to write these into requirements specifications. Typical software has a large number of requirements and the emphasis is shared between performing the quantification and managing the complexity of interaction among the large number of requirements. This is mainly the reason, why the *requirements specification* activity refers to the production of a document or its electronic equivalent, which is further possible to review, evaluate, and approve [8].

2.4.4 Requirements validation

Requirements validation concerns the activity of checking the requirements for consistency and completeness. This activity exposes errors in the requirements and points out problems to be corrected [26]. The aim is to pick up any problems before resources are committed to addressing the requirements. The validation is often performed in relation to the requirements specification. SWEBOK mentions four activities within requirements validation; requirements reviews, prototyping, model validation, and acceptance tests. All have different purposes and should all help in determining inconsistency in requirements [8].

2.4.5 Requirements process management

Requirements process management is concerned with process models, process actors, process support and management, and process quality and improvement [8]. A part of this activity is how requirements are managed and organized and how open the process model is to changes. The process model is concerned with how the activities of elicitation, negotiation, documentation, and validation are configured for the current project. Further the role of the actors in the process should be described. Finally, this activity is concerned with planning and estimating the project.

2.5 Project Structure

The overall composition of the project is as pictured in figure 2.2.

- **Chapter three** contains our theoretical foundation, the four employed paradigms: *Functionalism, social relativism, radical structuralism,* and *radical humanism*. The descriptions are based upon the theory presented by Burrell and Morgan, and Hirschheim and Klein.
- **Chapter four** contains a description of the two development methods Scrum and OOA&D.
- **Chapter five** introduces a functionalist narrator and the paradigms presented in chapter three, are exemplified as four project managers with different world views. Moreover, a complex project is introduced, where the functionalist asks for advice on the choice of development method. As a conclusion the three other project managers are introduced.
- **Chapter six** presents the functionalist's evaluation of opportunities and limitations of Scrum and OOA&D.
- In **chapter seven** the word is passed on to the three project managers. First, the three reviews of Scrum is presented. Next, follows the three reviews of OOA&D. Each review is summarized in a table to provide an overview.
- **Chapter eight** presents the challenges, which the three managers no-

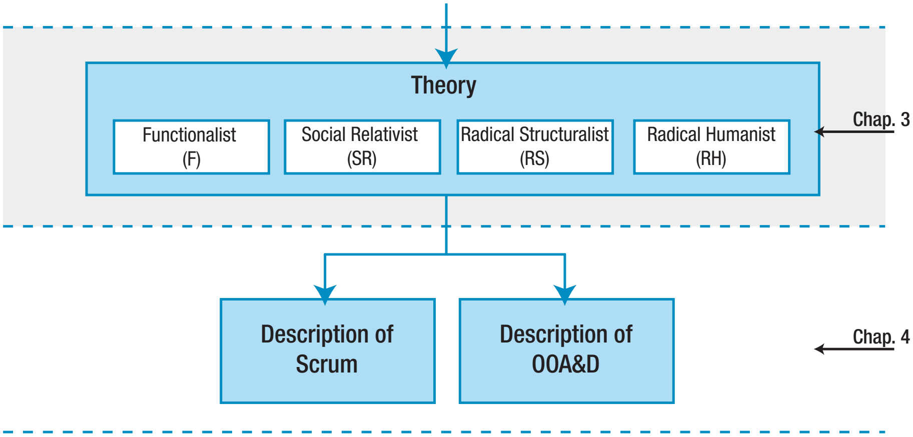
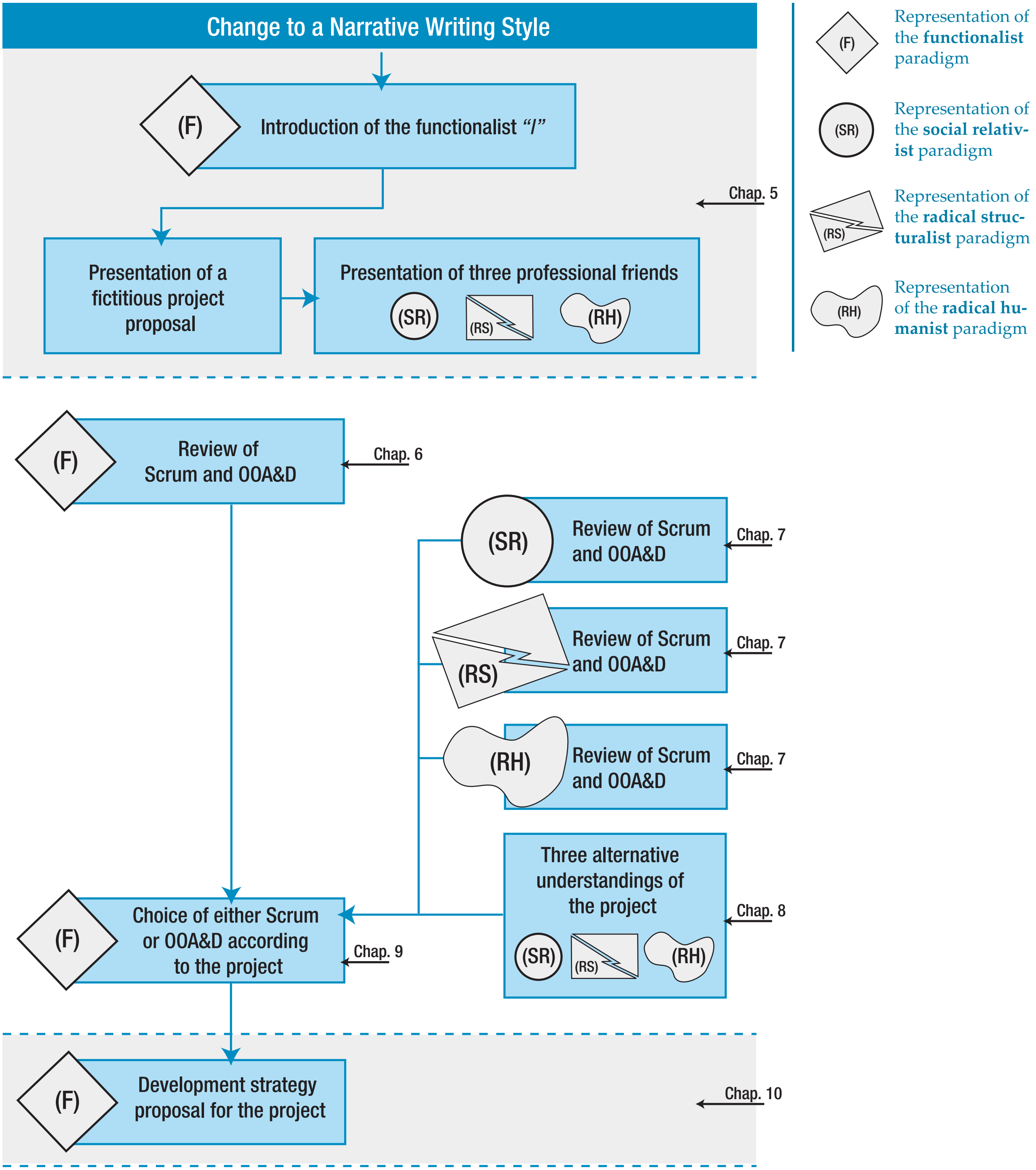


Figure 2.2: A representation of the structure of the project. It is continued on the next page.


Change to a Narrative Writing Style
(F)
Introduction of the functionalist "I"
Presentation of a fictitious project proposal
Presentation of three professional friends
(SR)
(RS)
(RH)
Chap. 5
(F)
Review of Scrum and OOA&D
Chap. 6
(SR)
Review of Scrum and OOA&D
Chap. 7
(RS)
Review of Scrum and OOA&D
Chap. 7
(RH)
Review of Scrum and OOA&D
Chap. 7
Three alternative understandings of the project
(SR)
(RS)
(RH)
Chap. 8
(F)
Choice of either Scrum or OOA&D according to the project
Chap. 9
(F)
Development strategy proposal for the project
Chap. 10
(F)
Representation of the functionalist paradigm
(SR)
Representation of the social relativist paradigm
(RS)
Representation of the radical structuralist paradigm
(RH)
Representation of the radical humanist paradigm

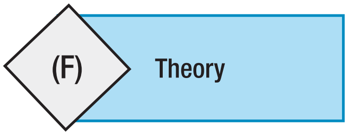
tice in the proposed project. Each makes an assessment of the project, points out main risks and concerns, and gives advice accordingly.

- In **chapter nine** the functionalist project manager makes his choice of development method by applying risk analysis.
- In **Chapter ten** OOA&D is refined, in order to make it suitable for the project. Finally, the refined development strategy is presented.

3. Theoretical Foundations

In this section the theoretical foundation of the four employed paradigms; functionalism, social relativism, radical structuralism and radical humanism are described. These descriptions are based on the theory of Burrell and Morgan. After each description the four paradigms are described in an information systems context by using Hirschheim and Klein's *"Four Paradigms of Information Systems Development"* [14].

3.1 Functionalism



Functionalism is a dominant theoretical perspective in sociology and other social sciences. The underlying philosophy is based on two aspects: Application of the scientific method in the objective social world and use of an analogy between the individual organism and society. The functionalist perceives the social world as an objective real and as an observable facticity [14].
Burrell and Morgen describes functionalism this way:

> *"It is characterised by a concern for providing explanations of the status quo, social order, consensus, social integration, solidarity, need satisfaction and actuality. It approaches these general sociological concerns from a standpoint which tends to be realist, positivist, determinist and nomothetic."*
>
> ***- Burrell and Morgan [9 p 26]***

Realism is concerned with the belief that properties exist independent of the things that manifest them [9].

Positivism is a philosophy developed by Auguste Comte in the beginning of the 19th century, which stated that only authentic knowledge is scientific knowledge.

According to Burrell and Morgan's axis division in the objective-subjective dimension and the regulation-radical change dimension, the functionalist is placed in the objective and regulation part of the axis. It indicates that a functionalist has an ontology of realism and believes that the world is observable and perceivable. He furthermore believes that factorial events in an organization can be measured as a cause-effect relationship, implying a positivist epistemology [14].

Determinism is concerned with an understanding of that everything that occur in a system, is based on physical outcomes of causality. Every action would produce a reaction, or effect.

A **nomothetic** explanation presents a generalized understanding of a given case.

Functionalism seeks to provide essential explanations of social affairs, and is concerned with the control and regulation of social affairs. It is done by understanding the society in a way which generates knowledge that can be put into use. It often has a problem-oriented approach to solutions and is concerned with practical solutions to practical problems. Functionalism emphasizes the importance of understanding order, equilibrium, stability in the society, and maintenance of it.

The functionalist approach to social science often tends to assume that the world is composed of relatively concrete empirical artefacts and relationships, which can be identified, studied and measured through approaches found in natural sciences [9]. The paradigm has evolved in the nineteenth century with great influence from social theorists as August Comte, Herbert Spencer, Emile Durkheim and Vilfredo Pareto.

The functionalist paradigm has been influenced by different elements from different social thoughts. The influence has come from the German idealism and Marxism and is rooted in the tradition of social positivism.

3.1.1 Functionalism in an IS Context

IS - short for information system.

Based on the previous description of functionalism we operationalize the basic ideas of functionalism in relation to IS. The functionalist believes in objectivity and social order. With an objective view, the world can be seen as an objective real, which can be analyzed and described. Furthermore, the world is seen as stable, which makes it possible to analyze and describe the cause and effect relations within the world. With the social order, the organism is understood as having one common goal, which is seen as a part of a united whole and is therefore supporting the society with different functions.

When developing systems it is suggested that all information systems are designed to contribute to specific ends. The ends are defined by the management of the organization. These ends can be described in system objectives, which can be translated into factual specifications. Conflicts can occur, but with methods and tools these can be prevented. It is assumed that the purpose and goals of the system are agreed by all parts [14].

Political and human factors are seen as disturbances and irrational, and the way to deal with these disturbances, are with tools and methods. The tools and methods make it possible to control the factors in such a way, that they do not complicate the development process [14].

Information systems developed from this perspective have the purpose to support rational organizational operation and effective and efficient project management. The effectiveness and efficiency of an information system can be tested by objective means. Requirements specifications are built on the notion of a manifest and rational organizational reality, which makes it possible to predict cause-effect relationships and to write them down [14].

As a positivist, the approach to systems development is to gain knowledge about the organization by searching for measurable cause-effect relationships. It is assumed that there are general laws or regular patterns, which can help explain and predict the reality. The developer is a neutral observer and has the ability to see chains of events. By being objective it is possible to minimize the risk of being prejudiced by other interests. When developing a system, the developer has to express an objective view of the world instead of his own world view. He thereby employs a clear distinction between himself and the problem domain he is going to examine. The developer has to remain in a neutral position, in order to represent the facts as they really are.

Scientific methods are preferred for analysis as it is possible to divide things into the most basic components. This is a rational activity and the cause-effect relation can be ensured mathematically. Product quality should be guaranteed, therefore testing is important to the functionalist. Moreover the testing is done to ensure the management, that what has been agreed is developed, and that the resulting product will provide the guaranteed optimizations.

A functionalist is convinced that all involved users and management, can agree on the set of goals and work towards these goals, accordingly. Empirical material from the stakeholders is preferred and is beneficial in order to model and generalize. The empirical material gives the functionalist the ability to make exact specifications, describing and explaining the strategy for solving a given problem [14].

3.2 Social Relativism

(SR) Theory

Social relativism falls into the paradigm labeled the interpretive paradigm by Burrell and Morgan. The paradigm concerns the subjective property of the individual, and strives to understand the fundamental nature of the social world and the subjective experience [9]. In Burrell and Morgans' own words:

> *"It seeks explanation within the realm of individual consciousness and subjectivity, within the frame of reference of the participant as opposed to the observer of action."*
>
> ***- Burrell and Morgan [9 p 28]***

The perception of the individual is the same as in relativism, which states that all people are different. From this it can be concluded that people will have different perceptions in reference to the same occurrences, objects, and statements.

The different understandings play a significant role in how we communicate and perceive occurrences, because it states, that in the interaction with other people and objects, meanings cannot be unambiguously defined. As a reaction to the above, the core belief of the relativist is that there exists no superior truth, as everybody has a different perspective on things.

Interpretive sociology is the term used by Burrell and Morgan to describe that people act in a social environment. The term is highly influenced by the relativist's perception of the individual. The theory does not acknowledge the social world as something that exists. On the other hand, it defines the world as "*an emergent social process which is created by the individuals concerned*" [9 p 28], and is regarded as a network of assumptions and intersubjectively shared meanings.

Focusing on the origins of the interpretive paradigm, the underlying epistemological claims were developed by the German philosopher Immanuel Kant, who stated that humans are incapable of mediated knowledge of the world without interpreting it. Further, he stated that all human experiences of the world are mediated through the human mind, which structures perceptions according to the humans own beliefs. This principle is founded in the German idealism. Prominent authors of the paradigm is Wilhelm Dilthey and Max Weber, who were both concerned with bridging the gap between idealism and positivism. The challenge was to place cultural sciences upon a firm foundation in terms of their objective validity [9].

Interpretive sociology consists of several categories of interpretive theory, which differs foremost in their degree of subjectivity. These are: Solipsism, phenomenology, phenomenological sociology, and hermeneutics. Solipsism stands at one end of the spectra, with the most extreme form of subjective idealism, and states that the world is the creation of the human mind. On the other end hermeneutics focus on interpretations and understanding the products of the human mind, which characterize the social and cultural world. Inhermeneutic theory the socio-cultural environment is seen as a human constructed phenomenon [9].

With the hermeneutic theory in mind Hirschheim and Klein have labeled the interpretive sociology as social relativism in a software development context.

According to Burrell and Morgan's axis division in objective-subjective dimension and regulation-radical change dimension, the social relativist is placed in the subjective and regulation part of the axis. This indicates that a social relativist has an ontology of nominalism, and perceives the reality as socially constructed. The social relativist believes in replacing causal and empirical explanations for social phenomena with sense making. This indicates that the social relativist is of an anti-positivistic epistemology [14].

3.2.1 Social Relativism in an IS Context

In software development, social relativism has emerged to address the shortcomings of the dominating functionalist paradigm, and in many ways social relativism is its direct opposite. Hence the epistemology of social relativism is that of anti-positivism as the search for causal empirical explanations is believed to be misguided and should be replaced by sense making. The ontology is that of nominalism, because the reality is not given, but is socially constructed as a product of the human mind [14].

In this paradigm the users and the system developers are considered key actors. The users act as organizational agents who interpret and make sense of their surroundings, whereas the developer is a change agent and facilitator, who helps the users make sense of the new system and its environment [14]. It is important to point out, that it is the management that defines what ends should be reached. But it is the perspective of the users, which has to be in focus when the developer tries to reach the ends.

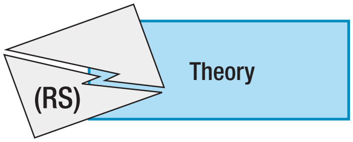
Social relativism recognizes that the reality, in which the system is developed for, evolves through changing traditions, social laws, conventions, cultural norms, and attitudes. Further, the paradigm states that no one, who is involved in the project, has a privileged source of knowledge, as all participants see different parts, and interprets the system in different ways [14]. Another key point is the paradigm's emphasis on acceptance. All activities are towards an acceptance from the intended users.

When developing systems from a social relativist perspective, the information system is believed to be part of the changing social environment and helps to identify desirable and feasible ends [14]. The system developer interacts with the management of the organization, to determine what type of system is appropriate. In other words, the system objectives emerge as a part of the organizational construction of reality. But as the reality is constantly evolving, there is no objective criterion for what is considered good or bad. In the end it is what the management and developer believe is true [14].

Social relativism questions the efficacy of objective and rigorous methods and tools in developing software and thereby takes further distance to the functionalist paradigm. Instead the paradigm favors an approach to systems development which facilitates the learning of all who are concerned and effected. In other words, the focus is not the result of the information system, but the way it is achieved. Hence, the paradigm emphasizes strong user participation as a key element in the development process.

As a final note, the concept of rationality does not play any significant role in the paradigm. The software developers are said to act rationally if they accept the prevailing attitudes and values, remain consistent with general opinion, and develop software which implement changes in such a way that it does not threaten social harmony [14].

3.3 Radical Structuralism



The underlying philosophical assumptions of structuralism is the rejection of the concept of human freedom and choice, and the focus is on the way human behavior is determined by various structures. Structuralism refuses the importance of human subjectivism and has no intention of enhancing the meaning of life. Structuralism focuses on how humans can adapt to structures in society [18].

According to Burrell and Morgan's axis division in objective-subjective dimension and regulation-radical change dimension, radical structuralism is placed in the objective and radical change part of the axis. It indicates that a radical structuralist has a nomothetic ontology and is concerned with generalities, which can be discovered and understood as principles, and which imply the epistemology of positivism. Being placed in the radical change dimension of the axis implies the interest for "*deep-seated conflict, modes of domination and structural contradiction*" [9 p 17]. The basic interest is the "*man's emancipation from the structures which limit and stunt his potential for development.*" [9 p 17].

In the radical structuralist paradigm both the natural and social world are viewed in a materialistic way. Objects in the natural world are seen as a facticity and within the paradigm of radical structuralist the social world is given the same value. The facticities within the social world are taken as much for granted as in the natural world.

The paradigm has much in common with the functionalist paradigm as both paradigms are of a positivist belief. In the functionalist paradigm it is believed that objects in the world can be discovered and understood in patterns and regularities. In the radical structuralist paradigm the same idea of discovering and understanding patterns and regularities is employed not only in the natural world but also in the social world. Objects to be discovered and understood in this paradigm would be: Structure, conflict, modes of domination, contradiction, and deprivation [9]. All these elements are seen as essential factors for understanding the social world.

As a radical structuralist the aim is not only to understand the actions within the social world but also the changes caused by the social elements mentioned above. The fact that some of these elements are in contradiction to each other is what makes the study of the social world interesting. The effects of contradictions can influence the economy and political structure in the social world. To put it in other words, the elements within the social world can be contradictory and therefore cause conflicts, which can lead to some kind of destruction [9].

The foundations of radical structuralism are grounded in Karl Marx's work and as his work has been exposed to many different interpretations it would be naive to describe radical structuralism from one of these interpretations. To support this thought Burrell and Morgan says:

> *"In essence, the radical structuralist paradigm constitutes a body of social theory as complex, conceptually rich and widely differentiated as any of the other three paradigms considered in this work"*
>
> ***- Burrell and Morgan [9 p 327]***

Therefore, Burrell and Morgan have recognized three very broad approaches to the social theory used in this paradigm. The approaches are not only from the marxistic perspectives but also from radical weberian perspective. Similarities and differences in the two perspectives should be clarified with a short description of the three approaches.

Russian Social Theory

Two approaches: Historical materialism by Bukharin and the anarchistic communism by Kropotin form the basis for social theory presented.

Bukharin aims to explain the story of human development in generality through the notion of equilibrium. Having a marxist perspective, the society is viewed as being in an unstable equilibrium state because of imbalance in the environment. He believes the development of technology is the approach to attain balance in the environment. With technology the relationship between the society and nature will be regulated. Bukharin explains social change as being the effect of imbalance, which leads to disagreements at times of crisis. Bukharin states that the acceptance of the natural world and the objects within it is total. The human consciousness should be seen as being dependent on economic production, which implies that the material productive forces are the foundation of the existence of human society [9]. He states it in the following way:

> *"Both in nature and in society there exists objectively (i.e. regardless of whether we wish it or not, whether we are conscious of it or not) a law of nature that is causal in character"*
>
> ***- Burrell And Morgan [9 p 338]***

Kropotin believed in anarchistic communism, which discards human rules and regulations. Being an anarchist, one decides by himself what to do and when to do it. As a biologist he believed in the laws of nature and the socially developed hierarchy was, in his eyes, a pathological development in man's history [9]. Moreover he claimed that the laws of nature form the basis for the laws of society.

Contemporary Mediterranean Marxism

The two philosophers; Althusser and Colletti form the basis for the mediterranean marxism presented.

Althusser is a supporter of the mature Marx, who believes that history is not created by man himself, but by particular configurations of structures in the society. Althusser focuses on the understanding of the totality, in which he recognizes four practices: The economic, the political, the idealogical, and the theoretical. All four practices have a great importance and are interrelated in the analysis of totality, but Althusser also claims that some might be more dominant than others, like the practice of economic. According to Althusser the real world consists of structures together with totality, which represents social formations. And the history and man's actions are determined by the social formations in which they exists. In this view, the subject is seen as an agent in the mode of production and is working according to the economical restrictions [9].

Colletti works with the concept of opposition. There are two views on opposition: Science and philosophy. In the scientific understanding of opposition, the concept is equated with the concept of contradiction. In the philosophical understanding of opposition, the concept is equated with alienation. According to Colletti, both the theories of contradiction and alienation is combined in a single theory. His aim is not to make general laws valid for all societies, but to state the difference [9].

Conflict Theory

The radical Weberian, **Dahrendorf**, forms the basis for the conflict theory presented. Dahrendorf seeks to assist Marx's work with a sociological insight from Weber. In his conflict theory he aims to explain the neglection of order within the industrial society. Within this order he focuses on the differential distribution of authority, which becomes the cause of conflicts [9]. He concentrates on organizations in which authority plays a significant role in the daily work. With this focus he makes a two-class model of structures. In the two-class model there are the ones who do the authoritative work and the ones who act according to the authoritative commands. Dahrendorf sees this as the cause for conflicts, as one could imagine that the aggregates of the authority would assure that the work done is for their benefit.

3.3.1 Radical Structuralism in an IS Context

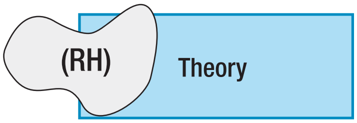
As a radical structuralist the primary focus, when developing an information system, is to ensure that the system is well structured and should be used to structure a given context. Technology is understood and used as a way to structure the working conditions. According to Bukharin, technology can help balance the environment. Moreover by using or regulating working conditions with technology it is possible to avoid unnecessary changes. The system is, by it self, seen as a change, but a change to the better.

When developing a system the surroundings are seen in a totality. The stakeholders, who are to use the system, are seen as one group and not as individuals. Therefore, the need for more than one stakeholder to help develop the system is not necessary according to a radical structuralist. The groups which are involved in the development are seen as being in a conflicting situation. It is assumed that the internal relationship between the two groups is uneven and there is a dominant part. Normally the management is the authority and therefore they decide what the system should be capable of or not. To summarize, the system is developed to decrease the chances for conflicts and improve structure, but is only developed in cooperation with one of the groups.

The system developer or the project manager do not, as in the social relativist paradigm, have the responsibility to assure that every group or individual is in agreement with the rest of the group members. In the radical structuralist's opinion, the system is developed to obtain some objective goals and to assure a higher degree of profitability.

When developing a system the primary focus is on the benefits it can give; depending on who the system is being developed for. The benefits can either be financially or benefits that improve working conditions and skills [9, 13].

3.4 Radical Humanism



The radical humanist paradigm stems from the same intellectual source as the social relativist paradigm, as it regards the reality of the universe to be spiritual rather than material. However, the two paradigms have fundamentally different objectives. They share the belief, that the individual creates the world he/she lives in. But, where the social relativist is satisfied with understanding the nature of this process, the radical humanist is critical towards this process with particular interest in how it creates a sense of alienation in individuals [9]. In contrast to the radical structuralist, whose critique of society is concentrated upon structural relationships within a realist social world, the radical humanists base their critique of society upon the notion of consciousness [9].

In relation to the objective-subjective dimension of Burrell and Morgan's analytical framework, the radical humanist paradigm is in the subjective dimension [9]. This implicates, that radical humanists have a nominalistic ontology, believing that the social world external to the individual cognition consists only of names and concepts that are used to structure reality. The epistemology is that of anti-positivism, which means that the social world is relativist and can only be understood by being involved in the activities under study. In the subjective dimension, man is viewed as completely free-willed in the relationship between human beings and their environment. The methods favored for social analysis are ideographic, implying that the researcher can only understand the social world by participating and gaining first-hand knowledge himself. The subjective experience is emphasized [9].

In relation to the other dimension in Burrell and Morgan's analytical framework, radical humanism is placed on the sociology of radical change-axis. The sociology of radical change is interested in creating a change for the better. Essentially, this kind of sociology is concerned with man's emancipation from the structures which limits his potential for development; it focuses on both material and psychic constraints. Basically, it is concerned

> *"with what is possible rather than with what is; with alternatives rather than with acceptance of the status quo"*
>
> ***- Burrell and Morgan [9 p 17]***

3.4.1 Different Directions within the Radical Humanist Paradigm

Solipsism is here used with the same meaning as in the social relativist paradigm.

