A survey of incremental development in Norwegian IT industry

Sunniva Finsvik Andersen and Per Christian Nødtvedt

November 23, 2001

NTNU
Faculty of Physics, Informatics and Mathematics
Department Of Computer and Information Science
SIF8904 Project
Abstract

During the last decade, incremental software development has evolved into one of the main practices of software engineering. The reasons for this are not clear, and the empirical evidence supporting incremental development as a superior method to other alternatives is not conclusive. This report gives an image of the state of the art of incremental software development, in addition to the