Within the radical humanist paradigm, there are several lines of work, which can be located within four different headings: Solipsism, French existentialism, anarchistic individualism, and critical theory. Common for these directions in social theory is a concern to develop a sociology of radical change from a subjectivistic point of view. Their objective is to free the human spirit, as they believe that the consciousness of man is dominated by ideological superstructures. Radical humanists seek to release man from the constraints that are placed upon him by existing social arrangements, that continuously both create and sustain man's everyday life. Their concern is to understand how this psychic domination of man occurs, so they can facilitate the development of man's potentialities. While there are varying degrees of subjectivism under this paradigm, all share the assumption, that reality is socially created and socially sustained. The radical humanists build upon the work of the young Karl Marx, who was concerned with the alienation of man [9].

In relation to the sociology of radical change-axis of Burrell and Morgan's analytical framework, the radical humanists place emphasis upon "*radical change, modes of domination, emancipation, deprivation, and potentiality*" [9 p 32]. Concepts as structural conflict and contradiction are not an issue under this paradigm, as they indicate a more objectivistic view of the social world [9].

3.4.2 The Critical Theory Branch

Hirschheim and Klein's interpretation of Burrell and Morgan's description of radical humanism is renamed to neohumanism, and they build almost solely on the branch in radical humanism called critical theory. We continue to refer to this paradigm as radical humanism, but we follow Hirschheim and Klein's elaboration of critical theory in an IS context. Their characteristics of information systems development in a radical humanist perspective follows Jürgen Habermas' work and particular his theory on the ideal speech situation [14]. Therefore, we start by giving a brief introduction to the work of Habermas, which is followed by a description of the implications a radical humanist perspective has on information system development.

Jürgen Habermas' Ideal Speech Situation

Jürgen Habermas is one of the younger representatives for the critical theory related to the Frankfurt School. Throughout his writing, he has been concerned with the conditions for a free and democratic dialogue in the modern and capitalistic industrial society, where a technical scientific culture is dominating, and the power is concentrated in large anonymous bureaucratic and economic control systems. Habermas' objective is a free and democratic dialogue, as well as reflection, which necessarily must build upon rational arguments [1].

Habermas is critical of this one-dimensional instrumental discourse present in modern scientific and capitalistic societies. He believes that rational arguments in a free democratic dialogue represent a possibility for breaking out of this one-dimensional straitjacket. He seeks to develop a more profound universal concept of rationality, that encompasses technical scientific rationality as well as moral and solidarity. A guiding principle for Habermas is to clarify, how a rational consensus between free citizens of modern society is possible [1].

Three Knowledge Interests

Habermas is concerned with three different knowledge interests, which he believes that everyone realizes and organizes their life according to. These three knowledge interests - technical control, mutual understanding, and emancipation - should be viewed as an alternative to positivism as they are the only theory of knowledge for both technical and social matters [1].

Each knowledge interest is related to a domain from which knowledge needs to be acquired, and each domain is related to a specific kind of knowledge. Below, we summarize the characteristics of each domain and its associated knowledge interests.

- The domain of work is associated with a knowledge interest in technical control. The logic of this knowledge interest is to free man from the coercion of nature. The aim is to increase technical control of nature and society and thereby overcoming obstacles to reproduction of the human species [14]. This technical knowledge interest is pursued positivistically.

- The domain of language is associated with a knowledge interest in mutual understanding. Language is fundamental for constituting and developing the social relations between people. Habermas perceives the history of mankind as a dialogue, that is flawed and distorted due to repression and power relationships. The social interaction takes place within institutions and norms, that are made legitimate through the use of langauge. These social relations cannot be understood positivistically; Habermas relates this kind of knowledge acquisition to hermeneutics and history.

- The domain of dominance is associated with a knowledge interest in emancipation. Language provides an opportunity for being critical and reflective. Habermas believes that it is the role of critical social theory to examine, when consensus on ideas are rationally grounded and consistent with general interests, and when consensus is a reflection of use of power, manipulation or other illegitimate power uses. The knowledge interest in emancipation is based upon the idea of the ideal speech situation, where agreement solely builds upon mutual appreciation of the weight of the arguments.

These three domains and associated knowledge interests distinguish between rational technical acts, in the domain of work, and communicative acts, in the domains of language and dominance. That is, society and the social life can be viewed from two different perspectives, where each has its own rationality and way of acting [1].

3.4.3 Radical Humanism in an IS Context

The development of information systems is governed by an interest in all three knowledge interests. It is important to gain knowledge from all three knowledge domains and to incorporate this knowledge into the information systems. Knowledge in the technical domain is necessary for considering various solutions and their effectiveness in relation to each other. In other words, technical knowledge is the basis for reasoning about different solutions, and then build the information system that can best improve organizational life, provide better working conditions, and perhaps raise the earnings. It is important to stress, that technical knowledge must always be used for the sake of both management and employees. The technical insight can never be used for achieving organizational conditions that are less favourable for the employees. That is why, the knowledge interest in mutual understanding and emancipation is particularly important. There has to be a social perspective on the process of system development and the information system to be built [14].

Legitimate system objectives are requirements that emerge from a free and open discussion. Hirschheim and Klein states it this way:

> *"The goal of information systems is to help with the institutionalization of an ideal speech situation which in turn validates a consensus about*

system objectives and modes of design and implementation."

- Hirschheim and Klein [14 p 1209]

The ideal speech situation constitutes the foundation for addressing all three fundamental objectives of information system development, namely improved technical control, better mutual understanding, and emancipation from social constraints and psychological compulsions towards a state of justice, freedom, and material well-being for all. It is the role of the system developer to facilitate an open discussion between all stakeholders. The system developer must take on the role as social therapist or emancipator, who through the open dialogue draws the different stakeholders together in order to create consensus [14].

It is important for the success of the information system to provide a basis for a free dialogue between stakeholders where arguments are appreciated according to their rationality. To do this, all stakeholders must meet and spend time reaching an understanding of each others' expectations for and interests in the system to be built. It is the job of the system developers to generate motivation and a positive attitude in all stakeholders, so they experience a joint commitment. The consensus must be evaluated according to a common sense, that values well-being for all - organizational, mental, and material. Providing the foundation for achieving consensus is a matter of removing conflicts, quarrels, distortion of power, repression, and communication barriers. This will result in all stakeholders listening and appreciating the rational weight of arguments instead of being prejudiced by pre-conceptions of the other stakeholders [14].

To summarize, when developing information systems, organizational life is changed. Therefore, gaining knowledge from all three knowledge interests is very important, in order to move the organization towards a better organizational life. In the radical humanist perspective, the information system is developed in order to remove distorting influences and other barriers to rational discourse [14].

4. Development Method Descriptions

This chapter contains a description of the two methods Scrum and OOA&D. First the ideas and concepts behind the methods; the agile- and plan-driven paradigm are presented, and then a description of the methods follows.

4.1 The Agile Approach

In the late 1990's several development methodologies have received much attention. New ideas on how to structure the development process have been presented. One of these approaches is agile software development, which is widely accepted today. The agile software development paradigm has a different focus compared to the plan-driven paradigm:

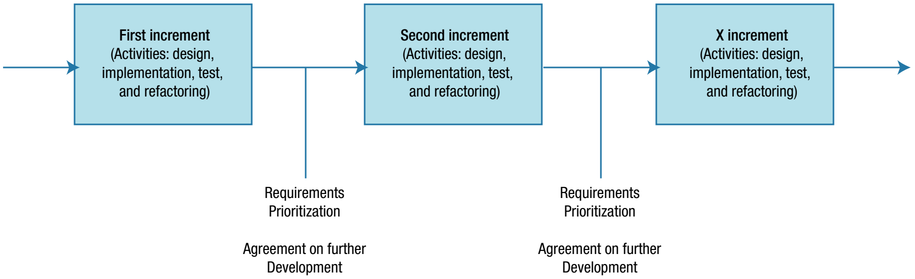
- Individuals and interactions over processes and tools.
- Working software over comprehensive documentation.
- Customer collaboration over contract negotiation.
- Responding to change over following a plan.

- Larman [19 p 27]

The plan-driven paradigm sees requirements management as a process which regards requirements elicitation as an up-front activity, where requirements are determined in the beginning of a project. In the agile paradigm requirements elicitation is seen as an integrated part of the process and requirements are therefore gathered throughout the process. Early requirements are considered as a starting point for a project, as the requirements are expected to evolve during the development process. Although some requirements are discovered later in the process than others, they are considered equally valid as the already discovered requirements. Requirements evolve through the first iterations during a series of workshops with stakeholders, users, and developers. For instance ninety percent of the requirements are determined after just three or four iterations [19].

By handling requirements in iterations, the agile paradigm prioritizes requirements in such a way, that high risk requirements are implemented and handled at the beginning of the project. When these risks are reduced it is claimed that it is easier to estimate a price on the development of the entire system; Hence, it is possible to present a fixed price and fixed time contract to the customer [19].

Figure 4.1: Our interpretation of the agile development concept.


First increment (Activities: design, implementation, test, and refactoring)
Second increment (Activities: design, implementation, test, and refactoring)
X increment (Activities: design, implementation, test, and refactoring)
Requirements Prioritization
Agreement on further Development
Requirements Prioritization
Agreement on further Development

As mentioned above, agile development is based on iterative development, which concerns development of systems in several steps. The basic idea is that a small part of a system is developed in each iteration and feedback is given to the developer, see figure 4.1. In modern methods an iteration has an optimal length of one to six weeks. Every iteration includes several disciplines, such as defining requirements, design, implementation, and tests. All are seen as essential stages in every iteration [19].

Iterative development requires that requirements, planning, estimates, and the product evolve and are refined in every iteration. It cannot be frozen in an up-front specification like in the plan-driven.

Description of Scrum

4.1.1 Scrum

Scrum is one of the latest agile development methods and is based on iterative development. Jeff Sutherland [27] invented many of the initial thoughts and practices prior to formalizing and commercializing Scrum with Ken Schwaber. The following sections are based on Craig Larman [19], Mike Beedle [25], and Ken Schwaber [25, 24].

The distinctive characteristics of Scrum are self-directing teams, daily team-measurements, and avoidance of prescriptive processes. The key practices can be summarized in the six bullets below:

- Self-directing and self-organizing team.
- No external addition of work to an iteration, once chosen.
- Daily stand-up meeting with special questions.
- Usually 30 day iterations.
- Demo to external stakeholders at the end of each iteration.
- Client-driven adaptive planning.

We have chosen to describe Scrum's use of documentation, formal steps, and reviews with the term ceremony, and its number and length of iterations with the term cycles. These terms are Larman's [19], and he uses them for setting up a two dimensional representation of various development methods. As it can be seen in figure 4.2 Scrum, with its 30 day iterations, has a longer iteration period than other agile methods, but has more frequent and shorter iterations than the sequential waterfall model. The figure also shows that Scrum is flexible on the ceremonial axis, as it falls silently in reference to how much or little ceremonial activities are needed for a particular project. The amount

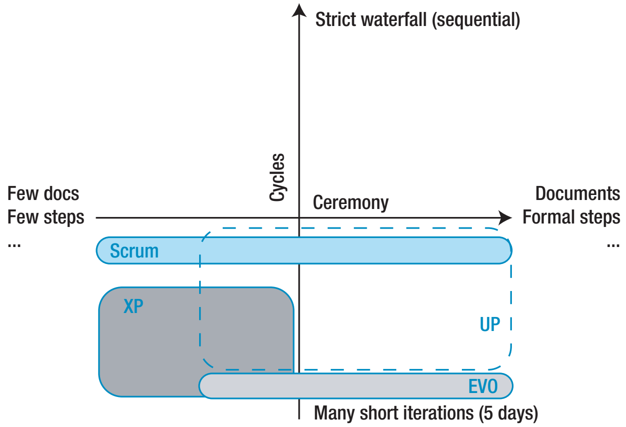


Figure 4.2: Scrum on the cycles and ceremony scale [19 p 110].

of ceremonial activities is determined by the self-directing and organizing teams.

4.1.2 Project Management in Scrum

As mentioned before, a fundamental premise of Scrum is that the scrum team is self-directing. The team consists of ideally seven members, where one of the members are chosen to be the scrum master. As few as five and as many as eleven members can be involved.

The development process is divided into 30 day iterations called sprints, where the team turns user stories into an increment of potentially shippable product functionality.

Morning Meetings

During a sprint, a daily scrum meeting is held for the scrum team. The morning meeting lasts for a maximum of twenty minutes. During this meeting the team members coordinate their work tasks and progress, and they report problems to the scrum master. The meeting also serves as a report on how the project evolves for the managers, as the meeting is considered more informative than reading a report.

The Scrum Master

The management of the developing organization is represented by the scrum master, who is responsible for implementing Scrum practices. In other words, the scrum master is responsible for the success of Scrum. It is also he who knows and reinforces the visions and goals of the project. Therefore the scrum master is a mediator between the management and the scrum team.

An issue of great importance is that the scrum master does not tell the scrum members what to do. He merely serves as a facilitator whose job is to eliminate problems that keep, team members from meeting their daily goals. The scrum master serves as a firewall, who ensures the scrum team is not interrupted by external issues, such as additional work requests from the customer. In his assignments development tasks are included.

The Product Owner

The product owner or customer is represented by one person who is in charge of creating and prioritizing the product backlog. Furthermore, he chooses the functionality which should be implemented during the next sprints and reviews.

Pigs and Chickens

In Scrum, a team member is committed to the project, and has a direct influence on the project. Such a person is referred as a pig. A chicken is, on the other hand, the definition of a person who is interested in the project, but has no formal scrum responsibilities and accountabilities. He is only allowed to observe but not to speak and interfere during iterations. An example of a chicken could be a member of the management, who is not involved in the project, but is interested in the status of the project.

4.1.3 Requirements Management in Scrum

Scrum contains a number of activities which relate to requirements management. More specifically they relate to, how requirements are documented, prioritized and validated and are done through the following activities:

Sprint Planning Meeting

At the beginning of each sprint, a sprint planning meeting is held with the product owner, users, and the scrum team. The meeting is headed by the scrum master, and is divided into two segments, each of four hours duration. In the first segment the product owner defines what part of the product backlog should have the highest priority. Together, the scrum team and the product owner decide how much and what parts of the backlog are possible to implement until the upcoming sprint. The result of the first segment of the meeting is a commitment from the scrum team, to implement a certain chunk of the backlog. The second segment is concerned with, in detail, planning and specifying the work to be done. This is documented in a sprint backlog. At the second meeting only the scrum team and the scrum master are present.

Pre-Game Planning Meeting

At the beginning of a scrum project a pre-game planning meeting is held, which essentially is the same as a sprint planning meeting. The only difference is that the product owner creates the product backlog, from which the system is developed. Otherwise, the meeting has the same structure as a sprint planning meeting.

Product Backlog

The product backlog is an evolving, prioritized queue of business and technical functionality that needs to be implemented into the system. The backlog is created by the product owner, who specifies what requirements the system should fulfill. The backlog is specified into further detail at the pre-game planning meeting, where all technical questions are discussed.

The product backlog is not a static backlog, as it evolves with the customers‘ needs. Every change performed in the requirements to the system is documented in the product backlog. The backlog thereby acts as a product history, where every change is registered.

Sprint Backlog

This backlog specifies which requirements are to be developed in the ongoing sprint. It includes time estimate documentation and responsibilities. The requirements stem from the product backlog, the last product increment and the capabilities of the scrum team. The scrum team meets with the product owner, management, and users to determine what functionality should be implemented in the next sprint. These requirements are only concerned with the current sprint, and no new requirements can be implemented during a sprint. In other words, the sprint backlog is locked during the sprint.

Prioritizing Requirements

At each sprint planning meeting, requirements are prioritized after highest risk by the product owner. This way, the requirements are clarified by the customer and revised after each workshop. The basic idea is, that after three to four iterations at least ninety percent of the requirements are determined and the highest prioritized requirements are handled in the beginning of the project. Thereby large changes will most probably not occur at the end of the project, and would therefore not result in further extensive expenses.

Sprint Review

To assure that the optimal system is developed, the increment of the system is evaluated as a conclusion to each sprint. Therefore, the prototype is the documentation of the project, rather than the written documents in the product and sprint backlogs.

A sprint review lasts for four hours, and is headed by the scrum master. The reviewers consist of project stakeholders, managers, developers, customers, and sometimes sales and marketing people. The decisions made at the sprint review can cause changes to the existing version of the system. These changes are then handled in the future sprints through code refactoring.

In the next section the plan-driven paradigm and the method OOA&D are presented.

4.2 The Plan-Driven Approach

In the plan-driven paradigm requirements elicitation is seen as an up-front activity, which is conducted in the beginning of the development project. The requirements found are then turned into design documents and different specifications.

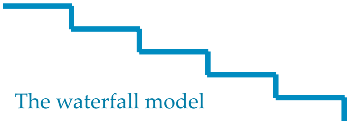
Development methods within this approach can be divided into two groups: Process models and software process iteration models, which are shortly described in the following sections.

4.2.1 Process Models

The process for developing software has evolved to exploit the capabilities of the people in an organization and the specific characteristics of the systems, which are being developed. Many different processes exist, but they all contain four fundamental activities, which are:

- Software specification.
- Software design and implementation.
- Software validation.
- Software evolution [26].

An example of a process model is the well known and widely used *waterfall model*. The model is divided into distinct stages, and therefore commitments must be made at an early stage in the process. This factor makes it difficult to respond to changing customer requirements, which is why the method must only be used in projects, where the requirements are well understood [26].



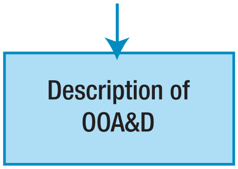
The waterfall model

4.2.2 Software Process Iteration Models

Spiral development

For large projects it is often necessary to use different approaches for different parts of the system, which calls for a hybrid model. The essence is that in iterative processes the specification is dynamic and changed as the system is developed. Therefore, there is no contract between the developer and customer compared to the waterfall model. An example of a software process iteration model is *spiral development*. Each loop of the spiral represents a phase in the software process. The inner loop may be concerned with system feasibility, the next with requirements definition and so forth. The most important difference between the spiral model and other models, is the explicit consideration of risks in the model.

4.2.3 OOA&D



We perceive OOA&D as a plan-driven development method because of its emphasis on analysis, its high degree of documentation, and its extensive use of tools and procedures in the analysis and design processes. OOA&D is an approach to analyze and design software systems, and it is a method which consists of different tools and principles that are derived from other well tested methods. The following sections are based on Lars Mathiassen et al. [21]. The method is built upon four principles that forms the basis for analysis and design of computerized systems:

- Model the system's context.
- Emphasize architectural considerations.
- Reuse patterns that express well-established design ideas.
- Tailor the method to each development situation

- Mathiassen et al. [21 p 4]

Furthermore, the method uses two key concepts, namely *objects* and *classes* to describe the context of the system. An object is an abstraction of a phenomenon in the context of the system. An object can for instance be a user of the system, and can be grouped into classes, which are a collection of objects that have the same structure, behavioral patterns, and attributes. This gives the opportunity to make a common description of objects instead of individual descriptions.

A key principle in OOA&D is that the success of a system depends on the developers ability to understand the systems' practical application. The method perceives the system context from two different angles, namely: *The problem domain* and *the application domain*. The problem domain describes the context of the system that is administrated, monitored or controlled by a system. In other words, the parts of the reality that the system effects. The application domain describes the organization that administrates, monitors or controls a problem domain.

The method stresses two key tasks when modelling the system context. The first is to model what the finished system will administrate, and the second is to model how the users of the system will interact with the system.

In figure 4.3 the four main activities in OOA&D are shown. These activities cannot be seen as separate activities, which can be carried out sequentially. The analysis and design process are on the other hand seen as iterative as considerations from one perspective can yield new considerations based on another perspective.

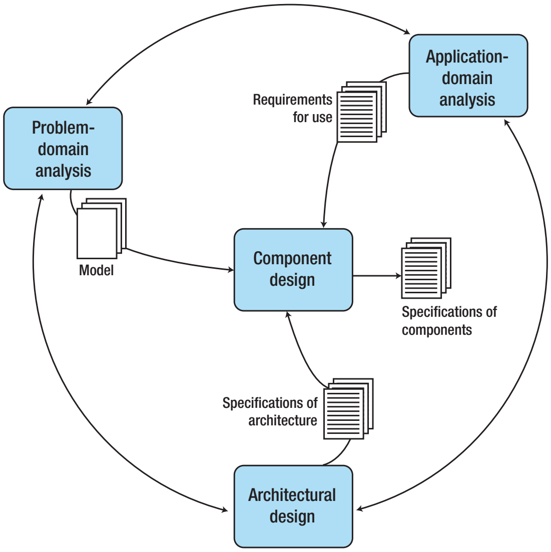

Application-domain analysis
Requirements for use
Problem-domain analysis
Model
Component design
Specifications of components
Specifications of architecture
Architectural design

Figure 4.3: Main activities and results in object-oriented analysis and design [21 p 15].

OOA&D is a systematic method for object oriented analysis and design, but as the four main activities are abstract tasks, the method is flexible and can be changed to fit the developing organization and project. For instance either a plan-driven top down or an use-case approach can be employed. In other words, the method sketches out several stages to go through in developing systems, and it is up to the practicing developer to design the method in such a way, that it supports the organizational context of development.

Documentation

The result of the analysis and design process is a detailed documentation. This document describes the analysis and design process, and can be used as a reference tool for the developers. Furthermore, it serves as contract, which contains the agreements made between the involved participants in the development process.

The analysis document should be clear as it forms the basis for a requirements specification. This specification is a formal written agreement between the users and developers. The design document has the purpose of functioning as a frame for the programmers. Therefore the document is preferred to be brief and precise in identifying the system components, their structure, functionality, and their interfaces. The method prescribes the areas where uncertainty is high and more detailed descriptions are needed.

The transition from analysis to design is fuzzy in the method, but it is made clear, that changes to the analysis document is not allowed. Instead a supplemental document is made to the design document, that describes what has been changed during the design process. This document is called a correction document.

Architectural Design

OOA&D is a method that emphasizes the importance of a powerful architectural design of a system. This is obvious from the following quote:

> *"Successful systems are distinguished from less successful ones by a powerful architectural design."*
>
> ***- Mathiassen et al. [21 p 173]***

The method sees system architecture as a means for satisfying certain design criteria, while it also forms a framework for future development activities.

Tools

In this section we describe some of the specific tools that OOA&D provides for reaching, what the authors call good documentation.

Rich Pictures: A rich picture gives an understanding and a quick overview of the situation in question. The pictures are drawn by the development team and focuses on important aspects of the situation. The picture can either describe a situation that is stable or a situation that is under transformation. This again depends on what the interpreter sees when he visits the user organization to gather information.

Class Diagrams: A class diagram describes how objects and classes are related as well as the structure of the problem domain. A class model should be both abstract and concrete as it describes both general and specific relations. The outcome is a map of the problem domain, where it is possible to see which objects are related.

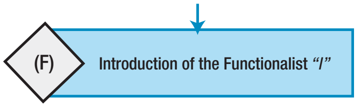
Statechart Diagrams: Statechart diagrams are used for describing the dynamics within the problem domain. The system is to function in a dynamic reality, where it is necessary to obtain an understanding of how it changes over time. The changes are expressed by adding behavioral patterns to each class in the class diagram. The pattern consists of an event trace, which is a sequence of events involving a specific object. This pattern is unique for each class and defines a legal sequence of events for each class.

Use Cases: Use cases are used for determining the interaction between the users and the application domain. These use cases should result in a complete list of usage requirements for the system. In an use case an actor is involved, who can either be an abstraction of users or other systems that interact with the system. The resulting use cases describe patterns for interaction between the system and actors in the application domain.

Functions: As opposed to the use cases, the functions determine what the system should do rather than how the system is to be used. The purpose of defining the functions of the system is to determine the system's processing capabilities. Furthermore, it is the functions, which are valuable to the users. There are many types of functions for a system and each describes a specific relation between the model and the context of the system. The list contains the functions in the system and should be complete and in agreement with the use cases.

From this stage the interfaces are designed according to the allocated functions. These consist of user interfaces and system interfaces. This is a process that we will not describe in further detail.

5. A Complex Project is Proposed



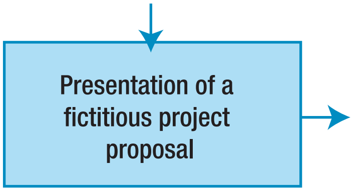
In the previous chapters you should have gained a theoretical insight, which is the foundation of the following journey. Therefore, it is time to introduce myself. I am the narrator, who will guide you through the journey of developing a strategy for the fictitious proposed project. Further, I am a project manager in a leading software company and my approach to software engineering has its roots in the functionalist paradigm.

At this very moment, I have been put into a problematic situation, because I recently received an interesting, but complex project proposal, which I am soon going to explain. I find it critical and challenging, as I have to prepare a solid and realistic development strategy for this project. My main concern is what development method to choose for the project and I have the choice of choosing either Scrum or OOA&D. Both of the methods clearly have strengths and weaknesses, but I am wondering if my competencies and understanding of the two methods is extensive enough to complete this project successfully. This is why, I have chosen to involve three of my friends, project managers, who can help me make this critical decision. My friends are going to evaluate both the methods and afterwards point out the main risks, they see in the project proposal. I will let the project managers introduce themselves in the end of this chapter, because it will give you a more clear understanding of the reasoning behind their reviews.

With the different reviews in mind, I choose risk analysis as a tool to assist me in deciding, which development method I should choose. I am aware that I probably have *to adapt the method to make it fit the project, which gives me the opportunity to present the concrete development strategy for you.*

Before presenting the project in more detail, I will shortly describe my role and vision as a project manager. My role in the developing company is to manage a group of system developers. I see myself as the expert, who takes system objectives and turns them into an implemented information system. My ideal is profit maximization and quality assurance, as the organization's primary goal is to maximize the stakeholders' wealth. Therefore, the system must contribute to more profitability, and in order to do that, I focus on planning. I am of the conviction that system objectives should be specified by the management; namely the Danish Ministry of Education in this concrete project proposal.

5.1 The EIS Project Proposal



The project is proposed by the Danish Ministry of Education and is a centralized information system for ten institutions placed at different locations in Denmark; see figure 5.1. At this very moment each institution has their own system for managing student data; such as grades, profiles, course descriptions and such. The Danish Ministry of Education has proposed this project, as they see the importance of centralizing student data. A centralization would make it possible and easy to perform statistics and analyses in between and across different institutions. By centralizing the data it will ease the possibility of making a nationwide comparison of student data. The EIS system, which it henceforth is called, should help the institutions with organizing data and making the student data comparable to other similar data.

The proposal is based on a real development project, which has been studied during our 9th semester, see Appendix A.

EIS - short for Educational Information System.

The project has an initial time span of three years, and the system should be put into use during the summer holidays. The Ministry is the main authority, and has decided to divide the development costs equally between the institutions. It is further requested that the development project has a fixed price contract for the initial three years. The Ministry prefers to keep a tight status of the project, and wants continuous reports on the progress of the development.

It is suggested that each institution assigns one super user, who should be the user representative in the development project. This user representative has the responsibility of involving users from their own institution and make decisions accordingly. These users constitute the group of user representatives.

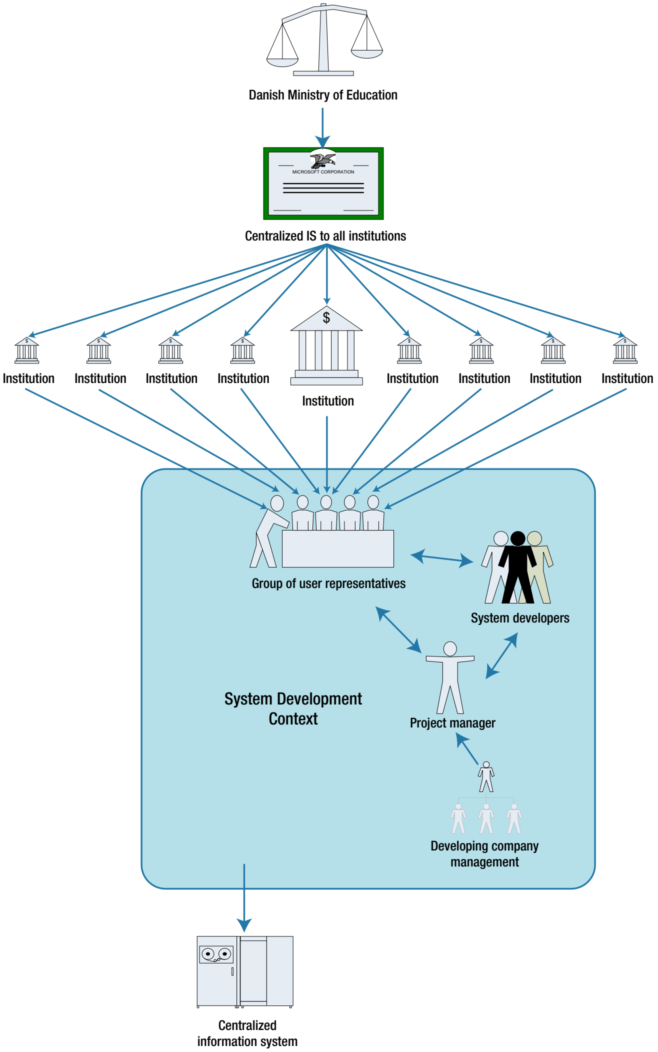

Danish Ministry of Education
MICROSOFT CORPORATION
Centralized IS to all institutions
$
Institution
Institution
Institution
Institution
Institution
Institution
Institution
Institution
Institution
Group of user representatives
System developers
System Development Context
Project manager
Developing company management
Centralized information system

Figure 5.1: Illustration of the project proposal.

I will shortly describe the involved participants in the EIS project.

Group of User Representatives

The group has the responsibility for prioritizing requirements during the development process. The super users act and prioritize according to their own institution's premises and study programme. Further, it is worth noticing, that the budget has a certain influence on the prioritization of the actual implementation of the requirements. The user representative do not have the authority to finally agree on the requirements prioritization, as this has to go through the finance manager in their institution.

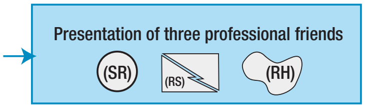
System Developers

The system developers' role is to advice, design and implement the system. They are further an essential part of the process of eliciting requirements and presenting them for the group of user representatives.

System Project Manager

As I am the project manager, I have the responsibility of involving and developing the system in accordance to the customer. My main concern is to make a satisfactory system within the given economical scope. I have a large responsibility in involving and advising both the management and users of the system during the entire project, in order to establish a trustworthy cooperation.

5.2 Presenting the three Project Managers



Now my three friends are going to introduce themselves and their views on the purpose of systems development.

I, the Social Relativist, as a Facilitator

My role, as a project manager in the development of an information system, is that of a facilitator, which means that my job is to help the users make sense of the new system and the environment in which it is going to be used. I see the world as a socially constructed reality amongst individuals, and in my view, every opinion is equally valid and every subjective thought carries the same weight.

When I develop systems I focus on the process rather than the actual end-product, because the system is constructed during the development process as a product of the minds of the involved stakeholders. Therefore, the system is developed from the users perspective and much attention is given to social change.

My primary goal is to develop systems which support the skills and subjective opinions of the users. This is achieved through co-operation and reaching consensus among the people involved.

I, the Radical Structuralist, as an Implementer

As a member of the management my primary role is to act as an agent of the management and focus on rules, regulations, hierarchy and the overall structuring of the system as well as the organization.

The organization, in which I am developing a system, is split into antagonistic groups of stakeholders, managers and employees. Even though, I know there is an existing conflict, I do not see it as my job to function as a facilitator or as a means of communication.

As a project manager my concern is to manage the development team and assure that the goals of the management is reached. Moreover, I believe that I can develop a system by only involving the owners as the representatives of the users, because I consider them as qualified to define the system objectives.

My primary goal is to assure that the productivity is increased and the management are gaining economical benefits and managerial control through structure, rules and regulations.

I, the Radical Humanist, as an Emancipator

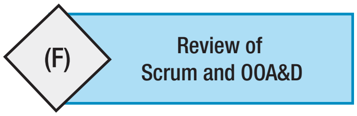
I see the purpose of system development to be a process of identifying a consensus on system objectives and the requirements that are to be implemented. These rational requirements should emerge from a free and open dialogue between all stakeholders. In the end, the system should provide emancipation from all kinds of social and psychological constraints and compulsions, resulting in a state of justice, freedom, and material well-being for all stakeholders.

I put emphasis on how developing an information system could improve organizational life; I am much less interested in the actual state of the organizational life. I see my role in system development as that of an emancipator or social therapist. It is my job to facilitate a free and open discussion between all stakeholders, where they - by appreciating rational arguments - can agree on setting a course for a better organizational life.

My primary goal is to improve the dialogue, reach a common understanding of the system, and assure that the users as well as their needs are visualized and given attention.

6. Scrum or OOA&D: The Functionalist Opinion

Now you have an idea of which aspects of system development I regard essential in order to achieve success; and which aspects my three professional project manager friends value and regard important when developing an information system. In the following, I present my evaluation of the two specific methods: Scrum and OOA&D. I describe the opportunities and limitations I see in Scrum and OOA&D. This gives me an overview of the pros and cons of each method, which will be useful when estimating which method suits the project proposal better.



6.1 Scrum

Scrum is a method where the focus is primarily on prototyping. With prototyping a great deal of user involvement is necessary, and Scrum suggests iterations with the length of a month. After each iteration, reviews are held with the customer, and the scrum team holds planning meetings. In general, Scrum has many great features, but it has little focus on written documentation and validation of requirements. This, I find risky, as I want to ensure the validity of the system in relation to the customer.

I the following I will describe my perception of the activities in Scrum and outline the strengths and weaknesses I see in this method.

Sprints - Only a Tool for Management

Scrum embeds the requirements in the product, which I find risky, as this do not guarantee the validity of the system. Scrum suggests two types of tools for managing requirements: Product backlog and sprint backlog. I see the two tools as an interesting and fairly simple way of dealing with prioritization of requirements. The backlogs are a nice tool for organizing and managing the work of the scrum team. However, Scrum does not describe the details of the requirements in the logs. This I find critical, because if I do not know the specific details of what to implement, how am I suppose to carry it out? This raises another interesting issue about Scrum, as the method focuses on implementing prototypes. I would prefer to analyze the problem area in the very beginning of the project and design models and specifications accordingly, as I believe it will make it easier to implement the system.

As a project manager, I find the sprint planning meetings essentially good. In my opinion the division of responsibility, where the stakeholders prioritize and define according to the product or release backlog, stands well with my overall principle of how requirements with the highest business value should be prioritized. I see a problem in refining and reprioritizing requirements at every sprint review, as the system to be developed could become fragile to constant changes. And these changes could influence the overall system architecture. Despite of this I like the fact, that there is only one person, the product owner, who is in charge of prioritizing requirements.

The sprint review has the purpose of demonstrating the development of the system for the customers. Basically, I like the idea but, I am a bit uncertain about the outcomes of the meeting. At a sprint review, changes might occur based on the prototype shown. These changes might have an influence on my planning and development of the system, which I am not fond of. Moreover if these changes occur after each iteration it will cause continuous breaks in my development planning. Prototypes only show a current stage of the development and therefore only fulfill the current requirements. There is no guarantee that the system actually contains the functionality and produces the correct data, as it is only a prototype.

Strict Planning - But Loose Organization and Management

As I see it, the scrum meetings are very formalized and time restricted. Only a couple of questions are allowed to be discussed in the meeting. This way a short status on the development of the system is held every day. The scrum master's work task is evenly divided in system development and project management assignments. He is not seen as a person in charge but as an equal team member. At the scrum meetings the scrum master is present with the group consisting of seven people. He is responsible for updating the sprint backlog and ensure that the estimations are up-to-date. The fact, that other developers and project managers are allowed to attend, but not speak at the meetings, is interesting as it eliminates unnecessary disturbance in the meetings. I find this rule very convenient and practical.

I like the fact that the scrum meetings are time restricted and consider them a good method for keeping track of the project. My concern regarding the project manager is that he is not given much responsibility and seen as an equal team member. I think that the scrum master should be managing and directing the overall process.

Scrum argues that it is not necessary to document and specify before implementing. This means, knowledge is stored in the developers' heads rather than in documents. In my opinion, this makes development vulnerable to changes in personnel resources.

In a scrum project there can be as many teams as necessary. The key point is, that the scrum master from each team meets with the other scrum masters. Concrete guidelines to this is not presented in the method, as the method is described on a more general level. I would find it complicated to have scrums of scrums in a project, which involves 500 developers for instance. The more developers, the more knowledge is created, and no guidelines are given on how and where should the knowledge should be kept.

Restricted User Involvement Controlled by the Product Owner

Scrum prescribes, that the customers should be involved once a month at the sprint review. I like that fact, that user involvement can only happen once a month; this restriction limits the troubles of requirements negotiation to the reviews. Further, it is practical that the product owner has the final word during negotiation; that way, negotiation will never enter a dead-lock. The drawback of the reviews is their continuous frequency, and the fact that they do not result in any formalized description of user needs. Due to the superficial nature of the stakeholder evaluation of the presented prototype, it will only be the user interface which gets evaluated; it will not be possible to evaluate the implemented functionality.

Lack of Documentation and Specification with Prototyping

The fact that Scrum elicits 80-90 percent of the requirements during the first three to four sprints, I find very useful. The idea of using prototyping to elicit requirements could be very useful to gather the system requirements from the users.

As 80-90 percent of the requirements are elicited at this relatively early point in development, it is possible to estimate and plan the project from this point onwards. I see a need for further specification of the requirements, as it is possible to model the entire system at this stage. Dan Turke and Bernard Rumpe states:

> *"The assumption that code refactoring removes the need to design for change may not hold for large complex systems in particular. In such software, there may be critical architectural aspects that are difficult to change because of the critical role they play in the core services offered by the system. In such cases, the cost of changing these aspects can be very high and therefore it pays to make extra efforts to anticipate such changes early."*
>
> ***- Turke and Rumpe [28 p 3]***

From this point and onwards, I do not necessarily see the need for user involvement every month, only if there is a need for specifying gathered requirements in detail.

One of the essential topics in Scrum is refactoring, which is done continuously during the development. The purpose of refactoring is to ensure that the system fulfills the requirements stated. I think the principle of refactoring is important, but the use in Scrum is not optimal. Refactoring only ensures that the code performs correctly, but this gives no guarantee that the requirements have been met. Barry Boehm mentions another important risk factor with refactoring:

> *"However, empirical evidence indicates that with less-than-great developers, refactoring effort increases with the number of requirements or stories. For very large systems, our Pareto analysis of rework costs at TRW indicated that the 20 percent of the problems causing 80 percent of the rework came largely from* ***architecture-breakers****, such as architecture discontinuities to accommodate performance, fault-tolerance, or security problems, in which no amount of refactoring could put Humpty Dumpty back together again."*
>
> ***- Boehm [4 p 3]***

	Strengths	Weaknesses
Requirements elicitation	▪ Limited user involvement ▪ Product owner in charge of requirements elicitation ▪ Prototyping as a technique for requirements elication	▪ Too frequent user involvement after intial 3-4 sprints ▪ Vague formalization of user needs
Requirements negotiation	▪ Product owner in charge of prioritizing requirements ▪ Requirements negotiation is limited to reviews ▪ Requirements negotiation controlled by the product owner	▪ Too fequent reviews
Requirements documentation		▪ No detailed specification of requirements ▪ Knowledge not kept in documents ▪ Requirements are embedded in the product
Requirements validation		▪ Refactoring not a proper way of validating ▪ Only the user interface gets validated at the reviews - no guarantee that the system actually contains the required functionality and produces the correct data
Requirements process management	▪ Backlogs are good for organizing work ▪ Time-restricted morning meetings are good for keeping track of work assignments ▪ Convinient and practical that only members of the scrum team can speak at the morning meetings	▪ Prototyping during the entire process is not beneficial or effective ▪ The scrum master has low degree of formal power ▪ Changing requirements are welcomed during the entire process ▪ No clear division of responsibilities in the scrum team ▪ Vulnerable to changes in personnel resources, as there is no documentation of knowledge ▪ This method makes it impossible to design and model the system early in the process

Table 6.1: Strengths and weaknesses in Scrum from a functionalist perspective.

6.1.1 My opinion on Scrum

I see good features in Scrum for supporting and tracking requirements during each Sprint. I see morning meetings and reviews as a good way of keeping track of the system. But the method is also vulnerable, when it comes to documentation. The correctness is based on what the users can see, and not on the system architecture.

I find the fact that there is only one product owner very beneficial for the development process. It helps the process of agreeing on requirements, and makes less interruptions in this process. I have summarized my opinion on Scrum in table 6.1.

6.2 OOA&D

I see OOA&D as a method which offers a wide toolset for analyzing, modelling and making design documents. The method is based on two activities, namely analysis and design. Even though the analysis and design phases in OOA&D are described as sequential activities, it is possible to use OOA&D both incrementally and iteratively.

The method does not prescribe a desired degree of user involvement. In the process of modelling and designing the problem domain I would draw on user experiences, but I would only involve them when I felt the need for it. I believe that the ideal time to involve the users is in the beginning of the project. It gives me the time to model and design according to the requirements, and further to write it down in a formalized way.

Documentation - a Quality Insurance

A great way of modelling and designing the hierarchy of objects is by presenting objects in class diagrams. This will provide a clear illustration of the relations between the objects and their inheritance. With the statechart diagrams the behavior of the objects and their states, in a given time frame, can be described. This type of documentation can help me to ensure the validity of the system in relation to the stakeholders.

If changes should occur during the process, OOA&D prescribes not to append them in the original design document. Instead I should make a correction document containing every change made during the process. The original design document is not changed during the process, which ensures that I always know what I agreed on with the customer. This way, I ensure that the result of negotiated requirements is clear.

I like the fact that all knowledge about requirements is embedded into documents, which are standardised via the method itself. This helps me to keep track of and validate the requirements.

Objects - The Concept for Modelling and Understanding the Problem Domain

OOA&D separates problem and application domain, making it easier to focus the analysis. In OOA&D the problem domain is modelled by means of objects representing real world objects. This way, the mapping from real world artifacts to technical design is eased, and the process of understanding is made clear. By using objects and relations, events and behaviors between and within the objects, can easily be visualized. Modelling requirements like this, makes it easy to communicate with the users, as a class diagram is an unambiguous representation of the real world. Therefore, I see several advantages in modelling the world with objects.

Before the problem domain analysis is made, a system definition should be written. The method suggests that I describe the problem context with the help of rich pictures. A rich picture describes the context in which the system should interact. In a rich picture, users, tasks, and objects in the context should be present.

Focus on Product Quality Instead of Comprehensive Stakeholder Involvement

The method is not specific regarding the amount of user involvement, and it does not state who and how many users should be involved. This fact gives me the possibiliy to decide whom to involve and when. I need the users to describe the requirements and problem domain once in the beginning; afterwards I am able to model and design the solution according to their needs. Actually, I think that this method does not enhance the need for negotiation, as the problem area is modelled like it is. In any case, disputes are resolved during analysis and modelling, so implementation is not complicated by changing requirements.

Use cases are a tool for describing requirements to the system. Use cases are good for describing the interaction between the user and the system. By making these use cases, I am able to make generalizations and model the behavior with statechart diagrams. When generalizing and finding patterns in the

Table 6.2: Strengths and weaknesses in OOA&D from a functionalist perspective.

	Strengths	Weaknesses
Requirements elicitation	▪ Possibility of doing up-front elicitation and analysis of requirements ▪ The object-oriented philosophy is very practical for representing the real world ▪ Tools and techniques for making generalizations in the problem domain ▪ Developers decide the degree of user involvement ▪ Rich pictures a nice tool for understanding the problem context ▪ Separation of problem and application domain makes analysis easier	
Requirements negotiation	▪ Any disputes are resolved early in the process	
Requirements documentation	▪ Developer model and design the system on the basis of specified user needs ▪ Objects and class diagrams are great for modelling ▪ Requirements are described in documents standardized by the method ▪ The object-oriented philosophy makes mapping from real world to technical design easier ▪ By making generalizations through the object-oriented design it is possible to design a scalable system ▪ Use cases are good for describing the interaction between users and the system ▪ The original design document is not changed during the process, which ensures that I always know what I agreed on with the customer	
Requirements validation	▪ Documentation can be used to validate against	▪ Lack of specific techniques for validation
Requirements process management	▪ Possible to employ plan-driven development process ▪ Documentation makes planning and estimations possible	▪ Lack of specific tools or techniques for planning, estimation and organizing work tasks

design, I can reuse and make general abstractions of what should be implemented. By making generalizations I can design the system to be more scalable and not vulnerable to extra functionality.

Possibility for Up-front Analysis and Successive Planning

OOA&D provides opportunity for up-front analysis and phase-driven development. This makes it possible to plan the entire project according to the analysis and design documents. Approaching a project like this is usually the better way in my opinion, as there is always a deadline and a budget to be kept. With no planning the deadline and budget will slip. The drawback of OOA&D is its lack of techniques and tools for the actual validation, planning, and estimation.

6.2.1 My Opinion on OOA&D

I find OOA&D interesting, as it is a method with comprehensive guidelines for modelling and analyzing problem domains. I like the high degree of documentation and specification as it can ensure a clear agreement between me as a developer and the customer. Furthermore, the method does not prescribe a certain amount of user involvement, which is positive from my point of view, as I can decide when to involve them and how much. I have summarized my opinion on OOA&D in table 6.2.

6.3 Comparison of Scrum and OOA&D

In this section I will pinpoint the differences I see in Scrum and OOA&D. Further, I will point out the important strengths and weaknesses of using each method.

I have previously summarized the two methods in table 6.1 and 6.2. Looking deeper into these tables, reveals that one of the significant differences between Scrum and OOA&D is the degree of documentation. Scrum uses prototyping as the primary medium for documenting requirements, while OOA&D uses a variety of different models and descriptions to document the requirements. I find the lack of written documentation risky in a development project, as documentation helps the company keep important knowledge available for all developers. It further eases the discussion between the customer and developers, as the specifics of the system can be pinpointed.

Another issue is validation of requirements. Scrum only validates through prototypes, which I find very risky, as there are no guarantees for an operating system. With prototyping only the graphical presentation of what the system does is validated; the operations executed behind the user interface are not validated.

When discussing the involvement of users, the methods prescribes different strategies. Scrum formulates user involvement more precise and strict compared to OOA&D which has a very open approach to user involvement. OOA&D does not mention how and when to involve users. Instead it focuses on modelling the problem and application domain. I find Scrum's amount of user involvement too extensive, especially when involving them during the entire process in each iteration.

Another important issue is how a common understanding is reached of the system. In Scrum, user stories are considered as an appropiate approach, whereas, OOA&D models the domain and context with rich pictures, class diagrams, and statechart diagrams. I see a need for a more structured and formalized requirements elicitation technique in Scrum. I do not find user stories sufficient for describing the problem domain, instead they are useful for getting the users to describe their needs. The more detailed and precise the problem can be described in the beginning, the easier it will be to develop the optimal solution. That is why I find OOA&D more suitable than Scrum, as it has a more defined and formalized technique for dealing with this issue.

7. Three Different Reviews of Scrum and OOA&D

After my review of Scrum and OOA&D, I will now pass on the word to my three professional friends, as I am interested in learning their view upon the methods. Each of them have highlighted different areas within the methods, which they regard important. First, the three reviews of Scrum are presented; then, the three reviews of OOA&D are presented. Each review is summarized in a table to provide an overview.

7.1 The Social Relativist's Opinion on Scrum

(SR) Review of Scrum

Scrum is a method where planning is not in focus. In the beginning of a project an initial planning meeting is held, but this is only to start off the project and outline the main objectives. This stands well with my view of an ever evolving and changing environment, which makes it impossible to plan and document an entire project with an up-front strategy. What little planning is needed, I handle at the daily scrum meetings, during sprint reviews, and product backlog re-estimations.

Some requirements are more vague and fluent than others.

> *"No longer do you fire and forget requirements and the move on to the next phase. Requirements may be introduced, modified, or removed in succesive iterations."*
>
> ***- Favaro [12 p 16]***

A property I like about Scrum is that it sets out to implement the more risky requirements. This feature gives the chance to remove high risks in the very beginning of the project, and it facilitates a basis for common sense making.

Scrums Lack of User Involvement

At the initial meeting between customer and developers, a user and costumer representative called a product owner is present; he sets up the initial requirements for the system. This is something I like as a project manager, because then I develop what the customer wants. These requirements define what the scrum team develops. However, I see problems with the role of the product owner. The product owner has the final responsibility for defining what should be developed. In my opinion it would be ideal if the users were involved in the entire process [11], as they are the ones who should define the system.

The amount of users involved in the process is not enough according to me. Every person has his own subjective opinion of the system, therefore more users have to be involved to make sure it is the correct system that is developed; this is supported by Hisayuki Horai [15]. Preferably, all the intended users of the finished system should be involved. Furthermore, Scrum does not facilitate that the owner organization should define the system requirements together. Scrum simply states that one product owner is responsible for creating the product backlog. Other stakeholders are welcome at the sprint reviews, but it is the product owner who has the final word. This I find problematic as this neglects the property of subjectivity, which is inherent in my world view. One has to remember that my role as a project manager is to reach a common acceptance of the system, and I cannot se how this is possible without user interaction.

The Composition of the Scrum Team

The scrum team consists of seven members, where I am the leader, called the scrum master. I find this composition both good and problematic. The overall idea is that everybody in the team is equally important, which is in harmony with my fundamental beliefs.

The scrum master is not to act as a leader in the traditional sense, the role of scrum master is more to act as a facilitator. My job would be to serve the members of the team; and my most important job would be to remove obstacles and act as a firewall between the team and the "outside world". In my view, a scrum master should not shield the team from the environment, but facilitate a dialogue during the process of developing the system. I accept the necessity of making sure that the team can work in peace and quiet, but what good is it if the environment, for which they develop the system, changes during the process? Scrum uses the term pigs for members of the scrum team, and the term chickens for outsiders. In my view this notation is definitely not useful, as everybody who is a part of the environment is equal and should be able to influence the development process.

I must admit that, in my point of view this particular composition is problematic, as the users have little possibility of gaining influence. It is my belief, that a high degree of user involvement is the only right way to develop successful systems. According to me the system should be developed from within the users' perspective, and the process should facilitate their sense making process.

> *"From the above discussion, we can say requirements process is a process which treats inconsistency. This inconsistency arises from divergence of viewpoints of participants who live dispersively."*
>
> ***- Horai [15 p 175]***

Therefore, they should be involved constantly and on an equal basis like the developers. In Scrum one user representative is responsible for prioritizing and checking, whether the system meets the requirements of the users. User are only involved at the initial sprint planning meeting, during the sprint review, and the re-estimation of the coming sprint. This is a step in the right direction as the users are kept informed. However, in my view the users should be an integrated part of the development team, as permanent members of the scrum team. It would facilitate a common sense making process for all involved parties, which is the only right way to develop software. It is through natural interaction that people make sense of the world, and come to terms with what they want from a system.

The Self Guiding Scrum Team

I like Scrum because the development team works together in an open environment. This important aspect of Scrum is the self guiding teams, where the scrum team is self regulating. I find this as the only right way to develop software, as it opens the development process for discussion.

I like the daily scrum meetings, as it gives each involved person in the team a chance to put forth his opinion. This creates an open forum for reaching a consensus. What I do not like about the meeting, is that it is highly formalized, as you can only ask three different kinds of questions. This puts the involved individuals in constraints, which I do not like. According to me the involved persons make sense of their surroundings in reference to the social environment in which they interact. Therefore, it is not ideal with highly formalized meetings, because it can inhibit the members of my team in reaching an acceptable software system. However, as good as the notion of self guiding teams is, it also contains a major problem which is the amount of users involved.

A Working System as Documentation

Scrum does not focus much on documenting the actual software product, but rather on documenting the development process. This is apparent in the use of a product- and sprint backlog for structuring requirements in the implementation process. The project backlog evolves as the customers needs evolve. Every change performed in the requirements to the system are documented in the project backlog. This clearly shows that scrum is open to changes, which I see as a fundamental prerequisite of a development method.

The combination of product- and sprint backlog creates a detailed dairy of the development process, from which you can analyze the process. This is a quality I like, because I see the process you go through in developing the software as more important than the actual product. The system should facilitate the needs of the users, and they emerge and get refined through the process of human interaction in the social environment.

Another positive aspect of Scrum is, that the requirements for the system are documented into a working system. You cannot misinterpret a working prototype, as you can with written documents. The prototype is a specific implementation of the requirements, whereas written requirements do not specify the actual implementation, it is the reader's job to imagine the actual implementation. Through the prototypes you can facilitate a common understanding of the system. Also the prototypes enables me, as a project manager, to show the concrete progress of the project.

Continuous Validation Through a Working System

As Scrum does not emphasize documentation, the system is validated at the monthly reviews by all interested stakeholders. These frequent reviews are essential for facilitating a common understanding of the system and the requirements to be implemented. Every month the working increments are presented to the stakeholders and are evaluated according to the goals set for the sprint. This enables the users to comment on the system, which is a nice aspect. Furthermore, it is possible for the users to make sense of the system in the environment of intended use, and it can therefore reveal whether the team is working in the right direction or not. The only ones who can test and validate the system, are the intended users.

The drawback in the monthly reviews is the restricted time-frame. As much time as possible should be spend on the crucial stakeholder interaction, as user interaction is important to facilitate. Another less attractive feature in relation to the process of validating the requirements is the power of the product owner. The product owner can possibly inhibit the negotiation of requirements, minimizing the very important social interaction between stakeholders.

7.1.1 My Opinion on Scrum

Overall I like the method as the notion of self guiding teams stands well with my world view of equally important subjective opinions. However, I dislike the division of stakeholders into pigs and chickens. I like the fact that the product backlog is prioritized by a customer representation. On the other hand, I believe user involvement should be more prominent, users should be an integrated part of the scrum team. I like the fact that the system is available to the users for testing in short increments, which facilitates the sense making process.

Table 7.1: Strengths and weaknesses in Scrum from a social relativist perspective.

	Strengths	Weaknesses
Requirements elicitation	▪ Customer decides requirements to be implemented ▪ Requirements elicitation not an up-front activity	▪ Restricted user involvement (too few involved too little) ▪ The role of the product owner plays down the crucial user interaction
Requirements negotiation	▪ The monthly reviews are essentiel for reaching a common understanding of system objectives and requirements ▪ The customer prioritizes which requirements are more risky and should be implemented first ▪ Prototypes are good for communicating requirements	▪ Time-limited reviews ▪ The product owner has the final responsibility of prioritizing requirements ▪ The power of the product owner may inhibit the negotiation process
Requirements documentation	▪ Requirements are embedded in the prototypes/product	
Requirements validation	▪ Users can evaluate and comment on the system at the monthly reviews	▪ Too much power to the product owner, as he decides which user comments to take serious
Requirements process management	▪ Planning is not in focus ▪ Product and sprint backlogs useful for managing and documenting the development process ▪ The more risky requirements are implemented first ▪ Changing and evolving requirements are welcomed during the entire process ▪ The "democratic" form of daily morning meetings ▪ The self-directing scrum team emphasizes equal power and facilitates a common sense making process	▪ The strict form of all the meetings ▪ During a sprint the scrum team is shielded from other stakeholders ▪ User representatives ought to be part of the scrum team

7.2 The Radical Structuralist's Opinion on Scrum

(RS) Review of Scrum

Scrum emphasizes stakeholder involvement throughout the development process. The customer-driven iterative development process assigns the responsibility of defining the requirements to be implemented in each iteration to the customer. This way the customer controls the project, iteration by iteration [19] requiring the development team to be in a constant dialogue with the customer.

Alarmingly High Degree of Stakeholder Involvement During the Entire Process

Engaging in a dialogue with all stakeholders is not completely unnecessary, but Scrum prescribes too much dialogue throughout the process. Ideally, a few customer representatives should be involved in the beginning of the project when elicitating requirements. I would definitely prefer to limit the time spent on dialogue to the beginning of the project. It would be difficult to control every request for change in requirements. I do not see the need for accommodating every change request just because the customer suddenly has a new understanding of the system. Further, there will always be dissatisfied stakeholders expressing alternative requirements. The opportunity of continuously defining new requirements is definitely not an advantage for me as a project manager. It is just the contrary; it will make the development process much more complicated and difficult to organize; and it will be much more vulnerable to possible conflicts.

In general, my work is not to act as a mediator. My working activities do not require me to obtain a common understanding or make sure that everybody involved have understood everything and accepts it. I know that every stakeholder has his own perception of matters, but I do not see the reason to involve every single stakeholder, as the social relativist suggests. Things will only become more complex; maybe even too complex compared to the system.

I see an information system as a tool for making a company's activities more effective; for instance, faster response time, less idle time. With this goal in mind, I cannot see the relevance of focusing on subjective views. The goal is to enhance profitable activities!

> *"...stamp it on your forehead if necessary: the purpose of the requirements process should not be to **cover all eventualities**, or to **limit the damage**, or to **minimize risk**, or even to **satisfy the customer**. The purpose of the requirements process is to add business value.*
>
> ***- Favaro [12 p 17]***

Scrum gives you the opportunity to add features throughout the process. This might be a good way of capturing all the requirements and especially those which are hard to find in the beginning, but it would not help me structuring and keeping track of my work as a project manager.

One Person in Charge of Defining Requirements

Involving stakeholders and asking for their evaluation of the system every month seems as a way of causing unnecessary trouble. However, the product owner is in charge of deciding the priority of the requirements and the review is time limited. This naturally sets a stop to unproductive quarrels between stakeholders. Actually, I do not see great value in the monthly reviews. The stakeholders can only validate the user interface; they cannot validate the functionality of the system.

Difficult to Structure Work

As mentioned, Scrum is an iterative method. The stakeholders are given much attention, as they continuously decide what assignments should have highest priority. The stakeholders choose the features to be implemented during the next iteration according to what has the highest business value [19]. The prioritized list of requirements is sorted according to risk. The technically risky requirements associated with the highest business value is prioritized highest. The technical risk is evaluated by the system developer, while the business value is evaluated by the customer; therefore, the ranking of assignments can be quite complicated.

Making a prioritized list can be beneficial in situations where time is short and you know that not the whole system can be developed. However, as a project manager, I do not find this structuring of work effective. I believe that whatever requirements exist they should be implemented within the time span available. I prefer to develop the whole system before presenting it; therefore, prioritizing requirements would not be necessary. Assuming, that I had to prioritize, then I would focus on which requirements are more profitable financially or in relation to an up-skilling of the workers' qualifications.

Fuzzy Division of Responsibility Within the Scrum Team

Scrum prescribes the scrum team to be self-regulating. The scrum master does not only work as a part of the management; he is also a developer. The fact, that the person in charge of development is highly involved in the actual development, confuses me. If I am working side by side with a system developer I might have problems sustaining my manager status. I believe the working positions are socially as well as organizationally defined; and I prefer a clear distinction of roles, in stead of becoming uncomfortable due to my involvement in the system developers' daily work.

I like the morning meetings as I can obtain an overview of what is happening in the development team. Also, the fact that everybody in the team tells about their assignments can assure me that they have understood given assignments.

Inadequate Degree of Documentation and Specification of Requirements

Scrum uses product and sprint backlogs as diaries for work tasks. Every day, remaining work tasks and their estimates are updated by the responsible members or by a daily tracker who consults each scrum member [19]. Besides the diaries, there are no documentation. Diaries might be good to keep track of what people are working on, but they cannot be used as the only documentation. Documentation is the evidence of what has been agreed on. I find it frustrating that there is no detailed documentation of requirements or the system. There is no way for me to show the customer that my team has produced what was agreed on in the very beginning of the project.

I think it is a bit too risky to have product- and sprint backlogs for documenting requirements; they only work as an internal way of keeping track of the work of the scrum team. The customer has no insurance whatsoever of what is actually implemented is what has been agreed upon. If misinterpretations and disagreements occur, how exactly should such a situation be handled in Scrum? There is no documentation to resolve the disagreements. In other words, my team would have to adjust and correct the system according to the new requirements?

7.2.1 My Opinion on Scrum

I dislike the continuously high degree of stakeholder involvement. I would prefer not to adjust or redo my planning as a project manager every time changes are made to the prioritized list of requirements. These interruptions

Table 7.2: Strengths and weaknesses in Scrum from a radical structuralist perspective.

	Strengths	Weaknesses
Requirements elicitation	▪ A single person (the product owner) in charge of requirements elicitation	▪ Vague formalization of user needs ▪ Every requirement ought to be defined early; a prioritized list is not effective ▪ The sprint reviews where several stakeholders attend can cause unneccessary trouble
Requirements negotiation	▪ One person (the product owner) in charge of negotiation	▪ The more predictable, and not the generally risky requirements should be prioritized
Requirements documentation		▪ No written documentation as proof of agreed requirements
Requirements validation		▪ Only the user interface is evaluated at the monthly reviews - no guarantee that the system contains the required functionality and produces the correct data
Requirements process management	▪ Backlogs are good for keeping track of what people are working on ▪ Morning meetings good for keeping track of the scrum team's work	▪ Too many stakeholders involved too often during the development process ▪ Dialogue (elicitation and negotiation) of requirements should be restricted to the beginning of the project ▪ Continuous stakeholder involvement makes the process vulnerable to conflicts ▪ Difficult to control every change required ▪ The process in Scrum makes it impossible to plan an entire project ▪ Too loose organization of work ▪ No clear distinction of responsibilities within the scrum team ▪ At the reviews all the work of the scrum team can be rejected ▪ Changing requirements should be welcomed during the entire project.

does not make my work more effective.

7.3 The Radical Humanist's Opinion on Scrum

(RH) Review of Scrum

Scrum assumes that developing software is all about creating something new. You can never repeat a particular development process again and again. This is due to the fact that a particular project has its own characteristics, priorities and demands, that necessarily have to be taken into consideration. Scrum builds upon the assumption, that it is impossible to define the entire development process at the beginning of the project [25].

My view on the framework, that Scrum sets for developing software, is that it has one main objective, which is to produce code fast. In order to do this, Scrum prescribes: Specific frameworks of time for different activities, delegates responsibilities according to roles, and focuses attention solely on functional requirements, that can optimize business value. I agree, that you can never repeat a development process, since each project is unpredictable and has its own issues and specific considerations to be made. However, I strongly disagree, that delegating responsibilities and strictly limiting the possibilities for interaction among the different stakeholders is the way forward. In addition to this, it puzzles me greatly, why the use of the word requirements are synonymous with functional requirements. In my opinion, the functional requirements are only one aspect of the needs that the development of a system must address. I believe, that non-functional requirements must be valued at least as high as functional requirements, when a system is to be successful.

Casual Requirements Elicitation

The way, Scrum proposes a strategy for eliciting requirements and working with these requirements and new ones, seems superficial and naive to me [15 p 175]. On the other hand, Scrum, quite right, has the philosophy that every requirement cannot be defined at the beginning of a project [12 p 16]. Initialising a project in a four hour session is just not possible. Most likely, all the different stakeholders have not met before, and setting a frame of four hours, and no more, for all to agree on and decide the main objective of the system simply cannot be done. First of all, four hours is not enough to create a free speech situation where all stakeholders can appreciate a rational argument. It requires more time and social work in order to eliminate obstacles to the human communication [15 p 175]. Requirements for a system are only legitimate if they emerge from an open and free discussion, where the rational arguments are acknowledged, and this cannot be obtained in four hours. Eliciting requirements is an activity of negotiating requirements according to their rational value. The time pressure will probably be an advantage for the most powerful stakeholders, meaning that their point of view is favoured on behalf of others'. There is not enough room for the crucially important social interaction from which a shared understanding of the future system emerges.

Blind towards Non-functional Requirements

Another objection, I have against Scrum's approach for eliciting requirements, is that attention is entirely on stating functional requirements. I would put much more focus on the social process of eliciting requirements as this is of great importance to the success of the system. If requirements do not stem from a free and open discussion, where all stakeholders appreciate the weight of rational arguments, there will never be a collective opinion of the purpose of the intended system. I cannot stress the importance of the social formation of meaning enough.

Stakeholders Perceived as Unequally Important

I like the fact that requirements are not tracked and controlled, they emerge at each sprint review. I also appreciate the fact, that all stakeholders should attend these meetings. It is, however, unfortunate, that the method distinguishes between stakeholders' importance, grouping them into either pigs or chickens. All the stakeholders of the system development process are in my opinion equally important and cannot be divided into the more important ones and the less important ones. It seems that the developers have a tendency to be the important ones, while the customer organization is given little attention. I find it problematic to put so much effort, as is done, in viewing the customer groups intervention during a sprint. As I see it, it is exactly the interaction among developers and customers that yields a better product. Restricting this interaction only results in a poorer understanding of each other.

All Power to the Product Owner?

Scrum keeps the backlogs open for all stakeholders which sends a positive signal. The actual effect, however, is of less significance, I believe. It is only, the product owner, who can make changes to the product- and sprint backlog. Other stakeholders can only hope that this stakeholder will listen to them at the sprint planning meetings, and not only act according to own interests. At the sprint review meeting, however, the product owner's influence is decreased, as all stakeholders can express their opinion and maybe discard some of the system. The sprint review is not just a review of the work the scrum team has done; indeed, it is also an evaluation of the product owner's decisions.

Continuous Validation - Very Nice Feature

The process of the development method which encompasses a continuous stakeholder validation of requirements documented in the actual system is important. It is better to evaluate the system than documents describing the system, as these can be misinterpreted; and letting stakeholders validate the system every month keeps them informed. It is much better than just showing the system at the end of the entire development process. Scrum is actually quite open when it comes to criticizing what is being developed. If requirements change and what has been developed during a sprint is no longer necessary, it can simply be discarded at the sprint review meeting, where all interested stakeholders can attend and express their view.

Organizing Work Internally in the Scrum Team

Internal to the development organization, I am very positive of the organization of work and people. The role of the scrum master is to work as a facilitator, and this is the only way to work as a project manager. The project manager can never be a leader that tells everyone what to do. Self-organizing teams with a facilitator helping out, when necessary, is the way forward. This enables everyone to have influence and speak out their opinion.

7.3.1 My Opinion on Scrum

Viewing the method as a whole, I find the excessive focus on coding problematic. The amount of time, where the developers are concerned with coding greatly exceeds the amount of time used for addressing social issues and facilitating the very important free dialogue between stakeholders. I am aware that there is a need for structure in order to develop a system, but in this case, structure is enforced at the expense of crucial social interaction. The structure also implies a strict division of responsibilities, which is problematic in the case of the product owner. One single person has the responsibility for making decisions that are very much affecting other people. This requires the person being product owner to act as a facilitator, just like the scrum master, otherwise it will not be possible to develop a system, that results in stakeholders having a better understanding of each other, and a system that institutes a state of more freedom and well-being for all stakeholders.

Table 7.3: Strengths and weaknesses in Scrum from a radical humanist perspective.

	Strengths	Weaknesses
Requirements elicitation		▪ The method strictly limits the time for stakeholder interaction ▪ Only focus on functional requirements - non-functional requirements are paid no attention ▪ Too superficial and naive approach for requirements elicitation ▪ No focus on the social side of requirements elicitation ▪ Product owner has too much power
Requirements negotiation	▪ Prototypes are better than written documents for communicating requirements	▪ The method strictly limits the time available for negotiation. Four hours much too little for reaching consensus ▪ Too superficial and naive approach for negotiation ▪ The more powerful stakeholders may keep down the weaker ones due to the conditions for negotiation ▪ No focus on the social side of negotiation ▪ Product owner has too much power
Requirements documentation	▪ Requirements are not tracked and controlled	
Requirements validation	▪ Monthly stakeholder evaluation of the system ▪ Every stakeholder can attend and speak at the reviews	
Requirements process management	▪ Every requirement is not to be defined from the beginning; they are allowed to emerge during the process ▪ Monthly reviews keep stakeholders informed ▪ New or changed requirements are welcomed ▪ Self-organizing teams with a scrum master as facilitater ▪ Every stakeholder can look in the backlogs	▪ The main objective is to produce code fast (much more time spent on coding than on social work) ▪ Too much time spent on implementation according to time spent on elicitation and negotiation ▪ Distinguishing between pigs and chickes ▪ Structure is enforced at the expense of social interaction ▪ Too strict time frames

A nice aspect of Scrum is the product and sprint backlogs, as they are open for all to see. This sends the right signal; everybody involved should be able to gain insight into the process and the product. The table below states the strengths and weaknesses of Scrum.

7.4 The Social Relativist's Opinion on OOA&D

OOA&D is a method that essentially consists of a tool set for the developers to use for thoroughly analyzing and modelling an information system. Looking at the method in an overall view, it emphasizes creating a set of detailed analysis and design documents which can be used as a basis for implementing the entire system.

The Stages in OOA&D

OOA&D focuses mainly on the analysis and design phases of information system development; afterwards it seems to be a matter of implementing exactly what is specified in the design documents. I regard elicitation and negotiation of requirements to be a part of the analysis activity. The method does not prescribe how many and how much to involve users during the process. I would prefer, if an adequate amount of user involvement was specified by the method, in order to be sure that users and developers communicate enough to reach a common understanding of the problem and the system to be developed. However, it is not possible to give guidelines on the amount of interaction, as the adequate amount depends in the situation in question. During elicitation and negotiation it is nice that it is possible to involve users as much as necessary, as OOA&D does not prescribe time restrictions like Scrum does.

The drawback of OOA&D is that the focus on making documentation naturally limits elicitation and negotiation of requirements to an early stage of development. Once the design documents have been made, implementation proceeds according to these documents.

> *"...up-front requirements was the single largest contributing factor for failure, being cited in 82% of the projects as the number one problem with an overall weighted failure influence of 25%."*
>
> ***- Larman [19 p 74]***

This way, the users are excluded from the process after analysis and design have been performed. Users are not involved again until the system is implemented. There is no user validation of requirements during the implementation stage. In the worst case, the users are presented to the entire system, once implementation is finished; and that is not a process that facilitates the users' sense making process.

Analysis and Modelling with the Aim to Produce Documentation

It is my perception, that the basic principle of the OOA&D method is to conduct a meticulous analysis, which results in detailed analysis document specifying the requirements for the system. The analysis document is to form the basis for the following design phase, which results in a document specifying a detailed design for the system. Analysis and design are described as two distinct activities; if changes occur during design, they are documented in a separate correction document. The original analysis document seems to be perceived as holy as the Bible, as it must not be changed.

Believing that the problem domain can be analyzed and expressed in a document that forms the basis for an entire system design, cannot be further away from my world view. It is simply foolish to think that the problem domain can be described in just one analysis document in an early stage of development process and then form the basis for all further design and modelling. First of all, the social world is not perceived similarly by two persons, so the documents will express a certain interpretation of the problem domain. Second, the social world is in an ever changing process [19]; therefore the analysis and design documents will quickly get out of date. Further, perceiving analysis and modelling as two distinct activities is simply wrong in my opinion. Analysis and modelling are two sides of the same coin; gaining more insight into the problem domain creates a clearer view of the system to be designed; and knowledge of the design opportunities affects the analysis to be conducted. In my view, analysis and design should not be up-front activities; the process of understanding the problem and the solution should continue during the entire development process.

Furthermore, I worry that the focus on analysis and design documents will result in a development process that is split into phases. Although the authors of OOA&D state, that the method can be followed both sequentially or incrementally, I still see the way each activity builds upon the previous as pulling the process in the direction of a phase-driven development. The requirements for the system should ideally emerge through interaction between the users and developers throughout the process, in order to facilitate the very important common sense making process.

The method is not completely useless, however. It is possible to pick and choose which tools to use, as well as how to use them. Therefore, it is up to me as project manager to decide how much users should be involved during the process, as well as which analysis and modelling technique to use. The fact, that I can adapt the method according to my own needs and understanding of the problem, is something I really appreciate.

Analysis and Modelling Techniques

I think that rich pictures and user stories are well-suited techniques for reaching a common understanding of the problem and the system to be developed. However, in order to utilize the potential of rich pictures many users have to be involved. I would regard it as a major problem, if the developers model the world themselves without consulting the users. I also like the concept of statechart diagrams, as they can help obtain a common understanding of the system by giving all the users the opportunity to draw their own statechart diagrams.

The problem with the modelling techniques in OOA&D is that the object-oriented philosophy supports static generalizations of the social world, and they generalize the users' perception of the problem domain. The method lists many tools for modelling the system. But these models do not include the user perspective which I find crucial. I do not believe that a system can be modelled without the users. There are a large number of subjective opinions on how the system should be, and this cannot be described using models. Communication between the users and developers, is the only way to reach consensus.

Extensive Documentation Freezes Understanding of Problem Domain

The use of OOA&D can be expected to result in a large amount of written documentation containing detailed specifications of the system to be developed. The great amount of work used for producing this documentation is wasted in my opinion. Soon after elaboration, the documentation will become out of date, as the world cannot be perceived as static. Further, extensive documentation complicates the incorporation of changing requirements.

I see OOA&D as a technical oriented method, which is a bit too narrow minded. I believe, that one has to focus on the foremost important issue, which is involvement of many users, not the reasoning about specific technical solutions [15 p 175]. The reason for this, is that all developers have technical expertise, and therefore technical guidelines should be kept in the background. I do not understand the urge to formalize everything in documents, as in my

Table 7.4: Strengths and weaknesses in OOA&D from a social relativist perspective.

	Strengths	Weaknesses
Requirements elicitation	▪ During the elicitation phase much user involvement is possible ▪ Rich pictures and user stories are well-suited for reaching a common understanding of the system	▪ The method does not prescribe enough communication between users and developers ▪ Models generalize the user's perception of the problem domain ▪ Problem and application domain should not be thought of as separate; they are interrelated ▪ Problematic that elicitation is limited to an early stage of the project
Requirements negotiation	▪ Rich pictures and user stories are good techniques for reaching a common understanding of the system ▪ Negotiation of requirements time-restrited	▪ The method essentially consists of a toolset for the developers to use for thoroughly analyzing and modelling the system ▪ Problematic that negotiation is predominantly limited to an early stage of the project
Requirements documentation	▪ Possible to spend as much time as necessary on the elicitation and negotiation phase for reaching a common understanding of the system	▪ Extensive documentation will soon after elaboration become out of date ▪ Extensive documentation does not welcome changing requirements ▪ Documentation creates a static image of a dynamic social world
Requirements validation		▪ No user validation of requirements
Requirements process management		▪ The method encourages phase-driven development ▪ Users are excluded from the process after analysis and design ▪ Analysis and design cannot be separated to two distinct activities ▪ The object-oriented modelling philosophy supports static generalizations of a social world ▪ OOA&D narrow-minded in the sense that focus is more on implementation issues than on the social sense making process

view the most important issue, the human sense making process, cannot be formalized.

7.4.1 My Opinion on OOA&D

I have been quite critical when describing my view on the method, I do, however, see positive aspects in OOA&D. My reservations concern the objective with extensive modelling documentation. To me, OOA&D seems a bit narrow-minded in the sense, that focus is mainly on reaching a good design; less attention is given to reaching social consensus on the problem and system to be implemented. The method does not directly facilitate the social sense making process. It may however be a question of using the analysis techniques for facilitating common sense making between all stakeholders, and organize the development process to support an evolving understanding of the system.

7.5 The Radical Structuralist's opinion on OOA&D

I see OOA&D as a method that, by means of different tools and techniques, aims to make detailed documentation that can be used as a recipe for implementation. Several analysis and modelling techniques are presented as recommendations for how to handle the requirements during system development. The OOA&D method emphasizes two phases; namely, analysis and design, which are further divided into analysis and design of single system elements.

Some Very Useful Modelling Techniques

What caught my attention in OOA&D was the use of tools for modelling. For instance, I find tools for creating statechart diagrams and class diagrams useful for organizing the work.

> *"Analysis and design models provide useful tools for thinking abstractly about complex problems."*
>
> ***- Malan [20 p 38]***

Obviously, each technique has a different purpose, and I do not find all the techniques useful. Some of the tools, given in OOA&D, may make it easier for the users to understand how we, as the development team, have understood the requirements. Drawing rich pictures and writing use cases can be useful techniques when interacting with the users. Personally, I find rich pictures and use cases a bit too risky to use, as they have a tendency to emphasize different interpretations of the same situation. I would prefer to avoid the risk of ambiguity all together. The possibility of misinterpretation is not desirable, as I am already navigating in a delicate and conflicting environment. If conflicts emerge due to ambiguities in the documentation, it can ruin my plans for the development process. Moreover, the degree of user involvement during the development process should, according to me, be as little as possible - preferably placed in the beginning of the process.

I am of the conviction that structuring your work, and making clear and generalized descriptions of the problem area, is the way to avoid misinterpretations and ambiguity. OOA&D emphasizes the use of UML, as UML provides a common grounding and an objective way of illustrating the problem area. Therefore, I regard UML as an excellent technique for avoiding conflicts and documenting the work.

Documentation As Proof of Agreements

As mentioned earlier I prefer structure in my work; and I believe that the use of documentation and specific analysis and modelling techniques only stresses the structure of work. Further, documentation makes my job much easier as everything is stated in written documents. In situations where conflicts might occur, these papers are my proof that what I have developed, is what was agreed upon. The detailed specifications minimize the occurrences of misunderstandings. When an agreement has been made and outlined in a document I consider the agreement valid.

I understand documentation in OOA&D as a closure to each part of the development process; once the documentation has been made, corrections are not allowed in these documents. If changes occur, corrections have to be made in a correction document. This way, I can keep track of what had been agreed in the very beginning, as well as what changes have occurred later. I am fond of organizing the documents this way, because then the customer and I can keep track of how requirements have been implemented. I see documentation as evidence in situations where disagreements between me an the customer occur. If the customer do not believe what has been implemented is what we agreed upon, I can refer to the documentation as a proof.

No Guidelines for Organizing Work

The method does not predefine to what degree the different tools should be used. It is up to the project manager to define the overall time span for each of the phases. I find this part of the method very convenient as well as a little disturbing. I find it convenient because I have the choice of deciding how much and to what extent I want to involve the users. I do not believe that the users need to participate in the whole development process as I prefer to finalize the agreements in the beginning of the development process. The reason why I find this freedom a little disturbing, is that I do not have any guidelines on how to structure the overall process, OOA&D hardly mentions this topic, and I by myself have to define the time span according to the project, which I might find a little difficult. I would have preferred more organized guidance in this area as I feel the need for an overall structure of all the models and documentation constructed through this method.

In continuation of the need for an overall structure of the development process, I think that a guide on how to distribute the work in the development team could be beneficial. Somehow I like the fact, that I have the ability to control and have the overview of what everybody is doing in the team but miss a clear definition of who does what and when.

Lack of Validation Techniques

When I am developing a system I prefer to keep my focus on the outcome - in the form of economical benefits as well as social benefits. Sometimes it can be difficult to make precise measurements of how beneficial the system will be. But with the use of, for instance, statechart diagrams and descriptions for functions I can keep track of parts which I am efficiently enhancing through the new system. Moreover, I can use the statechart diagrams and the descriptions of functions for validating the system. This way I do not need the users to get involved. The OOA&D method does not give any concrete suggestions on how to validate the system when implemented. Of course the detailed documentation can be used to validate against, but no specific validation techniques are specified.

7.5.1 My Opinion on OOA&D

I like the use of tools and models to structure my work as a project manager. This way I can keep track of what and how it is being developed in an objective way. I must admit that some of the models like rich pictures is not something I would use, as it does not give me an objective view of the surroundings. Instead I would prefer the use of statechart diagrams and descriptions of functions.

As a project manager I have the responsibility of assuring that the goal of profit maximization is reached. By using statechart diagrams I can actually measure the cause and effect of each function in the system.

Table 7.5: Strengths and weaknesses in OOA&D from a radical structuralist perspective.

	Strengths	Weaknesses
Requirements elicitation	▪ Useful tool set for formalizing and generalizing requirements ▪ The degree of user involvement can be decided by the project manager	▪ Rich pictures and use cases too risky to use, as they can be misinterpreted and cause conflicts
Requirements negotiation	▪ UML is an excellent modelling technique for objectively describing the problem domain	
Requirements documentation	▪ Documentation very good proof of agreed requirements in case of disputes ▪ Analysis and modelling documentation very detailed ▪ Extensive documentation eases my work	
Requirements validation	▪ Possible to validate against the detailed documentation	▪ Lack of specific techniques for validation
Requirements process management	▪ System development proceeds on the development team's terms ▪ The method facilitates stage-driven development ▪ Specific techniques for handling requirements during the development process ▪ Documentation useful for organizing work ▪ User involvement can be kept at a minimum and be restricted to the beginning of the process ▪ Each phase in the development process is closed by documents that are not allowed to be changed ▪ Modelling documentation can be used for structuring work	▪ Lack of guidelines for organizing work in the development team ▪ Lack of techniques or tools for keeping track of the developers' work

So my overall verdict is that when using OOA&D, development proceeds on the development team's terms; and stage-driven development is possible. Some of the techniques and models within OOA&D are very useful, but somehow I miss an overall process control tool or method, as there are no specific recommendations on how to organize and structure the development team.

7.6 The Radical Humanist's opinion on OOA&D

I see OOA&D as a method, that puts great effort in analyzing the problem domain and modelling a possible solution at the beginning of a project. At the initial phase of the project, there is great focus on developers and users gaining a mutual understanding of the problem domain. Different techniques are used for supporting the interaction between developers and users in achieving a shared understanding. For instance, stakeholders can express their opinion on the subject by means of use cases and rich pictures, that can be used as a basis for negotiation among the stakeholders.

OOA&D is essentially concerned with gaining a better understanding of the problem domain and modelling a complete solution early in the process. This is unfortunate, as is the fact that OOA&D ignores changing requirements and does not care about the users after the modelling phase. Excluding the user organization from the process after the analysis and modelling phase violate my perception of system development. The process of developing the system should facilitate an institutionalization of an ideal speech situation, therefore excluding the users from the process, after the initial analysis and design, is very unfortunate [19 p 74]. The system as well as the user organization would only benefit from letting changed and evolved requirements be seriously considered. Using OOA&D means, that validating the system does not even concern the users, as it is done according to design documents made in the early phase of the development process. In order for the system to be validated properly, I strongly advice, that the future users should test it several times during development. This way, the users can follow the development of the system and comment on it, and not just at some point in time be obliged to use a system, they have had little influence on.

"Stakeholder representatives should be Crack performers: collaborative, representative, authorized, committed, and knowledgeable. Shortfalls in any of these capabilities can lead to frustration, delay, and wasted project effort, not to mention an unacceptable product."

- Boehm and Lynch [3 p 62]

The Process in OOA&D

OOA&D does not explicitly describe a development process, it suggests using an appropriate amount of analyzing and modelling techniques. It does, however, prescribe that the system is built according to design documents, either in one go or stepwise. This means that during the process, analysis, modelling, and documentation always preceed implementation.

I think, that the amount of documentation in OOA&D can take attention away from the very import process of developing a mutual understanding of the objectives of the system to be developed. When something has been documented, it seems that OOA&D ignores later changes. I find this odd, as it is a fact that requirements keep changing, even after implementation has begun, and it is not necessarily a bad thing that requirements change, as new insight can put attention to important issues.

Modelling Techniques of Varying Quality

It is only some of OOA&D's analysis and modelling techniques that I can vouch for. I see use cases and rich pictures as excellent means for facilitating a discussion among stakeholders that can aid a mutual understanding of system objectives and form the basis for a free speech situation. Also, I am very fond of the paper mock ups or prototypes used for reasoning about what system should be built. Using graphic representations of the intended system is a great way of letting users comment on the system.

Using class diagrams is not a modelling activity, that makes much sense to me. I do not like the idea of observing the "real world" and dividing it into classes and giving them attributes. For me, system development is all about facilitating a change for the better, and for this reason, it makes no sense to create a model of status quo. I understand, that a class diagram can be percieved as a representation of some view on the world, but it definitely cannot be percieved as modelling an underlying truth. A specific class diagram implies a certain abstract view of the world, and in this context, I see its relevance for reasoning about implementation issues. Internal to the team of system developers, it may also be beneficial for communicating the structure of the system. It is not, however, useful for facilitating negotiation among stakeholders.

Fragmented World View

The concepts of analysis and modelling implies that there is an underlying structure and truth to be observed through thorough analysis. This, I do not agree with. To me, truth is a matter of facilitating the free speech situation and letting the rational arguments speek for themselves. The only legitimate requirements, are the ones that emerge from a free speech situation. Analysis will always be coloured by the analyzing person's world wiew and own interests.

OOA&D separates problem domain from system domain. In my world view this is simply not possible. What these two concepts refer to is two sides of the same coin. They define and constitute each other. Seperating them, would be the same as separating milk from chocolate in chocolate milk.

Validating the System According to Documents

OOA&D does not describe methods or techniques for validating the system or increments of the system. The system is validated according to the design documents. I firmly believe that to validate a system, it is absolutely necessary letting the future users evaluate the system - preferably several times during the process, in order to keep them updated and giving them the opportunity to express their meaning.

If OOA&D was to be used incrementally, and the users validated each increment, I would be much more positive towards this method. An approach like that would facilitate gaining a thorough mutual understanding between developers and users, and the stepwise user-validation would keep the users informed and involved, giving them influence throughout the process. Further, the stepwise analysis, modelling, implementation, and validation also makes it possible to build a strong architecture; so, although heavy on documentation, I believe, that the incremental approach makes a sound development possible.

Naive Perception of Stakeholders

There is one aspects in relation to stakeholders, that puzzles me. It seems to me, that OOA&D employs a undifferentiated view of stakeholders, as they only collaborate with users. There is no mentioning of other stakeholders. It makes me wonder, how this method handles the diverse group of stakeholders, as it is wrong to assume that stakeholders are synonymous to a homogeneous group of users.

7.6.1 My Opinion on OOA&D

Speaking to OOA&D's advantage is the fact, that some of the analysis and modelling techniques can be used to facilitate mutual understanding and a free speech situation, where stakeholders on the basis of rational arguments can negotiate and agree on legitimate requirements. OOA&D does not prescribe a specific amount of time to be used for elicitation and negotiation, so as much time as necessary can be used. However, I do think, that the intense focus on analysis and modelling will often lead to a development process that is dominated by documents. It is fundamentally wrong, if the documentation freezes the understanding of requirements. I believe that the documents tend to be used as a form of contract between customer and developer organization, and this is why they are also used for validation. It is a fundamental drawback to the method that it does not take changing and evolving requirements into consideration.

> *"The lack of user input contributes to incomplete requirements and specifications, because only the system users collectively have the necessary understanding of the needs to be fulfilled"*
>
> ***- Herlea [13 p 30]***

Valuable information might get lost when ignoring information gained during the development process. However, if OOA&D was applied incrementally and the users evaluated each increment, I would be much more prone to using this method. That way, the method naturally allows for more user involvement.

I am quite ambivalent towards this method, as it has both good and problematic features. There is the possibility of using OOA&D in a bureaucratic way, where users at some point are ignored and maybe even repressed; also there is the possibility of using OOA&D in a more democratic way which involves and gives influence to users. Should I be in favour of this method, it would necessarily have to be applied in a way that values the institutionalization of a free dialogue throughout the process. Then perhaps it would be a sound context both for considering technical aspects, facilitating mutual understanding, and emancipation.

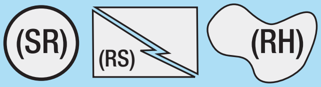
Table 7.6: Strengths and weaknesses in OOA&D from a radical humanist perspective.

	Strengths	Weaknesses
Requirements elicitation	▪ Stakeholders can express their view of the problem domain by use of rich pictures and use cases	▪ Separation of problem and application domain which are really interrelated
Requirements negotiation	▪ Use cases and rich pictures are excellent means for facilitating a discussion among stakeholders that can aid mutual understanding ▪ Paper mock-ups or implemented prototypes great for reasoning about requirements ▪ The only legitimate requirements emerge from a free speech situation	
Requirements documentation	▪ Class diagrams only useful for reasoning about implementation issues	▪ The amount of documentation may take the attention away from the more important process of developing mutual understanding of system objectives ▪ Detailed specifications can freeze the understanding of requirements
Requirements validation		▪ Validation against documentation ▪ Users do not validate the system
Requirements process management	▪ OOA&D used incrementally	▪ User should evaluate and comment on the system during the entire process ▪ OOA&D employs an undifferentiated view of stakeholders ▪ Risk of employing a bureaucratic process ▪ Analysis, modelling and documentation will always preceed implementation ▪ Fundamental drawback that the method do not welcome changing requirements

8. Three Alternative Understandings of the Project

Both Scrum and OOA&D have now been thoroughly reviewed, which is why I am now interested in knowing the challenges my three fellows see in the project proposal. I have therefore asked them to make an assessment of the project pointing out the main risks and concerns, and advice according to them.

Three alternative understandings of the project



8.1 The Social Relativist's Assessment

When I study the assignment the functionalist has been given, I see critical elements in the EIS project, which I will outline and give my advice on.

8.1.1 Risks

The first thing that has caught my attention, is the decision of a centralized solution made by the Ministry of Education. I find this decision problematic, as the system should be developed according to a rising need from the users. The second thing that has caught my attention, is that the system is suppose to be a common system for ten institutions placed at different locations in Denmark. There is a risk that the users of the system would not obtain a common understanding of the system, as the social environment differs from site to site. It would therefore be difficult to model a common understanding of the system, when the user groups are split geographically.

Another aspect, which concerns me, is the fact that the institutions are different and therefore have different needs for the system. In my view the system should be developed from within the users perspective. This does not harmonize with the functionalist's understanding of the project, where one centralized system is provided for all the institutions. I would prefer to develop an individual solution for each institution. However I do realize this is not possible within the budget and given time frame.

Furthermore, I do not like the fact that each institution has decided to assign just one super user as a representative in the project. One user representative is just not enough. All the users should be involved and it is not enough that the super user has the responsibility of involving other users from their institution. The decisions should be made by all the users together.

8.1.2 My Advice

I have to say, that both methods have their problems when it comes to the diverse user groups, which are geographically dispersed. I believe it is possible to make a design that can help this, and the solution would be to develop the system in modules. This way a specialized module is developed for each institution, where the user group is smaller. Further, the development of the following modules can benefit from the knowledge gathered from the first module and so on. The system should contain a central server, which ties the modules together and where all the information is kept.

My advice to the functionalist is to use Scrum as a development method, but with a few modifications. I do not like OOA&D because the object-oriented idea seems very far from my world view. For me the method is not applicable in this case.

A common understanding of the project should be reached by the users. But this could be very hard, as the system is very large and therefore a lot of users would be involved. Frankly, I do not know, if that is possible in such a large project. Furthermore, there are a lot of stakeholders, who would try to influence the development and it is therefore important to shield the system from that influence.

I like Scrum because everybody in the development team are considered equal and it is the users, who define the system requirements. But Scrum does not have enough user involvement, therefore the product owner should be replaced by a group of all the users. If this is not possible an agreement should be made to assure the way the user representatives elicitate requirements is standardized. It should be possible to see all the different user opinions in the requirements. Therefore the functionalist should make an agreement on how the users reach a common understanding of the system.

The user involvement in Scrum is not frequent enough, and I would advice the functionalist to include the users in the scrum team. Thereby the users can participate in the development and reach a common understanding of the system, together with the developers. The concept of an onsite customer has proved possible in the XP method, and therefore I believe it would be applicable in Scrum as well. I also believe that the functionalist can learn from the systems already in use. These systems are a product of a common understanding at the institutions and many lessons can be learned from them. The fact that the scrum master shields the scrum team from external impressions is ideal, if the users are a part of the system. This way it is the users that dictate the design of the system, which is perfect. But I have to stress that the users should be part of the team before this is valid. Otherwise the scrum team would not develop the system from the users standpoint.
So my advice is to use Scrum with a higher degree of user involvement in the development of a modular designed system.

8.2 The Radical Structuralist's Assessment

When I study the EIS case I see some critical elements, which I will outline here, and give my advice on.

8.2.1 Risks

My main concerns regarding this particular case are the political motive behind this project and the choice of integrating all ten institutions.
The political motive behind the EIS case is, as I see it, a question of increasing control of the Danish Education System. And the control will be used as a way to assure that all ten institutions adapt the rules and regulations given by the Ministry. The motive, which is obvious, might turn some of the institutions against the development and therefore complicate the whole process of development.

I see conflicts arising when trying to clarify the requirements specification. The fact that the Ministry has decided that all ten institutions should be involved in the development only complicates the process. I see all ten institutions having their own political motive with the system and the Ministry having their own. Eleven different motives cannot be combined into one project, it can only raise a huge conflict.

The structure for the project, as presented by the functionalist, is infeasible. Involving so many different parties cannot be seen as an advantage for the project. It only makes the development process longer and more conflict ridden. The system could have been better structured if not so many parties had been involved. Just imagine the conflicts arising in the group of user representatives!

8.2.2 My Advice

I have presented the risks I see in this project, and as I see it, it is conflict ridden and can be difficult to structure with the terms given by the Ministry. I would therefore recommend the functionalist to choose a method, which can give him an overview, tools to structure, tools to objectively describe requirements given by the ten institutions, and would strongly recommend him not to allow too much space for discussions in the groups, as I could imagine them ticking like a bomb, waiting to explode.

I would recommend the functionalist to assure the needs of the Ministry are fulfilled in the first place and the system is developed accordingly. After all it is the Ministry, who has proposed the idea and need for this system and therefore are best aware of what it should contain. I would suggest that an overview of the functions needed is made as general as possible. By generalizing them you will avoid too many corrections, and adjustments from the user groups.

I would strongly recommend the functionalist to capture the requirements in an early stage of the development and thereafter concentrate on implementing them. Do not involve all the parties throughout the project. You would not be able to organize or structure the assignment properly if corrections, additions etc. keep booming in.

Moreover, I see the Ministry as the real users of the system and the institutions as the ones who have "to enter the data" for the Ministry. So, what has to be assured is that the user groups are given proper instructions in how the data should be entered. Therefore they do not need to be involved throughout the process.

I would recommend you to develop the system at once and not in small prototypes. Making prototypes will only give the involved parties an unnecessary chance to add changes. Of course there might be some small adjustments to the system but not something, which cannot be done after final implementation. Making one implementation of the system and adjusting the user interface according to each of the institutions would be my recommendation.

Having the choice of using either Scrum or OOA&D, I would advice the functionalist to use the OOA&D method in the EIS case. The OOA&D method will give the opportunity to structure the product with tools. Tools to make an overview of the functions needed, tools for clarifying and generalizing the requirements presented.

8.3 The Radical Humanist's Assessment

Studying the project, I see critical elements. Here I will outline these elements, and give my advice on them.

8.3.1 Risks

My main concern with the functionalist's assignment is the scope of the project - it is a very large and complex system involving many parties. The main risk is the diversity of the educational institutions; first of all, each provide educations that are not all structured equally, making it difficult to unite the administrative work in one system, let alone make analyses and statistics across the institutions. The question is, if there are enough common needs, the ten institutions and the Ministry can agree on. If not, I do not see the need for cooperating on a large joint system. The functionalist did not mention any initial analysis of which needs are shared by all ten institutions; that would be the first step, in my opinion.

The most important prerequisite for the success of the project will be all ten organizations' motivation for participating in this project. I wonder if the Ministry did not carry out a survey of all the institutions' view on a joint and central system. If they have not, that would definitely be the first step to take. The project is ridden from the beginning, if it starts out from a place where it will not be possible to provide the basis for a free dialogue where consensus can be reached. In all information system development projects, motivation from all parties and a mutual understanding of the objectives of the system are essential. In a project of this scope and complexity, motivation and mutual understanding are even more important, as the project is even more vulnerable.

If it turns out, that there is a mutual commitment from all ten institutions on the initial objectives, I would still hesitate accepting the assignment. Due to the fact, that the objective is a centralized system, I am concerned that the system might result in more bureaucratic administration on each of the institutions. This is not an objective to have in my opinion. More bureaucracy will not lead to better organizational conditions, better working conditions, or more material well-being for any of the institutions.

Another aspect, that must be considered is the fact, that this is a large public project. If any problems occur along the process, which should be expected, the media will probably not hesitate to write something critical about the development process or the product. Critical and negative media coverage could easily affect the stakeholders opinion of the project. Thus, their motivation and commitment to the project will be in danger, jeopardizing the development process and the whole project.

8.3.2 My Advice

The way, I see it, the assignment is like a ticking bomb, that can go off at any moment considering the size, complexity, and risks it entails. I do not think, that I would accept the assignment, but if I were compelled to accept it, I would choose the development method that best facilitates motivation, commitment and mutual understanding.

The free dialogue and consensus are the key areas, that will have to be nurtured throughout the process, otherwise the project will fail. Problems and perhaps bad publicity will inevitable happen, and for this not to threaten the entire project, the commitment and objectives of the system must have strong roots in each of the stakeholders. To achieve this solid foundation for cooperation and development, the initial process is critical. Commitment and mutual understanding do not just emerge, much social work will have to be done, and all stakeholders must get the feeling, that their needs are taken seriously, and the question is, if it is actually possible.

The functionalist describes the project as having four groups of stakeholders. This is a bit naive, I believe, as it involves a very diverse group of users, a very diverse group of representatives from each of the educational institutions, a financial committee, and a variety of system developers and project managers. Besides these four complex and varied groups, for instance the Ministry, can be expected to have great interest in the project and may even try to influence it during the development process. I mention the different groups of stakeholders to draw attention to the fact, that the stakeholders have differing interests and maybe even conflicting needs; and I have a feeling that the project as the Ministry has proposed it, may be an utopian idea.

My advice to the functionalist is, that he should organize the process in a way that mitigates the risks of the project. He should start with developing a system in close cooperation with just one of the institutions. This particular institution would then get a system that (almost) for sure would address their needs, and the other institutions could then consider if they would benefit from something similar. In order for this approach to work, the system could be built in modules, that could communicate with a central unit. Other institutions would get systems that consist of a combination of the same modules and new modules that fit their special needs.

With this approach, the risk of achieving continuous consensus between ten different institutions is removed. Also the system will be much less bureaucratic, as the central unit is not the main concern here.

I would advice the functionalist to apply an incremental use of OOA&D as a development method, as this method gives room for achieving consensus. I simply do not think, that the little time Scrum allows for stakeholders to reason about objectives and requirements is enough. Also, I would be very cautious letting one person be in charge of the requirements to be implemented. If OOA&D were applied incrementally, letting the users evaluate each increment before proceeding further development, that would be the optimal approach. In this situation, documentation would actually be useful, if modules were to be distributed to other institutions.

9. The Critical Decision: Scrum or OOA&D?

A lot of new issues have been pointed out, which I by myself never would have thought of. Most of the advice given to me is very beneficial, but I see challenges in realizing it as well. Some of the advice are contradictory, which does not make my job any easier. To be able to choose between Scrum and OOA&D as the development method for the project proposal, I will use risk analysis as a tool for supporting and easing the decision.

(F) Choice of either Scrum or OOA&D according to the project

9.1 Discussion of the EIS Project

I have collected the opinions on the project proposal from my three friends; they have described, in great detail, pros and cons of the methods as well as risks in the project. It is clear that my three friends do not agree on what activities should be given most attention in the project. This fact does not make my decision any easier. No matter what, I need to proceed and gain an overview of the activities and risks, which I should be aware of. Surely, I will have all three project managers' arguments in mind while considering the project. I have summarized the main risk areas under five headings:

1. Which challenges do I face substituting and centralizing ten different systems with one?
2. How should requirements elicitation be organized?
3. To what degree should the system be documented?
4. How can I ensure product quality?

5. To what degree should the development process be planned?

I find these subjects very essential when considering the project proposal. The following is a short summary of the other three project managers' views on the risks in the project.

1. Which challenges do I face substituting and centralizing ten different systems with one?

According to my friends; the social relativist and the radical humanist, the involvement of users throughout the development is essential. Both of them believe, that in order to reach a common understanding of the needs, it is important to involve the users much more than I am used to. Another thing the radical humanist pointed out is the fact, that a mutual commitment from all the institutions could be hard to achieve. The reason for this could be the motivation level and the difficulty in identifying common needs, if it is actually possible to have common needs.

The radical structuralist believes that it could be difficult to implement a centralized system, as it is certain that different political interests and conflicts will appear. She points out that when different representatives have different agendas, it is difficult to agree on a common set of requirements. According to the radical structuralist I would be better off, if I developed a decentralized system with individual user interfaces for each institution.

2. How should requirements elicitation be organized?

Once again the social relativist emphasized the importance of user involvement. He found the decision to have only one user representative from each institution critical as all users should have the possibility to be involved in the process of sense making. The radical humanist has the same perception as the social relativist, and she has clearly expressed strong opposition against up-front activities as it does not facilitate motivation, mutual commitment and common grounding.

In opposition to the suggestions made, the radical structuralist has proposed that only one super user should be involved and in charge when eliciting requirements. As a possible solution she has suggested to find the most competent super user and then use one of the existing systems as a model for the new centralized system. Once again she emphasizes the risk of conflicts if more users are involved.

3. To what degree should the system be documented?

It seems that the topic of documentation is where all three project managers have their major disagreements. The social relativist points out a critique towards written documentation, as it creates a static view of the world and freezes the requirements. However, he states that documentation could be used as a mean for reaching a common understanding, but not as a tool for documenting requirements, which I would be inclined to do.

So far, the social relativist and radical humanist have agreed on most of what has been said, but on this particular topic they are of different opinions. The radical humanist has a rather mixed relation to documentation. She states that documentation is not a goal intrinsically, but rather a technique for reasoning about technical implementation details as well as a technique for reaching a common understanding. She actually recommends to use documentation, as the system is to be developed at different places. Not very surprisingly a different opinion on documentation is presented by the radical structuralist. She recommends detailed documentation, as the details would help to prevent misunderstandings and future conflicts occur between the stakeholders.

4. How can I ensure product quality?

Both the social relativist and radical humanist advocate a high degree of user involvement, in order to make the users a part of the development. The social relativist would obtain the high degree of user involvement by continuously validating the system through prototypes. On the other hand the radical humanist has her concerns about maintaining a high degree of motivation throughout the process. She claims that if the motivation is not there, it can be difficult to fulfill the needs. She suggests that the motivation can be achieved by involving and informing all the users throughout the development.

The radical structuralist proposes a need for detailed requirements specifications, which can be used to validate against. A thorough and structured validation process would according to her be necessary, in order to ensure that the system fulfills the goals and objectives set.

5. To what degree should the development process be planned?

The social relativist has a rather non-strict approach to the planning and organization of the development process. He recommends to plan a set of milestones within the time frame and ensure user involvement. Moreover he believes that the planning should be adapted to the current situation.

Surprisingly, the radical structuralist agrees with the social relativist to a certain degree. She prefers a more strict planning in order to ensure and minimize the risk of possible conflicts. She suggests to use a plan-driven development approach, by analyzing, designing, implementing, testing, and validating. This way I can easily monitor and keep a status on the project.

The most controversial suggestion, according to my opinion, comes from the radical humanist who simply claims not to plan the process. She says that it is not possible to plan and would prefer that the effort is put into reaching a common grounding through incremental development.

Varied, but definitely thought-provoking viewpoints, which I myself never would have thought of or given any attention. In order to make the right decision I need to organize all the input I have received and find out what risks and consequences there might be in the method I choose. In the next section I will outline the risks and consequences according to the requirements issues and organize them in two tables.

The **requirements issues** are mentioned in chapter 2.

9.2 The Decision

In this section I will present my choice of development method for the EIS project. I have chosen to use the risk analysis framework to evaluate the two methods. My evaluation is based on the three project managers', and certainly my own, view upon the project and methods. The results of the evaluation are summarized in two tables, see table 9.1 and 9.2, which are based on previously made evaluations of Scrum and OOA&D. The tables contain an overview of requirements issues under each method, activities within the require-

The evaluations of Scrum and OOA&D can be viewed in chapter 6 and 7.

ments issues, the risk of the activities, the consequences and the total risk. The risk analysis calculation is described in further detail in section 2.3.1.

I am aware that the my friends will probably find it inappropriate to use risk analysis for determining my choice of method, but that is okay. I prefer to calculate the pros and cons of a method in order to make a binary decision of either Scrum or OOA&D.

Risk Analysis of Scrum						
	Activities	**Risk Factor**	**Consequences**	**S**	**P**	**T**
Elicitation	**Incremental** ▪ User stories ▪ Pre-game planning	▪ Narrative description of user needs	▪ Generalization difficult	2	2	4
Negotiation	**Negotiation according to existing prototypes** ▪ Sprint planning meeting	▪ Common requirements agreement ▪ Time consuming process	▪ More Costly	5	4	20
Documentation	**Prototypes as documentation**	▪ Not enough focus on architectural design ▪ No written technical documentation	▪ Bad implementation and no proof of agreements made	4	4	16
Validation	**User validation via reviews according to prototypes** ▪ Sprint review	▪ Too frequent possibility for changes	▪ More costly	3	2	6
Process Management	**30 day iterations** ▪ Self-directing teams ▪ Scrum master ▪ Sprint and product backlog ▪ Morning meetings	▪ Loose planning ▪ Lack of clear work division ▪ Too frequent iterations	▪ Project can run out of control	5	4	20
					Total	**66**

Figure 9.1: Risk analysis of Scrum

- Severity: **S**, the economic cost of loss attributed to such an exposure.
- Risk probability: **P**, is the probability of an exposure's occurrence.
- Risk: **R**, the risk related to this exposure is calculated as the product of the two elements; **R = P * S**.

The tables have been divided into the five requirements issues and each of the issues are further discussed according to activities, risk factors, and consequences. In the three rightmost columns, severity (**S:** The severity of an activity in relation to the project), probability (**P:** The probability for failure when applying a certain set of activities in the project) and the total of both **S** and **P** is presented.

9.2.1 Incremental Elicitation as the Optimal Approach

Compared to OOA&D, Scrum has an advantage of having an incremental approach for eliciting requirements. The incremental approach is assisted with easy-to-understand user stories, which gives the users a better understanding of the system in relation to the context. OOA&D on the other hand has a higher risk probability than Scrum as the method suggests a rather formalized approach for eliciting requirements. The approach may have the disadvantage of being difficult for the users to relate to. Furthermore, the use cases might also give a too general analysis of the users' needs.

The elicitation activity has a low severity as I see the elicitation of requirements being not very problematic and fairly easy.

Figure 9.2: Risk analysis of OOA&D

- Severity: **S**, the economic cost of loss attributed to such an exposure.
- Risk probability: **P**, is the probability of an exposure's occurrence.
- Risk: **R**, the risk related to this exposure is calculated as the product of the two elements; **R = P * S**.

Risk Analysis of OOA&D

	Activities	Risk Factor	Consequences	S	P	T
Elicitation	**Incremental** ▪ Use cases	▪ Too formalized descriptions ▪ Too generalized descriptions ▪ No guidance for user involvement	▪ Difficult for user to relate own context and too narrow analysis of the needs	2	4	8
Negotiation	**Negotiation according to predefined models**	▪ Control of models ▪ Difficulty in understanding models	▪ Unnecessary requirements implemented	5	2	10
Documentation	**Detailed written documentation** ▪ Analysis document ▪ Design document	▪ Static documentation	▪ Costly requirements changes	4	3	12
Validation	**Validation against written specifications** ▪ Models	▪ Static documentation	▪ Costly requirements changes	3	2	6
Process Management	**Plan-driven development** ▪ Correction-documents	▪ No process planning ▪ Static view on requirements	▪ Project can run out of control	5	4	20
					Total	56

9.2.2 Models to Handle Negotiation

The activity of negotiation is one of the most critical activities according to the specified project. The severity is set to be high as the possibility for conflicting opinions on what to develop could occur. Ten different institutions, with ten different systems, have to cooperate and agree on a common set of requirements. This seems as a very difficult task and the two methods have two very different approaches. Scrum leaves the prioritizing and negotiation of requirements to the product owner and let him decide in the sprint planning meetings. OOA&D, on the other hand, uses a great deal of modelling with the purpose of describing and helping the users understand the context and what requirements they really need.

The main concern in the negotiation activity is, that an agreement on the requirements is never reached and that the users might change their mind regarding what they think should be developed. At this point, Scrum seems to be vulnerable as a monthly iteration with sprint review and sprint planning can change the course of the project. With OOA&D it is possible to use more time on analyzing and modelling, which makes it easier for the developer and customer to agree on a set of more formalized descriptions. Despite the simplicity, it raises the concern of whether the models can be misused as in the power of models. The use of models places the developer as en expert and gives him the advantage of dominating the negotiation process and thereby manipulate the users understanding. This way the developer can convince the users in choosing a solution, which suits the developer.

9.2.3 Documentation to Ensure Better Implementation

A rather important topic to discuss in this project is requirements documentation. As the project has a time frame of three years, it is seen as a necessity to maintain and keep track of the requirements through documentation. In Scrum the documentation only exists through actual prototypes and are not kept in written form. It is believed that the system should be programmed from the beginning and through refactoring reach a better standard. I am not comfortable with this approach as there are no prescriptions on how to ensure proper system architecture.

OOA&D, on the other hand, is more specialized concerning documentation. The essential activity in OOA&D is to analyze and model, in which the results are written into specifications. The documentation in OOA&D is a good way of describing the requirements, which should make the requirements unambiguous. The downside with much written documentation is the time spent on making and updating these documents. It could make the method vulnerable to changes in requirements during the process, as they would become more expensive to develop.

9.2.4 Validation as a Tool for Quality Insurance

Requirements validation is not seen as a severe risk activity in the EIS project. The element of user validation of requirements is strongly connected with negotiation, and negotiation was earlier ranked as more severe. In Scrum, I see a problem in the continuous user validation through prototypes. The validation occurs every month and influences the development to be in constant change. On the other hand, in OOA&D the validation is performed through written specifications. This approach is not suitable for changing requirements as it will increase the price of the project.

9.2.5 Structured Planning for Successful Development

This is one of the absolute critical topics in the EIS project. The time span has a great influence on the organization and planning of the project. The two methods have two very different approaches for this. Scrum has an incremental approach, which has some risk factors that we cannot avoid discussing. The first risk I see, is the thirty day iteration. Short 30 day iterations during three years require many planning meetings and reviews, which will probably result in much time spent on discussing possible unnecessary changes. Moreover, Scrum does not suggest how to plan the project, it just states that the planning should be for thirty days ahead. The last point is the lack of clear work division, as the teams are self-directing. Along with no documentation, a great deal of reliance is placed on the developers.

OOA&D has a different approach as it can be used as a plan-driven approach. One of the main risks is the static view on requirements, due to the high degree of documentation. There is further no guidelines for managing and planning the process, which I find very unfortunate; especially in this project.

In general, the rankings of the two methods were fairly equal, but in the two activities: negotiation and documentation, Scrum suffered from both a high probability and a high severity. This was mainly the reason why Scrum was not chosen as a development method. Though both methods have useful and risky activities, we find the two methods different in how they prescribe and specify different activities. In general it seems like Scrum has strict guidelines on the management level and OOA&D has more focus on designing and analyzing in detail. The differing focus of the two methods have had a decisive importance in the evaluation and ranking of the different activities. To sum up, my choice of method is OOA&D and to apply the method, I believe, I need to do some refinements in order to address some of the shortcomings of OOA&D.

The suggested refinements will be presented in the next chapter.

10. Refinements of OOA&D

I chose OOA&D as the development method, but I feel the need for refining it, in order to make it suitable for the project. The refinements are necessary, as some of the risks presented by the others cannot be addressed by using the pure version of OOA&D. I am going to discuss my position according to the challenges in each situation and finally I will present a realistic and refined development strategy.

(F) Development Strategy Proposal for the Project

10.1 Responding to Main Risks

I have, in chapter 9, outlined the risks in this project, which my friends have pointed out and I agree they are valid risks.

10.1.1 Handling Stakeholders

This project is characterised by having many stakeholders. The Ministry of Education is the customer and ten different educational institutions will be the actual users of the system. Each of these ten institutions are quite large; comprising many people that will potentially come in contact with the system or somehow be affected by the exchange of the old system with the new one. The other three project managers have raised my awareness of the complexity and difficulty of handling all stakeholders smoothly during the process.

If, I was not to listen to my fellow project managers, I would find it most effective to only involve a single representative from each institution. These ten representatives would then form a committee that I could consult with when necessary. This way, the process of elicitation and negotiation would be less complicated, as the number of stakeholders with influence would be decreased considerably. However, I do recognize possible problems in relation to each of these ten representatives, as they might have conflicting interests, being both the representative of the institution in the committee, as well as the committee's representative in the institution. Their job is difficult, as they have ambiguous responsibilities, risking making decisions that will not be popular in their own institutions. However, relieving some of the responsibility of these representatives would entail involving more stakeholders from each institution and for me, as project manager, to act as a facilitator for getting the stakeholders to agree on a common goal. This is simply not realistic, as it will be too time consuming to accommodate for mutual understanding and common grounding among all stakeholders.

I believe that it is the responsibility of the representative of each institution to inform relevant people and organize the process of eliciting and negotiating requirements at that institution and facilitate the implementation of the system in the organization. I can only be responsible for developing what the stakeholders ask for. However, the process of eliciting and negotiating requirements can be arranged to involve and relieve some of the pressure of the representatives in the committee.

Instead of letting the committee of representatives negotiate requirements, a team of developers could hold ten meetings, one with a group of representatives from each institution to discuss their specific needs and requirements to an administrative system. This way, unnecessary wrangling between the ten institutions will be avoided and the developers will gain valuable insight into the needs of each institution. The important aspect of this approach is, that the group of representatives from each institution consists of different types of users, administrative personnel, as well as managers; it should not be a homogeneous group. Approaching elicitation of requirements this way facilitates the process of gaining a mutual understanding within the institution.

Given the information on needs and requirements from the meetings with each of the ten institutions, the developers can analyze their data and design a solution that will be beneficial for all ten institutions. The developers present their suggested solution to the committee of the ten representatives. Each representative can then present the suggestion to their institution and after that give the institution's feedback through the committee report.

I do believe, that it is possible to construct a joint administrative system for all ten institutions to use, it is a matter of constructing the right system. Dissatisfaction and frustration among the stakeholders during the process, will not be possible to avoid in a project of this size. The important thing is, that it does not mess up the entire project.

By involving a group of stakeholders including users and giving them influence during the elicitation phase, I and the developers can gain much information early in the process, which perhaps would emerge during the process and sabotage the project later in the process. Actually, I do see a point in involving users during the development process to keep them informed. Keeping the users out of the process from requirements elicitation until implementation at the institutions will probably result in some resistance, as they are suddenly, after perhaps 18 months, introduced to a system they have never seen before. Involving users from all institutions in the process do, however, impose certain difficulties, which I will describe later in the chapter.

Another issue, related to involving users at certain points during the development process, is the question of how the user involvement should be organized. There will be no point in asking for user evaluation if it results in ten different requests for changes in the user interface. The evaluation should be organized such that users from each institution can present their change requests to each other before they agree on the change requests, that should be presented to the developers. This entails forming a committee of users from each institution, that could consist of ten super users - one from each institution. The job of this committee will then be to negotiate, which change requests should be presented to the developers.

To sum up, I will handle the diverse stakeholders by delegating responsibilities to stakeholders at each institution. Having a single group of representatives to be responsible for all decisions may entail a too bureaucratic process, which may result resistance to the project from some stakeholders. My friends have informed me that the involvement can create motivation and mutual understanding, therefore I think, two committees should be formed; a committee of user representatives from each institution, and a committee consisting of senior representatives that are responsible for managing the process at each institution. Furthermore, the senior representative at each institution appoints a group consisting of diverse stakeholders to discuss needs and requirements with the developers in the elicitation phase. Involving stakeholders in this manner keeps stakeholders informed and reduces the bureaucratic aspect of the process. I am aware that this constellation adds a risk of conflict between the user and senior groups. But never the less, the elicitation process would benefit from the assurance of involvement of diverse stakeholders.

I wish to avoid getting caught in disputes and conflicts in the committees, the committees should be organized as self-directing teams; the same way as the scrum team is organized. I do not wish to spend time acting as a facilitator for the committees. Their job does not directly concern my daily work, and therefore I do not see it as my job to coordinate their communication of requirements.

In an overall perspective, the development of the front-end of the system should be organized incrementally in collaboration with the committee of user representatives. The analysis and design of architecture and the development of the back-end should proceed in a more linear fashion placing emphasis on comprehensive and thorough analysis and documentation early in the process.

10.1.2 Validation

I am concerned with the way OOA&D validates functional and non-functional requirements, or the lack of description of how to actually do the validation. By validating requirements I mean, to assure the elicited requirements can be implemented and are not conflicting. I need more complete guidance on how to validate. I like the fact that I am able to validate functional requirements by examining the analysis and design documents to determine how they influence each other.

Both the social relativist and radical humanist had some good arguments, when they said they do not like validating through documents. I appreciate their view and see the point, when stipulating that validation against documentation does not give an image of how the users will perceive the finished system. Instead they want a continuous validation of the system, which is something I can accommodate, but only concerning the validation of the user interface. Furthermore, I can only allow comments on the user interface design and request functions involving changes of a smaller scale. Acknowledging greater changes would possibly degrade the architectural design of the system, or it would require an extensive amount of work to re-design the architecture. Therefore, users can only comment on the user interface, once the system architecture has been defined. My point is, that it is only a small fraction of the system the users come into contact with, which is primarily the user interface. The users do not have any idea on how the system is designed and for instance how the data flows between components. My main concern is to develop a system that has a solid architecture, because a solid architecture is open to changes and many different data extractions, which is something that OOA&D is good for. I can take the validated interface of the users and integrate it with the architecture of the system.

I find this very reasonable because the users do not have any skills when it comes to computing theory; they are simply not able to validate an architecturial design. Therefore, I believe the division between a documentation validated architecture and a continously validated user interface incorporates the best of both worlds.

Another point the social relativist and radical humanist seems to forget is, that at some point you have to make a decision on the design of the system. I do not live in a world where I can discuss forever how the system should perform. The system is developed for a customer who wishes to maximize his profit, and therefore deadlines must be kept.

10.1.3 Planning the Development Process

With respect to the overall planning of the development process, I see some shortcomings in OOA&D. Although the method provides a large set of tools for analyzing and designing systems, it does not give any guidelines for structuring the actual development process. What, I am especially missing here, is a way to handle dynamic requirements, as I have to expect that changes will occur with the diverse group of stakeholders.

OOA&D is also a method that bares a risk of the development process becoming plan-driven and bureaucratic. Normally I would not consider this a problem, as I like to plan and control the entire process. But the focus on the diverse group of stakeholders, has convinced me to employ an incremental elicitation process at an early stage of the project. This is to accommodate the views of the social relativist and the radical humanist, where it is not ideal to base the project on early elicitated requirements, because it creates a static view of the plan-driven process.

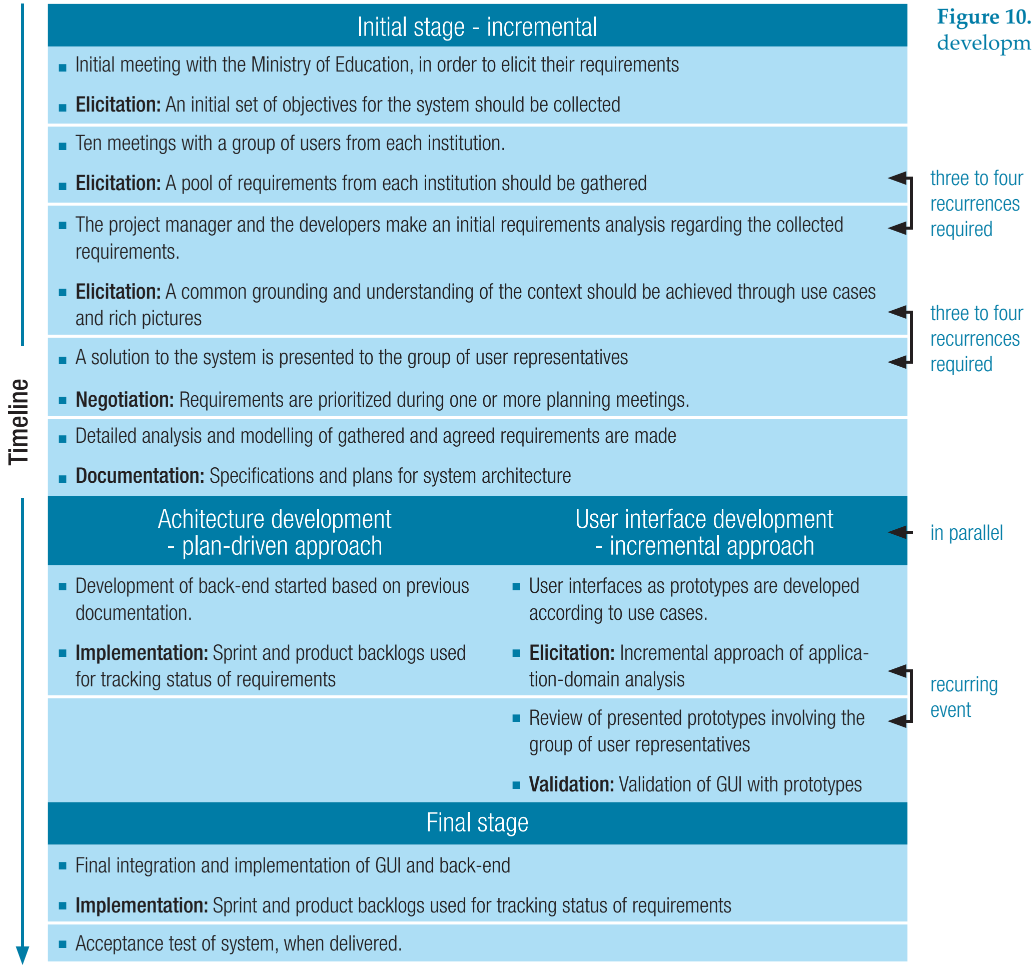
I can see it is reasonable to use OOA&D incrementally in the early elicitation process, as the user representatives will have changing wishes for functionality. By making the elicitation incremental the user can validate the functionality in steps and gain a better understanding of the system. Here I will use throw away prototyping as a tool for reaching a common understanding of the system and to negotiate what requirements should be implemented. But after elicitation I am going to lock the process from user involvement in order to implement the system. The system is only open to changes, when it comes to the design of the user interface. This enables me to involve the users and at the same time plan the rest of the process, which I see as critical in a project with a time span of three years.

I will use the backlogs described in Scrum as a tool for planning the development process. These tools give me an excellent overview of the process, as the product backlog contains documentation that show the overall progress of the project. I will not use the sprint backlog as described in Scrum, as the development is not divided into sprints. I will instead use this tool for managing the elicitation process in the beginning of the project.

By incorporating the views of the other project managers I believe I can modify OOA&D in such a way that it fits the circumstances of the EIS project better.

10.2 Refined Development Strategy

Now I will present the refined development approach based on the OOA&D method. The approach is described in general terms, and is divided into three overall stages. The first stage is an incremental approach to elicit and negotiate requirements with the users and the Ministry of Education. This first stage of the project should help understanding the problem context and aid the gathering of viewpoints from all the stakeholders and the Ministry. The stage is finalized with a thorough analysis and modelling activity, where the focus is to build the foundation for the system architecture. Next the development process is separated into two parallel processes. The development of architecture is plan-driven and is proceeded without any further user involvement. This is done to isolate high-technical architecture decisions from the users, as their understanding within this area must be considered limited. On the other hand the users should be involved in the development of user interfaces, as this is what they are going to work with. This is done incrementally



Initial stage - incremental

- Initial meeting with the Ministry of Education, in order to elicit their requirements
- **Elicitation:** An initial set of objectives for the system should be collected
- Ten meetings with a group of users from each institution.
- **Elicitation:** A pool of requirements from each institution should be gathered
- The project manager and the developers make an initial requirements analysis regarding the collected requirements.
- **Elicitation:** A common grounding and understanding of the context should be achieved through use cases and rich pictures
- A solution to the system is presented to the group of user representatives
- **Negotiation:** Requirements are prioritized during one or more planning meetings.
- Detailed analysis and modelling of gathered and agreed requirements are made
- **Documentation:** Specifications and plans for system architecture

Achitecture development - plan-driven approach

- Development of back-end started based on previous documentation.
- **Implementation:** Sprint and product backlogs used for tracking status of requirements

User interface development - incremental approach

- User interfaces as prototypes are developed according to use cases.
- **Elicitation:** Incremental approach of application-domain analysis
- Review of presented prototypes involving the group of user representatives
- **Validation:** Validation of GUI with prototypes

Final stage

- Final integration and implementation of GUI and back-end
- **Implementation:** Sprint and product backlogs used for tracking status of requirements
- Acceptance test of system, when delivered.

Timeline
three to four recurrences required
three to four recurrences required
in parallel
recurring event

Figure 10.1: Presentation of the development strategy.

in parallel, in order to get a better and mutual understanding and agreement among the different stakeholders.

I have chosen to present the summarized development approach in figure 10.1. Here it is possible to gain an overview of the process according to the time frame given for the project.

Initial Stages

When eliciting requirements I will use rich pictures and use cases from OOA&D. This is an incremental approach, with focus on continuous validation from the users, in order to gain the best possible understanding of the problem context. Both the Ministry and users from each institution should be involved to get the most nuanced picture of the project. The solution, which is presented to the group of user representatives should be refined and validated through reviews. The validation process helps in resolving conflicting requirements.

In the negotiation process, class diagrams, state diagrams, and throw away prototypes are presented. These models help me in describing the solution to the users. The models further give an overview of the system, and thereby a more thorough insight in the system. It is important in this part of the process, that it is open to changes from the users, as it is a process of gaining a mutual understanding of the system.

The documentation process of the system in the early stage is widely focused on the architecture. This documentation plays an important role, because it is necessary to ensure the foundation of the system is well made. If the outcome of this process is well made, it would be easy to integrate user interfaces later in the process. The documentation further serves the purpose of being a common base for knowledge about the design and models for the system. This way the developers will have a common point of reference.

Architecture and User Interface Development

In this stage of the process the back-end is implemented according to the previous made design documents. This approach is plan-driven and isolated from the users. It is not possible to make radical changes suggested by the user during this development phase, as it would be risky to change well considered architectural decisions. Possible changes should also have been minimized by having an initial project setup, which has been performed incrementally.

Along with the implementation of the back-end, the user interface development is processed. This approach is incremental; again it is important to involve the users in order to gain a mutual and common understanding and agreement on the interfaces. The approach would be to apply prototyping, and validate these with the users through recurring reviews.

Final stage

The users are at this point isolated from the development process, and ought not to have more influence before the final system is delivered and put into operation. The integration of user interfaces and architecture, though, might be problematic if validation of both is not done properly. This is, though, not the scope of this project to discuss. After a confirmed acceptance test, the system should be put into use, and a quite different project concerning maintenance is started.

I have now proposed a refinement of OOA&D, which has been inspired by Scrum. This ends the journey.

11. Conclusion

Up until now we have described the challenges within the field of requirements management. We have introduced four philosophical paradigms in information systems development, in order to give alternative perspectives on requirements management. The paradigms have been exemplified in four project managers. Further, a complex project has been introduced as an assignment to the functionalist project manager. To manage and plan the project proposal, two methods; Scrum and OOA&D, have been introduced. Each of the project managers have given their opinion on what method to choose according to the project. The functionalist has considered the advice and refined OOA&D accordingly.

The conclusion is divided into four parts, in which each of the four problem statements will be answered.
The first question in our problem statement is:

- **Which strengths and weaknesses of Scrum and OOA&D do each of the four philosophical paradigms observe?**

11.1 Scrum

Supporters of both the functionalist and radical structuralist paradigm likes the fact that user involvement is restricted to only one representative, namely the product owner. The supporters also agree upon the beneficiaries of backlogs and morning meetings and see them as a tool for measuring the progress of the project. The low degree of documentation, validation, and the openness towards changes are issues both do not like. Apart from this, supporters of the functionalist paradigm like the use of prototypes for elicitation of requirements and they like the fact that the negotiation of requirements is limited to reviews. Supporters of the radical structuralist paradigm, on the other hand, dislike the elicitation process as it is considered to be too time consuming.

Adherents of the social relativist paradigm appreciate the openness towards project planning. They like the self-directing teams and morning meetings, which keep everybody up to date and give room for everybody involved. They also emphasize the importance of obtaining a common understanding, which is facilitated through reviews and validation through prototypes. Supporters of the paradigm also likes the use of backlogs for keeping track of the process. On the other hand, Scrum does not leave enough space for user involvement as the reviews are not held frequently. Finally they find the role of the product owner problematic, as he is given too much power and represents too many different users.

Supporters of the radical humanist paradigm like the independence given to the scrum teams and that the validation of the system is done continuously through prototypes. Further, they do not like the fact that all the responsibility is given to the product owner and the fact that there is not enough space for open discussion among the rest of the users. Moreover, the supporters state that the time frame of Scrum is too restrictive and too much attention is given to the implementation of the system.

11.2 OOA&D

Once again, supporters of the functionalist and radical structuralist paradigm agree upon beneficiaries of the extensive use of tools for analysis, modelling, formalization, and the high degree of documentation. On the other hand, both agree on the lack of tools for validation but like the fact that validation is done against documentation. Supporters of the radical structuralist paradigm prefer the “freedom” inherent in the method, which enables the developers to decide how much time should be spent on different activities during the process. Also people of the functionalist paradigm agree regarding the upfront analysis of OOA&D, where a detailed model determines the design of the system architecture, as a suitable way to conduct analysis.

Supporters of the social relativist paradigm likes the openness towards the degree of user involvement in the method, and the tools for modelling when used for reaching a common understanding of the system. They also agree with the fact, that the developers are in charge and decide the degree of negotiation throughout the process. They dislike being prohibited to only negotiate in the early stages of the process. Overall, adherents of the paradigm like the loose time frame as the developers have the power to decide how much effort for instance, the analysis part should be given. On the contrary they dislike the tools for modelling as they emphasize a general view of all the users. The extensive use of documentation creates a static image of the world, which the adherents of the radical humanist paradigm also dislike. Furthermore, supporters of the social relativist paradigm believe the application and problem domain cannot be separated, as it give a fragmented view of the world.

People of the radical humanist paradigm agree with the people of the social relativist paradigm on the use of tools for reaching a common understanding. Moreover, they agree with the use of UML as it gives a common ground for communication within the development team. On the contrary, they find the understanding of stakeholders naive and dislikes the separation of application and problem domain.

The second question in our problem statement is:

- **Which main risks do each of the four philosophical paradigms notice in the fictitious project?**

Adherents of the functionalist paradigm sees a complication in the process of elicitation as ten different institutions' are involved. Another risk is, that the ten institutions needs would not fully comply with each other, which makes the elicitation process even more challenging.

From a social relativist view, a risk is that the disperse user groups will make it difficult to reach a common understanding of the system and the fact that the need stems from the Ministry and not the users complicates matters further. Furthermore, there is a risk of users being neglected as the role of a super user is applied.

Supporters of the radical structuralist paradigm are concerned with the political motives behind the project. A possible lack of motivation from the users might occur if the purpose of the system is not properly clarified. Moreover, they state that the process of requirements elicitation could be conflicting, as all ten institutions and the Ministry have their own political motives.

People of the radical humanist paradigm see the main risks as being the diversity of the educational institutions and especially the motivation for cooperating with the Ministry. They believe that free dialogue would not be possible in a project with so many participants. Moreover they state, that the media can have an impact on stakeholder interest and involvement, because negative critique might lead the stakeholders to loose their commitment. Finally, adherents of the radical humanist paradigm fear the system will be used as a controlling element benefitting the Ministry.

The third question in our problem statement is:

- **Which method would a functionalist project manager choose for the fictitious project, when taking the views of the three alternative paradigms into consideration?**

The risk analysis rankings of Scrum and OOA&D have been fairly equal, in reference to what developers of the functionalist paradigm consider important. But in two activities, Scrum has had shortcomings. It had a too high probability for negotiation of requirements amongst the developers and users. Further, the lack of documentation seemed problematic as a functionalist sees it as an essential part of a development process.

Overall, Scrum and OOA&D have different approaches towards system development. Scrum has a set of strict guidelines on the management level, whereas OOA&D focuses on designing and analyzing in detail. In this project the functionalist project manager prioritizes the design and analysis features more than the overall structuring of the process. Based on the arguments above, the functionalist prefers the use OOA&D because of the emphasis on tools for modelling and documentation.

The fourth question in our problem statement is:

- **With the chosen development method in mind, how can a possible development strategy be organized?**

The refined OOA&D method is divided into two stages. The first stage is incremental and concerns the elicitation, negotiation, and documentation of requirements. The next stage is split into two processes, which are run in parallel. One process is plan-driven and concerns the design and implementation of the system architecture, with no user involvement. The other process is incremental, where the user interface is implemented and validated in order to reach a mutual understanding of the system. This is done in coorporation with the users. Finally the architecture and user interface are integrated and sent to acceptance test.

What separates the traditional OOA&D method from our strategy, is that negotiation is seen as a separate activity, which is closely related to the elicitation process. Also, our strategy specifies the need for three to four recurrences in the process of elicitation and negotiation. This opens the method for changes, from the users, and facilitates a common understanding of the system.

Our division of the implementation stage, into the architectural and user interface processes, has been done to assure a solid architectural design, and at the same time, facilitate a mutual user understanding. The mutual understanding is emphasized through an incremental user interface development, and at the same time giving the developers space for implementing and assuring a stable and solid architecture.

During the implementation of the system, sprint and product backlogs are used for tracking the status of the requirements. This gives a status of the development process.

11.3 Postscript by the Functionalist Project Manager

In retrospect the involvement of my three professional friends in the project, has been useful to me. The fact that I have shared my concerns of the project with other developers, has helped me realizing issues, which I might not have seen otherwise. Especially the complexity of the user groups, was an area where I gained a more nuanced understanding of the project. It is hard to admit, but it is an area, which I would not have given much attention if I had planned the project by myself.

This whole process has emphasized the importance of being more open minded and looking into the unknown. It has strengthened my awareness of factors, other than those already known.

11.4 Postscript by the Authors

It is our opinion that the course of the project has been useful. In many ways it has been a different process, as the use of the four paradigms for nuancing requirements management has been a new approach for us. At one level the paradigms have given us a nuanced image of requirements management, which is what we have sought in this project. At the other level it has been a useful tool for analyzing the importance of social aspects of information systems development.

The use of the narrative technique has also been a challenge. It gave the opportunity to dig deeper into the paradigms and adapt the thinking. Moreover, it has made our descriptions of paradigms, development methods, and the fictitious project more detailed and realistic. We feel the technique has helped us to understand the paradigms at a higher level.

But what about the results of this project? Are they creditable? Yes, we believe they are, as our descriptions of the four paradigms and the statements given are based on acclaimed theories. We have also made a point in referencing the statements of the project managers to validate our statements. Further, the fictitious EIS project is closely related to an actual project, and our refinements of the OOA&D method are subtle and feasible.

References

[1] ANDERSEN, H., and KASPERSEN, L. B., *Klassisk og moderne samfundsteori,* Hans Reitzels Forlag, 3. udgave, 2005, 8741202333.

[2] BASKERVILLE, R. L., and STAGE, J., ”Controlling Prototype Development Through Risk Analysis,” *MIS Quarterly*, vol. 20, no. 4, 1996, pp. 481-504.

[3] BOEHM, B., ABI-ANTOUN, M., PORT, D., KWAN, J., and Lynch, A., “Requirements engineering, expectations management, and the Two Cultures,” *Proceedings of the IEEE International Symposium on Requirements Engineering* (ISRE’99), IEEE, 1999, pp. 14-22.

[4] BOEHM, B., “Get Ready for Agile Methods, with Care,” *Computer Society Press*, vol. 35. no. 1, 2002, pp. 64-69.

[5] BOEHM, B. W. and DeMarco, T., “Software Risk Management,” *IEEE Software*, vol. 14, no. 3, 1997, pp. 17-19.

[6] BOEHM, B., and TURNER, R., “Balancing Agility and Discipline: Evaluating and integrating Agile and Plan-driven Methods,” *Proceedings of the 26th International Conference on Software Engineering* (ICSE’04), IEEE, 2004, pp. 718-719.

[7] BOEHM, B., and TURNER, R., “Using Risk to Balance Agile and Plan-driven Methods,” *IEEE Computer Society*, vol. 36, no. 6, 2003, pp. 57-66.

[8] BOURQUE, P., DUPUIS, R., ABRAN, A., MOORE, J. W., AND TRIPP, L., *The Guide to the Software Engineering Body of Knowledge*, Version 2004, IEEE Press, 2004, 0-7695-2330-7.

[9] BURRELL, G., and MORGAN, G., *Sociological Paradigms and Organisational Analysis*, 1. ed., Heinemann, 1979, 0-435-82131-8.

[10] CHURCHMAN, C. W., *The Design of Inquiring Systems: Basic Concepts of Systems and Organization*, Basic Books Inc., 1972, 0465016081.

[11] EBERLEIN, A., and DO PRADO LEITE, J. C. S., “Agile Requirements Definition: A

View from Requirements Engineering", *Agile Alliance*.

[12] FAVARO, J., "Managing requirements for business value," *IEEE Software*, vol. 19, no. 2, 2002, pp. 15-17.

[13] HERLEA, D. E., "User Participation in Requirements Negotiation," *ACM SIGGROUP Bulletin*, vol. 20, no. 1, 1999, pp. 30-35.

[14] HIRSCHHEIM, R., and KLEIN, H. K., "Four Paradigms and Information Systems Development," *Communications of the ACM*, vol. 32, no. 10, 1989, pp.1199-1216.

[15] HORAI, H., "Multi Viewpoint Analysis in Requirements Process," *Proceedings of the second international software architecture workshop* (ISAW-2) *and international workshop on multiple perspectives in software development* (Viewpoints'96) *on* SIGSOFT 96 *workshops*, ACM Press, 1996, pp. 175-179.

[16] JOHNSON, R. A., "The Ups and Downs of Object-Oriented Systems Development," *Communications of the ACM*, vol. 43, no. 10, 2000, pp. 68-73.

[17] KIENHOLZ, A., "Systems ReThinking: An Inquiring Systems Approach to the Art and Practice of the Learning Organization," *Foundations of Information Systems*, 1999.

[18] KØPPE, S., and COLLIN, F., *Humanistisk Videnskabsteori*, 2. udgave, DR Multimedie, 2003, 87-7953-363-9.

[19] LARMAN, C., *Agile & Iterative development - A Manager's Guide*, Addison-Wesley, 2003, 0131111558.

[20] MALAN, R., COLEMAN, D., and LETSINGER, R., "Lessons from the Experiences of Leading-Edge Object Technology Projects in Hewlett-Packard," *Proceedings of the 10th annual conference on Object-oriented programming systems, languages, and* applications (OOPSLA'95), vol. 30, no. 10, 1995, pp. 33-46.

[21] MATHIASSEN, L., MUNK-MADSEN, A., NIELSEN, P. A., and STAGE, J., *Object Oriented Analysis & Design*. 1. ed., Forlaget Marko, 2000, 87-7751-150-6.

[22] NUSEIBEH, B., and EASTERBROOK, S., "Requirements engineering: A roadmap," *Proceedings of the Conference on the Future of Software Engineering* (ICSE'00), ACM Press, 2000, pp. 35-46.

[23] POPPENDIECK, M., "Wicked problems," May 2002, http://www.poppendieck.com/wicked.htm.

[24] SCHWABER, K., *Agile Project Management with Scrum*, Microsoft Press, 2004, 0-7356-1993-X.

[25] SCHWABER, K., and BEEDLE, M., *Agile Software Development with Scrum*, Prentice Hall, 2002, 0-13-067634-9.

[26] SOMMERVILLE, I., *Software Engineering*, 6. ed., Addison Wesley, 2001, 0-201-39815-X.

[27] SUTHERLAND, J., "Agile Development: Lessons Learned from the first Scrum," *Agile Project Management Advisory* Service, vol. 5, no. 20, 2004.

[28] TURKE, D., FRANCE, R., and RUMPE, B., "Limitations of Agile Software Processes," *Agile Alliance*, pp. 43-46.

PART THREE

Reflection and Further Research

This part contains a reflection on the journey of the four project managers as well as the theory applied. The following seven topics will be discussed:

- The perception of software engineering in our education
- The impact of a certain world view on the systems development process
- Tailoring development methods to projects
- Literature critique
- The theoretical impact on present research
- The validity of the theory used according to present research
- Further research

We would like to emphasize the need for a broader world view in systems development. It is not possible for us to set up definitive recommendations, but we believe that openness towards different world views will be beneficial for future system development. ■

Reflection

We wish to reflect on our scientific contribution throughout the project. We will discuss issues within the project, which we find interesting but have not discussed previously.

We will take a critical standpoint to the literature, which we have referenced, described, applied, and based the project upon. The chapter concludes with thoughts on how our findings can inspire further research in the field of computer science.

A Nuanced Perception of Software Engineering in our Education

We chose to represent the four perspectives to illustrate their differences in their views upon system development and to emphasize the need for observing the development process in a more nuanced way. As mentioned in chapter 1, a traditional functionalist view upon system development has been dominating the field of software development. This view has also been dominating our studies, even though we have had a communicative insight during our studies. The literature we have encountered during our education, has not mentioned anything about what blessings and curses the choice of a certain perspective comes with. During this project we have dived into the four perspectives and discussed what a choice of perspective means for the development process. We have come to know that perspectives employ

certain restraints on your world view, which again affects your way of developing systems.
This particular subject has not been given much attention in the academic software engineering community lately, and viewing system development from theoretically different perspectives is not something which has been practiced much after Hirschheim and Klein [7]. It is a shame that not much attention has been given to this area and that the primary focus has been the process of implementation [1]. Our results in the literature survey showed a clear overweight of articles written within the traditional functionalist paradigm, which had a dominating view of requirements as being static. With this knowledge we aimed, in the second part, to nuance the view on requirements.

We believe that the world cannot be viewed as an observable truth [3]. It is of great importance that the subjective influence and participation in systems development are emphasized, so developing a system is not only understood as a manufactoring a technical product, but as developing a product which has to function in a dynamic social environment.

This is the reason why we aimed to describe how differently system development can be understood and what complications are seen as critical. We see it as an essential part of system development that different non-technical aspects are taken into account.

We therefore hope our work can raise an interest amongst researchers and emphasize the necessity for understanding system development from different philosophical perspectives. Our contribution should be seen as a teaser and a starting point for interested researchers.

We do not believe that just one distinct method can be applied to all kind of projects. And when choosing a method one will have to adjust it according to the project. We are aware that the practitioners employ and adjust methods and do not practice just one kind of method (see Appendix A). The strategy we presented in our second part of the project is a refinement of OOA&D, which emphasizes the fact that it is difficult to use a pure version of a method.

However, the necessity for modifying methods is not something startling, as G. B. Davis in his article *Strategies for information requirements determination* [5] from 1982, states three reasons for eliciting a set of correct and complete requirements. Based on this, he finds it necessary to select an overall strategy for a project, which outlines the strategy for the project. Moreover, a number of methodologies should be selected for the elicitation and documentation of the requirements. These methodologies are selected according to the degree of uncertainty in the project, as illustrated by Davis in figure 2.

We agree with Davis at some points, but we have had a different focus with the four paradigms. We have not focused on the project, but on the involved people in the project, and how their world view affects the development. We reach the same conclusion, but have taken a different road to reach the destination.

The World View has an Impact on the Development Process

Throughout the project we have emphasized the fact, that project managers' world view has a great impact on the way system development is approached. Hirschheim and Klein are not the only ones to employ this line of thought. In other scientific areas, for instance the field of management studies, Burrell and Morgan are widely used to create awareness to scientists of their own role:

> *"...certain fundamental theoretical and philosophical assumptions underlie any piece of research - there is no such thing as a totally objective or value free investigation"*
>
> ***- Hopper [8 p 429]***

This quote emphasizes that a researcher's own underlying philosophy and beliefs tend to "colour" your way of thinking. Therefore, it is of great importance that one is aware of implications when choosing a method, as a certain focus can shift attention away from critical issues in a project.

The Development Method in Relation to the EIS Project

Looking back at the EIS project the question is whether it is a realistic example of a software development case. The answer to this is yes, as the case is written with a strong foundation in the VUE project [12]. Prior to this project, we conducted a case study (see Appendix A) of the VUE project. It gave insight in the development process, which is why we used this case as inspiration. We, therefore, consider the EIS project to be plausible.

When reading the article it is clear that all involved stakeholders and the stages in the EIS project are identical to the VUE case. The fact that the rich picture, in figure three, in the article has been validated by developers further documents the validity of the case.

We also have to consider whether our modification of the OOA&D is only applicable in the EIS project or if it can be used in other projects.

Looking at our refinements of the method, we have focused on user involvement and tools for process planning. This has yielded changes in the method which calls for an incremental elicitation and negotiation process with a high degree of user involvement. Further, we have introduced the use of backlogs from Scrum to manage the development process.

Allthough our refined method follows the stages known from traditional methods, it is not certain the method is applicable to other projects than the EIS project. As described before, modifications of development methods are considered normal in the software industry, as the method used has to comply with the organization in question. Therefore, it would not be plausible to use exactly the same refinements of the method in other projects.

Literature Critique

In this section we take a critical standpoint with regard to the literature we have applied throughout the project.

Burrell and Morgan

Burrell and Morgan's description of the four philosophical paradigms have been the main theoretical source in this project. These paradigms are well known and used within many branches of scientific research. The paradigms have also served as a framework or guideline for researchers, who validate their work against the established Burrell and Morgan theories. Hirschheim and Klein, which have been a source of inspiration for us, is an example of this.

What you have to ask yourself, is whether or not it is a good idea to place yourself within a certain philosophical paradigm, and use this as a reference. The fact that Burrell and Morgan describes their grid of four paradigms as being mutually exclusive dichotomies is something that we find troublesome. Chua [4] agrees, as he aims a strong criticism against this division and points out, that authors like Bhaskar and Habermas, have different views. Burrell and Morgan's division rests on the assumption, that humans are either determined by their surrounding society or are completely autonomous. Bhaskar states that society changes continually as a result of human actions, although they have originated prior to humans. Habermas is of the conviction that although people shape their own meanings, they are still bound by structures of a dominating society. We stand side by side with Bhaskar and Habermas, as we are of the conviction that the paradigms cannot be strictly separated. It seems as if Hirschheim and Klein has come to the same conclusion, as they state:

> *"The four paradigms, as depicted through the stories, are not as clear cut nor as animated as they are made out to seem. There is overlap and their differences are overstated..."*
>
> ***- Hirschheim and Klein [7 p 1202]***

Although this gives a less rigid image of the paradigms, problems still exist, which we believe is a relic from Burrell and Morgan. We will address Hirschheim and Klein's theory shortly.

Stanley Deetz [6] also raises questions to Burrell and Morgan. He explains that Burrell and Morgan gains rapid acceptance by mainstream science, by showing that they are mainly functionalist minded. This standpoint was adapted from sociological functionalism, and from this paradigm, "other" paradigms were identified and described. Conclusion have also been raised as to what the subjective - objective terms connote. More specifically the substitution of the term interpretive with subjective connotes that its member's are more subject to management control. At the same time the objective term falls in favor of the functionalist studies. Instead Deetz introduces a linguistic grid, which focuses on discursive moves and social relations, in which he introduces a revised grid, which consist of a local/emergent - elite/a priori scale instead of subjective-objective, and dissensus-consensus instead of radical change-regulation. Deetz rejects the paradigm term, as his aim is to give a presentation where there are no division. This is due to the discursive approach where a number of conflicts reside within each discourse, and the further edges are not clearly separable.

We see strengths and weaknesses in Deetzs discursive approach. We find Burrell and Morgans paradigms useful for dissecting scientifical research by using different philosophical paradigms. On the other hand, we have learned that the paradigms are rarely viewed separately and in fact supplement each other. Therefore, Deetz has a point, when he employs a discursive approach. We have come to the conclusion, that the division of the paradigms is useful when analyzing scientific approaches. They can help to raise awareness of the practitioners own role in the development process.

Hirschheim and Klein

When reviewing the framework set up by Hirschheim and Klein, we see both good and bad things.

First of all we see a value in the way that a paradigm is constructed as an archetype. This way, what is in focus and is not in focus can be illustrated very clearly. It has been of great value to us, when describing how the world views effect the development process.

Another aspect is whether the paradigms are useful for practitioners. We believe that this is not directly the case. The reason for this, is that it would not be possible to have such a clear cut view on how to develop systems. No matter what paradigm you belong to, you are a part of an organization, which is under the regulation of legislative and social laws, economic interests, and human relation aspects, which will transcend the boundaries of the paradigm. On the other hand the paradigms have been proved useful in scientific research, such as this master thesis. The separation of the paradigms is useful for dissecting complex theories, and comparing their core characteristics. Overall we see Hirschheim and Klein's framework as a useful analyzing toolkit for raising awareness of different development approaches.

Like Nurminen [11] points out, we see a problem in the way that Hirschheim and Klein adopt the conflict term between social classes and put them into an organizational context. This transcendence from macro to micro level, seems to happen without any adaption. It is most apparent in the radical structuralist paradigm, where the class struggle between the owners and workers seems spurious and out of date. We believe that this struggle could be more elegantly described using Morgans *Images of organizatio*n from 1986. The nature of the organization is described using eight metaphors [10], which can shed light on how the class struggle manifests itself on an organizational level. A more detailed description of the composition of organizations can be found in Henry Mintzbergs *Structure in fives - Designing effective organizations*, where he states, that an organization consists of five parts, which influences each other[9]. In our opinion both Morgan and Mintzberg would be useful for adapting conflict, to fit in a micro context.

Theoretical Impact on Present Research

Although Burrell and Morgan's theories are well established and widely used, we believe that their theories, do not solely have a good impact on present research because we are of the belief that the framework can be restrictive, rather than be helpful guidelines for good research. For example a young researcher formulates a new theory that does not fit into any known theoretical frame. The new theory has to be tested against other theories in order to gain recognition, amongst fellow researchers. This validation is caused by an underlying meaning in a particular paradigm, which is something that has been acquired through a lengthy socialization process [11 p 39]. Therefore a paradigm carries with it a set of rules, which states what is considered to be good science and what is not. We believe these rules can hold new theories back, when put forth by innovative researchers. Further, innovative researchers are forced to formulate their ideas using terms into which their ideas do not fit [6]. This is problematic in the sense, that evolution within a field calls for new ways of thinking, and new ways of thinking means challenging what is already accepted.

Theoretical Validity According to Present Research

As mentioned before, Hirschheim and Klein's paradigms are adapted from Burrell and Morgan's theories from 1979, and the framework itself is 17 years old. Not much has been written since, and you have to wonder why. Is it because this thinking approach has gone out of style and research focuses on other aspects of computer science? We believe that the approach is out of style, even though Hirschheim and Klein are well referenced amongst researchers. No new articles are written about the paradigms impact on systems development, and the latest critique we have been able to find is written by Nurminen in 1997.

The big question is whether the paradigms are still valid in their current form. The answer is yes, as they are based on Burrell and Morgan's sociological theories, which are widely acclaimed. As mentioned before, new theories tends to be evaluated against socially established theories. Therefore Hirschheim and Klein are seen as a valid reference for researchers. Though, it could be discussed, if the paradigms should be refined in relation to modern software development approaches, as this pracsis is evolving fast.

Further Research

We have introduced four perspectives to systems development in order to create a nuanced image of the functionalist perspective of requirements management. The reason for this is our assumption, that the functionalist perspective is the most predominant perspective in practice. This assumption is supported by the conducted literature survey.

What we have learned is that the perspectives introduced by Hirschheim and Klein are very narrow minded, and we wonder if any of these perspectives are in use by practitioners and researchers. We believe this is not the case, as the real world does not allow for such single mindedness. But it would be interesting to study the degree of purity in practitioners perspectives and which aspects they employ from the different perspectives, if they use a mixed approach.

Society is becoming more and more dependent on technology, which raises a need for more complex systems. This complexity encourages the need for a more varied view upon system development and a need for more flexibility. We must admit that due to the increasing complexity of system development it is not possible to obtain an objective view and therefore we are not able to give any definitive or complete guideline for how system development should continue. We just want to emphasize the awareness of differences and the need for integrating them into the development.

We find it worrying that even though the changes occurring in software development is becoming a natural part of the development process [2] the plan-driven approach, where requirements are seen as static and an objective part of the development, is still in dominance.

Like any other field, computer science has to evolve with the environment in which it rests. Sometimes the evolution has to come, not in the form of technology, but from the persons who create and use the technology. We believe that the introduction of social sciences can create awareness of previously overlooked aspects in computer science:

> *"Without such an awareness there is a danger that people become entrenched within well-defined and righteously guarded positions; unproductive claims and counter-claims may proliferate and constructive academic debate may be stifled"*
>
> ***- Hopper [8 p 430]***

After all, the systems have to be used by people and are developed by people. To make our point clear you can look at the medical science, which is divided between the medical scientific research and the social aspects of patient care. Research has shown that patients that get more personal care and has the same doctor throughout his/her stay at the hospital, heal faster, and as a result leave the hospital quicker. This has nothing to do with the medical advances made, but is related to the social environment of the hospital.

In relation to computer science, it is clear that the social sciences are not in focus when viewing the results of our literature survey. Therefore social sciences should not be neglected in future research in the field of computer science, as it might be here future breakthroughs reside.

Finally it might prove beneficial for researchers and practitioners to take a look inside instead of focusing on tools. We see the focus on tools, as a symptom of the dominating functionalist paradigm, and in order to view the field of computer science from a social science perspective, practitioners and researchers have to ask themselves the question; *Do I believe that social science has anything to offer me?*. As it is with social science, it is the people involved who constitute society, and if they do not see the relevance of social theories, then what is the use [8]?

References

[1] ANDERSEN, K. S., HVID, L. D., KLAUSEN, J., and VARMA, G., “Evolving and changing requirements: a literature survey,” 2006, pp. 15-46.

[2] BROOKS, F. P., “No silver bullet: Essence and accidents of software engineering,” *Computer Magazine,* April 1987, pp. 10–19.

[3] BURRELL, G., and MORGAN, G., *Sociological Paradigms and Organisational Analysis*, 1st ed., Heinemann, 1979, 0-435-82131-8.

[4] CHUA, W. F., “Radical Developments in Accounting Thought,” *The Accounting Review*, vol. 61, no. 4, 1986, pp. 601-632.

[5] DAVIS, G. B., “Strategies for information requirements determination,” *IBM Systems Journal,* vol. 21, no. 1, 1982, pp. 4-30.

[6] DEETZ, S., “Describing differences in Approaches to Organization Science: Rethinking Burrell and Morgan and Their Legacy,” *Organization Science*, vol. 7, no. 2, 1996, pp. 191-207.

[7] HIRSCHHEIM, R., and KLEIN, H. K., “Four Paradigms and Information Systems Development,” *Communications of the ACM,* vol. 32, no. 10, 1989, pp.1199-1216.

[8] HOPPER, T., and POWELL, A., “Making Sense of Research into the Organizational and Social Aspects of Management Accounting: A review of its Underlying Assumptions,” *Journal of Management Studies*, vol.22, no. 5, 1985, pp. 429, 37.

[9] MINTZBERG, H., *Structure in Fives: Designing Effective Organizations*, Prentice Hall, 1993, 0-13-854191-4.

[10] MORGAN, G., *Images of Organization*, Sage Publications, 1986, 0-8039-2830-0.

[11] NURMINEN, M. I., “Paradigms for Sale: Information systems in the process of radical Change,” *Scandinavian Journal of Information Systems*, vol. 9, no. 1, 1997, pp. 25-42.

[12] LARSEN, P., BAADSGAARD, H. P., ENGELL, H., MORTENSEN, H., PEDERSEN, T., and THORUP, H., ”Beretning om Videregående Uddannelsers Edb-system (VUE-projektet),” *Rigsrevisionen*, no. 11, 1999, http://www.ft.dk/BAGGRUND/statsrev/0998.htm, date: 07-06-06.

PART FOUR
Appendix

This part contains a case study, which we have conducted on our 9th semester; it is this case we have used as inspiration for the EIS project in the thesis.

The goal of this study was to elucidate how evolving and changing requirements are handled in a software industry. This was done by conducting a case study of a study-administrative information system (STADS) for all higher education institutions in Denmark.

The empirical data was collected using qualitative interviews with respondents representing the developing company, customer, and end-users of the system. ■

The Requirements Engineering Process in a Software Project: a Case Study

Abstract

The goal of this article is to elucidate how evolving and changing requirements are handled in a software industry. This is done by conducting a case study of a study-administrative information system (STADS) for all higher education institutions in Denmark. The empirical data was collected using qualitative interviews with respondents representing the developing company, customer, and end-users of the system. We conclude that the strategy for developing new releases of the system follows the waterfall model, where the the focus heavily lies on documentation. An interesting observation made, is that the requirements negotiation is a very time consuming process, and, in this case, more time consuming than the entire process of developing the system.

Keywords

Requirements management, requirements process, requirements elicitation, requirements negotiation.

1. Introduction

A consistent problem is how to manage evolving and changing requirements in the software engineering process. The SWEBOK states that it is impossible to have static requirements throughout an entire development process because it cannot be expected that the initial requirements are the same through the entire process.

Appendix A A Case Study

In reference to the literature survey, the academic world has little to say about the reasons for evolving requirements [1]. Instead, the research literature focuses on methods and tools to assist software developers in handling changes.

The motivation for conducting this case study lies in the literature survey conducted on how evolving and changing requirements are handled in the academic literature. The result shows that the main focus is on how to "*solve*" or handle the changes with methods and tools. Very little is said about the cause of changes. With this case study we would like to relate the result, with the result of the literature survey.

VUE is the danish shortening for "*a computer system for higher education*"

The aim of this paper is to describe how evolving and changing requirements are handled in a software project developed for the public sector in Denmark. The name of the project is VUE, a common system for all higher education institutions in Denmark with the purpose of centralizing and standardizing student data. The focus of this study is not the entire system, but the study administrative system, a subsystem.

Rest of the paper is organized as following: **section 2** describes the case STADS. **Section 3** describes our method for conducting the case study. **Section 4** describes the process of developing new versions of STADS, a process containing six stages. **Section 5** presents a short analysis and **section 6** concludes.

2. The STADS Project

"***Study administrative system***" is shortened STADS

To be able to understand the requirements process in this particular case, we will briefly introduce the background [8] of the project. The case study is about the STADS project. The general idea behind the project was to make a central system for all higher institutions in Denmark. This way it would be easier for the Danish Ministry of Education to compile nationwide statistics.

The project was first presented to the The Finance committee of the Danish Parliament in May 1991 and was at this point called VUE. The project consisted of further development of an institution system plus a modernization of existing systems to the Danish Ministry of Education.

A common it-system for all of the institutions was agreed to be the best solution. Even though it was a central system, the system would be installed at each institution.

Until October 1995 the system was released in several versions, all with minor problems. This resulted in the need for further funding before the final system could be put in operation. Before this happened the Danish Ministry of Education had the superior responsibility for the project, but now the responsibility was handed out to the individual institutions. It was expected that the system should be put into operation in August 1996.

The reorganization in October meant that the project management group in the Ministry of Education was closed and the responsibility was handed over to the VUE-centre as an independent institution under the Ministry of Education. Also a financial reorganization took place - besides the given funding, all further expenses for additional functionality was funded by each of the institutions. Further, it was decided that the institutions should have the possibility of deselecting STADS. This resulted in the University of Copenhagen and Copenhagen Business College deselecting STADS. Today the STADS committee consists of eight institutions placed throughout Denmark.
In 1999 the VUE centre was closed and the development of STADS was bought by a subcontractor, namely WM data. Further a group called the STADS collaboration was established and they now had the responsibility for STADS.
The financial expenses of the entire VUE project have been estimated to be at least three times as much as first assumed [5]. There is still activity in the project, as new versions are released once or twice a year.
In the next section we briefly discuss our research approach.

The institutions are:

- *Aalborg University* (AAU)
- *Roskilde University Centre* (RUC)
- *The Danish Technical University* (DTU)
- *Aarhus Business College*
- *The Danish pharmaceutical University*
- *The royal veterinary and agricultural college*
- *The danish engineering colleges*

3. Research Method

The focus of this section is to describe our approach for gathering empirical data concerning the case. The data was, in a large scale collected through semi structured interviews in accordance to interview guidelines made by Steinar Kvale [7]. The characteristics and focus of our interviews can be viewed in figure 1.

Appendix A A Case Study

Figure 1: The Characteristics of the Conducted Interviews

- Three interviews were conducted
- Similar interview guide structure for all interviews to facilitate data comparability
- The respondents received the guides minimum two days beforehand.
- The same two students conducted each interview with either the developers, customers or users, to avoid too much preconceived opinions

Interview setup

The main focus of the interviews was to end up with a description of the process of the case project, as the respondents perceived it. To facilitate this we designed a series of cards, which the respondent could move around on a large piece of paper, while describing what he/she was trying to illustrate. The cards had pre-made illustrations of elements and activities, which we imagined would be involved in the project process. As an introduction, it was made clear to the respondents, that the cards were only guidelines, and they should feel free to draw and write anything, they felt would emphasize elements in the process.

An example of the cards can be seen in figure 2. We have chosen to interview the following people, as they represent different organizational levels. By interviewing them we can obtain a more nuanced image of the entire development process as they will present different viewpoints.

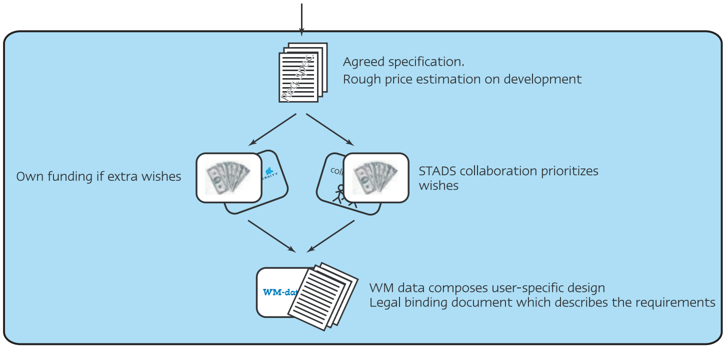

Agreed specification.
Rough price estimation on development
Own funding if extra wishes
STADS collaboration prioritizes wishes
WM-da
WM data composes user-specific design
Legal binding document which describes the requirements

Figure 2: Snapshot picturing the second stage of the software process.

The Customer Interview

The Customer Respondent is the head of the Budget and Planning Administration at Aalborg University. He was originally involved in another part of the VUE project.

The Developer Interview

- **The Commercial Project Manager** on STADS from WM data. Her role in the process is in the beginning and in the end of the project. She enters the negotiation if implications between customer and developer occur
- **The Technical Project Manager** on STADS from WM data. He is involved in the process from the beginning to the end with the responsibility for design and technical solutions
- **The Project Manager** from WM data. He has not been directly involved in STADS.

The User Interview

The End-user of STADS at Aalborg University. She is the representative from Aalborg University in the STADS collaboration.

Capturing and validating data

We have chosen to summarize certain passages we believe hold important information, which can be useful in answering our research question. Here the voice recordings of the interviews make it possible to capture exactly what the respondents said.

To validate the summaries they were sent to the respondents for confirmation. Furthermore, elaborating questions were sent to clarify uncertainties in our summaries.

Furthermore, our empirical data was designed to fulfill the property of triangulation, which means, that our data is derived from three different sources. The first source is the documents collected regarding the background history of VUE [8]. The second source are the interviews, which give an inside perspective of the process, as told by the respondents. The third source, is documents describing how the requirements for STADS were actually handled, for instance the design specifications. These three types of data contributes to validating our data, leaving us with a more nuanced image of the project.

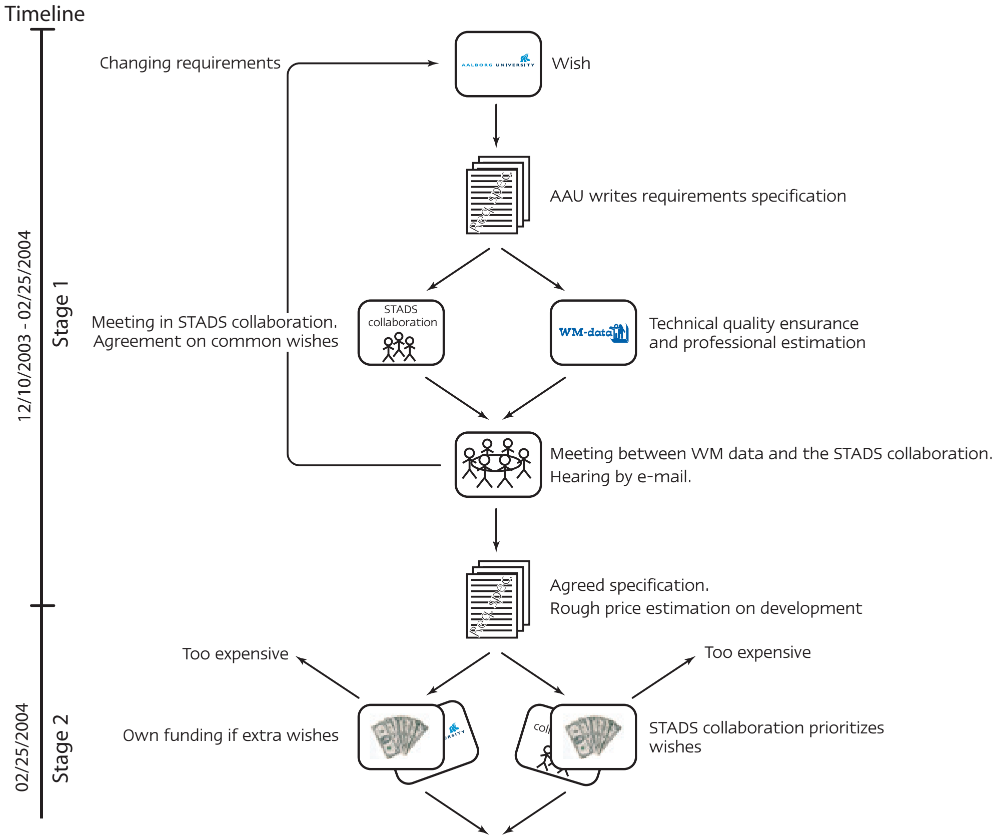
4. The Software Development Process

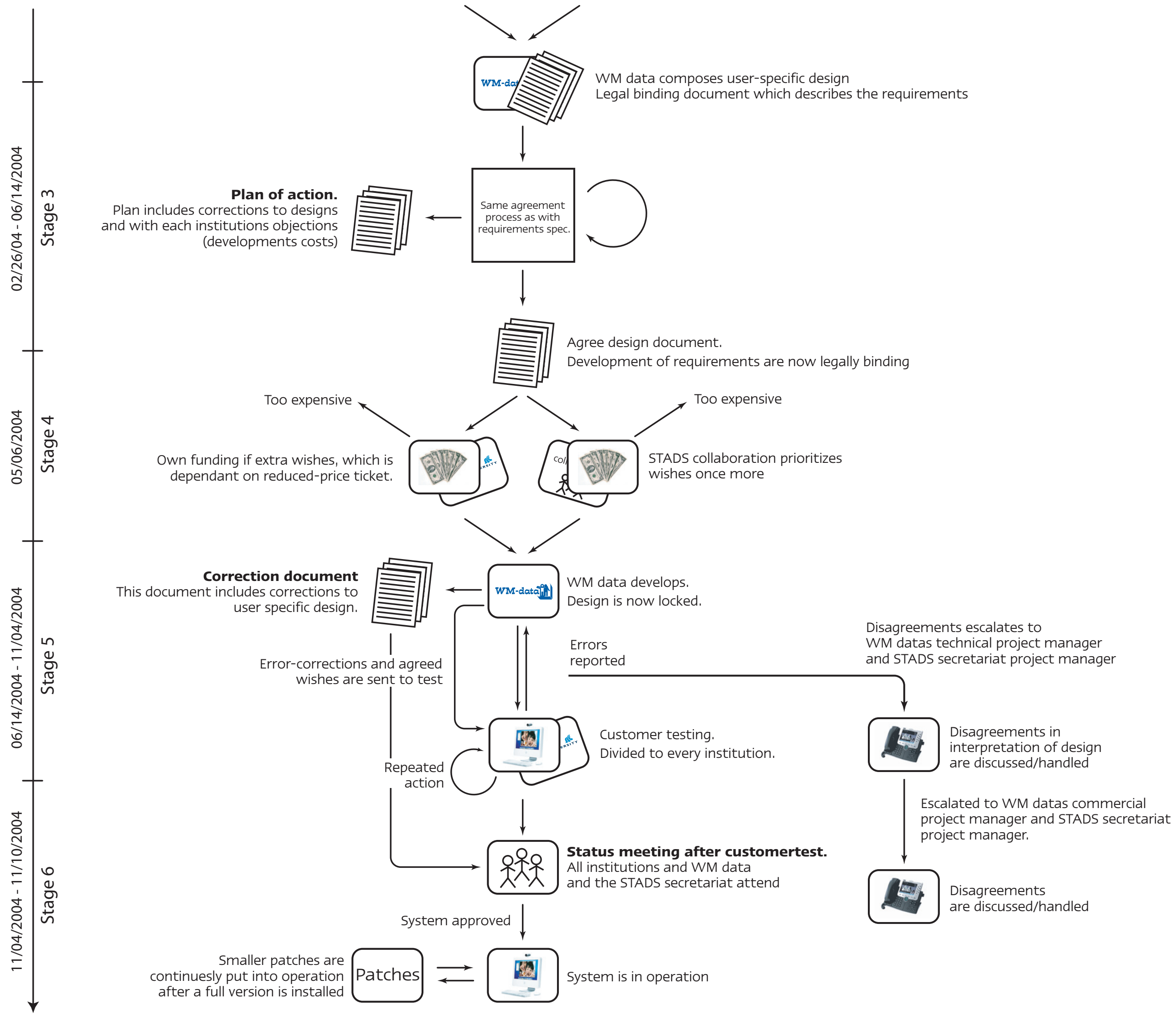
The purpose of this section is give a brief overview of the entire process and how new versions of STADS are developed. We have combined the three respondents' descriptions of the process into one single illustration, see figure 3. The illustration will be described in further detail in the next section.

Stage 1: The process of developing a new version of STADS starts when the institutions have new requirements or changes. Each institution writes a requirement specification for each of their requirements. The requirements specification is then validated at meetings with both WM data and the STADS collaboration. The outcome on this stage is an agreed upon requirements specification.

Stage 2: After validating the new requirements, they are prioritized. This is done in a meeting in the STADS collaboration, where all the institutions have

Figure 3: The process described by WM data and the end-user


Timeline
12/10/2003 - 02/25/2004
Stage 1
02/25/2004
Stage 2
Changing requirements
AALBORG UNIVERSITY
Wish
Req. spec.
AAU writes requirements specification
STADS collaboration
Meeting in STADS collaboration. Agreement on common wishes
WM-data
Technical quality ensurance and professional estimation
Meeting between WM data and the STADS collaboration. Hearing by e-mail.
Req. spec.
Agreed specification. Rough price estimation on development
Too expensive
Too expensive
Own funding if extra wishes
STADS collaboration prioritizes wishes


02/26/04 - 06/14/2004
Stage 3
05/06/2004
Stage 4
06/14/2004 - 11/04/2004
Stage 5
11/04/2004 - 11/10/2004
Stage 6
WM-data
WM data composes user-specific design
Legal binding document which describes the requirements
Plan of action.
Plan includes corrections to designs
and with each institutions objections
(developments costs)
Same agreement process as with requirements spec.
Agree design document.
Development of requirements are now legally binding
Too expensive
Too expensive
Own funding if extra wishes, which is
dependant on reduced-price ticket.
STADS collaboration prioritizes
wishes once more
Correction document
This document includes corrections to
user specific design.
WM-data
WM data develops.
Design is now locked.
Errors
reported
Disagreements escalates to
WM datas technical project manager
and STADS secretariat project manager
Error-corrections and agreed
wishes are sent to test
Customer testing.
Divided to every institution.
Disagreements in
interpretation of design
are discussed/handled
Repeated
action
Escalated to WM datas commercial
project manager and STADS secretariat
project manager.
Status meeting after customertest.
All institutions and WM data
and the STADS secretariat attend
Disagreements
are discussed/handled
System approved
Smaller patches are
continuesly put into operation
after a full version is installed
Patches
System is in operation

to agree upon new functionality. This process is repeated until everybody has agreed on, which requirements the STADS collaboration wants WM data to develop. As an outcome, WM data creates a detailed user-specific design specification, which is a formal specification made by WM data with the users requirements specification in mind. The design specification then contains user interfaces and descriptions to these.

Stage 3: After receiving the design specifications from WM data the institutions develop a plan of action. In this plan corrections to the design specification according to the previous requirements specification are marked down.
Stage 4: The design specification is presented to the STADS collaboration and they have to decide, which of the design suggestions they want developed. This is done at another prioritization meeting after which the design specification is finally signed by both the customer and WM data. This specification is now legally binding and the users are not involved in the project until in the later test stage.
Stage 5: At this stage the design specification is turned into code. In parallel, the system is tested by WM data and the end-users. Errors and corrections are reported back to WM data and they handle it in three ways. Either they correct it, put the correction in a correction document or reject the requirement.
Stage 6: When the test phase is finished, a final status meeting is held where the STADS collaboration and WM data are present. At this meeting the system is approved and afterwards the system is launched. It is worth noting, that smaller patches are released several times in parallel with this described process. These small patches correct minor problems or difficulties with the system, as it is sometimes urgent to make smaller corrections.

5. Analysis

The results of our conducted literature survey stipulated that some aspects of requirements engineering were well covered in academic literature and some were not. A large amount of research focuses on the traditional software engineering school and more specific on the ideas of the plan-driven approach.

With this in mind, the entire requirements management process of STADS described earlier can be described as following the classic waterfall development paradigm. The development is divided into different phases, which are similar to phases in the waterfall model. The fact that each phase is finalized and not recurring and the large amount of documents used throughout the development are some of the characteristics of this development paradigm. Furthermore, it was stated from WM data that they are following this model, and they describe this model as being divided into four stages: analysis, design, implementation and test. It is further noticeable to see that each stage in the entire process is finally completed and cannot be revised in the later stages.

A wide range of requirements engineering tasks are described in literature and many solutions on how to handle these tasks are presented. Some of the most well described areas are about requirements specifications, requirements management, the general requirements process and requirements elicitation. With these topics in mind, we emphasize certain sub processes to clarify, which aspects of requirements engineering are in focus in practice as opposed to the theory.

First snapshot

In this snapshot (see figure 4) topics as negotiation and requirements management are in focus. Eight organizations have to agree upon, which requirements they want the user-specific design made for. This introduces several political issues and opens up for an interesting aspects on how to handle the negotiation process.

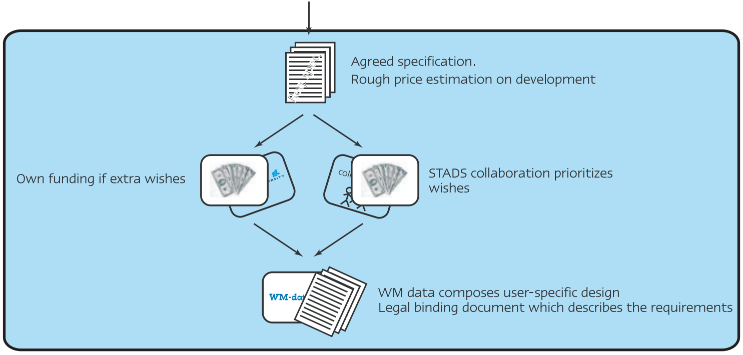

Agreed specification.
Rough price estimation on development
Own funding if extra wishes
STADS collaboration prioritizes wishes
WM-da
WM data composes user-specific design
Legal binding document which describes the requirements

Figure 4: First snapshot

The negotiation can be problematic as all the institutions have to agree on a set of common requirements. With the background of STADS in mind, the project was originally proposed from the Danish Ministry of Education. It was not up to the single institution to decide whether they wanted to be involved in the common it-system but instead a governmental decision. This decision might have produced some complications regarding the standardization of their daily work tasks and procedures.

It raises questions as well on how to obtain a common understanding of which work tasks the system should facilitate and how it should support these tasks. It is worth noting that the individual institutions have the possibility to choose own requirements, if they fund it themselves. If they do, this implementation is applicable for all the institutions and might therefore conflict with other institutions work procedures.

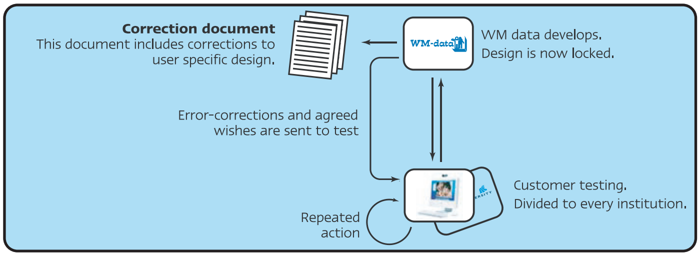
It is not surprising that WM data describes the process of negotiation as heavy, challenging, and time consuming. The academic literature states that this is a very complicated and important issue in the development of software. In particular it has been claimed that the negotiation between different stakeholders and the reconciliation of their conflicting viewpoints has been recognized as a major problem [4].

This sub process is basically like Stage 1 and Stage 3 in figure 3, as the institutions in these stages have to either prioritize or agree on common requirements. The outcome of these stages are documents describing the revised requirements.

Second snapshot

This snapshot concerns the development stage and how changes and errors at this point are handled (see figure 5). This is interesting because corrections are negotiated back and forth between customer and developer. These change requests are documented in standardized forms, which improves traceability between corrections. The information in this change document consist of: *correction descriptions, correction date, correction author and further.*

Figure 5: Second snapshot


Correction document
This document includes corrections to user specific design.
WM-data
WM data develops.
Design is now locked.
Error-corrections and agreed wishes are sent to test
Customer testing.
Divided to every institution.
Repeated action

In this stage the institutions report errors and corrections to WM data, and WM data handles the requests. Some of the requests are about errors, and these are corrected by WM data simultaneously. Other requests concern misunderstandings from the institutions according to the user-specific design specification, which then are rejected by WM data. In both situations the requests are closed and they are not mentioned in the correction document. A small amount of requests might be mentioned in the correction document, as they can cause misinterpretations within the institutions because of a vague formulated design specification. In this case the correction to the design specification is described in further detail in the correction document. The last possible situation is if the institutions cannot use the functionality, as it is described in the design specification. In this situation WM data will adjust the functionality right away and the institutions fund the expenses. All this is added in the correction document.

The main focus in this process is always to ensure that the developed system is consistent with the design specification. This is why the correction document is made after the design specification and is used as a supplement to the design specification.

We have identified three different types of specifications in the process. The first one being the requirements specification written by the user, the second a specification "*translating*" the users requirements into what WM data calls a user-specific design specification and the third a correction document. The way the entire "*process*" of developing a "*requirements specification*" is described by WM data resembles the way SWEBOK explains how a requirements specification can be divided into smaller specifications with different purposes. What WM data describes as requirements specification is what is referred to as a user specification in SWEBOK. The user-specific design specification is similar to the system specification in SWEBOK. WM data's correction document can somehow be related to what is called review in SWEBOK. The review is understood as a validation of the already existing documents [2].

It is interesting how changes are handled in this situation and further why the changes occur in the first place. One possible explanation for changes occurring in this late stage could be, that the customer and the developer do not interpret the design specification in the same way, which in this case would result in a negotiation process with the purpose to solve an eventual conflict.

Third snapshot

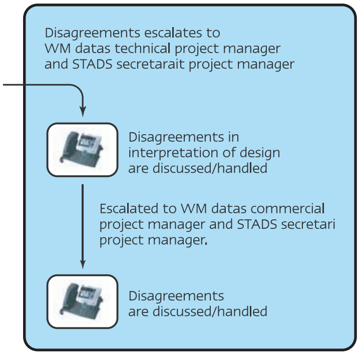

Disagreements escalates to
WM datas technical project manager
and STADS secretarait project manager
Disagreements in
interpretation of design
are discussed/handled
Escalated to WM datas commercial
project manager and STADS secretari
project manager.
Disagreements
are discussed/handled

Figure 6: Third snapshot

This stage concerns negotiation of conflicts regarding changes. Described at snapshot two, there were two possibilities to handle change requests, either to correct or to send the errors up the ladder for further discussion. The first step in this negotiation is for the Technical Project Manager and the STADS Secretariat Manager to discuss the interpretation of the design specification. The main focus for the discussion is how WM data and the customer have interpreted the user-specific design specification. The design specification is legally binding, and agreed by both the customer and the developing company, which is why the interpretation question is important.

If an agreement on the interpretation is not reached at this level, the conflict is sent further up the ladder to the Commercial Project Manager of WM data and the STADS Secretariat Project Manager.

As described in snapshot two, WM data work with more than just one specific documentation. Even though they try to overcome the conflict of misunderstandings regarding the written documentation, they occur. WM data seem to be aware of this factor and have already a procedure for handling negotiation on misinterpretations. A procedure which does not include any specific tools but simple face to face communication. In the literature survey [1] these issues are mostly handled with tools [3, 6].

6. Summary

When looking at the overall process and how changes are handled several questions arise. The first interesting aspect is how requirements management is handled and in more detail how change requests and traceability is handled. We can see that all requests are handled carefully with information on who, how and when a change request has occurred. This gives the possibility for tracing the requests and further it documents the entire development process. It is worth noting that the analysis phase is very time consuming and changes are incorporated in this part of the process. But as development starts, new change requests are handled, but only in a way, in which new functionality or changes according to the design specification would be made in the next version of the system. Even though the overall process is like the waterfall model, smaller sub processes are more of an incremental character.

For instance the negotiations in the analysis phase are a repeated action. Another point of interest is the time used on negotiation on requirements between the institutions. This process had a span of about seven months whereas development and test lasted for about five months. This is interesting, because the actual process of agreeing on which requirements should be developed is actually longer than the actual development process. Furthermore, the important documents in the entire process is the design specification, which should be unambiguous and easy for the customer to interpret correct. This specification has a standardized format with screenshots and descriptions of the functionality, which should be made. The main task for the developing company is to make this specification unambiguous and precise.

7. Conclusion

This case study will be summarized from three different levels of perspective: rationale, strategy, and operations.

From the rationale point of view, we see how requirements are seen upon from the interviewed respondents. The process has the characteristics of the traditional waterfall development paradigm and changes in the process are understood as dynamic and are handled with three types of specifications: requirements specifications, user-specific design specifications and corrections documents. This way of handling requirements indicate a heavy focus on the need for good traceability and documentation regarding the corrections of requirements. Moreover, the requirements negotiation phase lasts up to seven months, whereas the actual development period lasts for about five months.

The development process indicates an aspect of how to handle conflicts regarding misinterpretations in written documents. This is carefully described by WM data as a specific process. If such conflicts occur, the process describes how they are negotiated back and forth between project managers from WM data and the STADS Secretariat. In the academic literature this topic is not described on a rationale or strategic level, only different tools for supporting negotiation are suggested.

We have not identified specific tools used in the development process, but certain tools must have been used to keep track of the different versions of specifications.

Compared to the academic literature, the study indicates great focus on documentation, which corresponds to what the literature describes. Furthermore, it seems like WM data recognizes that misinterpretations of documents occur by describing the specific process of how they handle conflicts in this matter.

Limitations

Due to the time span of the project, the empirical data of this paper, comes with a few limitations.

To achieve a more nuanced image of the software process more interviews, with more than one respondent from each organization, could be conducted. As people can have different personal opinions of what occurred, it could be beneficial, to the validation of our data, to conduct more interviews in order to cross reference our results.

Furthermore, our empirical data describes only this specific case. For this reason, we are not able to generalize and draw general conclusions, as to whether this process is similar to other software projects.

Acknowledgements

We wish to thank our supervisors for their professional guidance.

References

[1] ANDERSEN, K. S., HVID, L. D., KLAUSEN, J., and VARMA, G., "Evolving and changing requirements: a literature survey," 2006, pp. 15-46.

[2] BOURQUE, P., DUPUIS, R., ABRAN, A., MOORE, J. W., AND TRIPP, L., *The Guide to the Software Engineering Body of Knowledge*, Version 2004, IEEE Press, 2004, 0-7695-2330-7.

[3] BRIGGS, R. O., and GRUENBACHER, P., "EasyWwinWin: Managing Complexity in Requirements Negotiation with GSS," *Proceedings of the 35th Annual Hawaii International Conference on System Sciences* (HICSS-35'02), IEEE Computer Society, 2002, pp. 10.

[4] HERLEA DAMIAN, D. E., EBERLEIN, A., and SHAW, M. L., GAINES, B. R., "The Effects of Communication Media on Group Performance in Requirements Engineering," *Proceedings of the 4th International Conference on Requirements Engineering* (RE'00), IEEE, 2000, pp.191.

[5] DJØRUP, H., "Ministerium slipper i VUE-sag," *Ingeniøren*, 2000, http://ing.dk/arkiv/1800/vuesag.html, date: 07-06-06

[6] HOH IN, OLSEN, D., and RODGERS, T., “A Requirements Negotiation Model Based on Multi-criteria Analysis,” *Proceedings of the 5th IEEE International Symposium on Requirements Engineering* (RE’01), IEEE, 2001, pp. 312–313.

[7] KVALE, S., *Interview – En introduktion til det kvalitative forskningsinterview*, 1. udgave, Hans Reitzels Forlag, 2000, 87-412-2816-2.

[8] LARSEN, P., BAADSGAARD, H. P., ENGELL, H., MORTENSEN, H., PEDERSEN, T., and THORUP, H., ”Beretning om Videregående Uddannelsers Edb-system (VUE-projektet),” *Rigsrevisionen*, no. 11, 1999, http://www.ft.dk/BAGGRUND/statsrev/0998.htm, date: 07-06-06.

About the Authors

Karsten Stig Andersen

Although he is the oldest member of the group, this 28 year old is young at heart. His interests, during his time of study, has been systems methodology with special attention on developer and customer communication, and the influence of computerized information systems in organizations.

Jeppe Klausen

Throughout his studies he has worked as a self-employed web application developer, and IT administrator at a local advertising agency. The main focus of the 23 year old, during his education, has been technical challenges, implementation, and graphical solutions.

Gauri Varma

At 25 years of age, she has been an active member on the study board at The Faculty of Engineering and Science at Aalborg University. In the past two years she has worked as a student counselor. Her main interests are impact of social science on systems development and the handling of human computer interaction.

Louise Dalum Hvid

At 26 years of age she has had a strong theoretical interest in the field of science. In particular systems development methodology, communicative theories and their influence on development methods, and organizational theory.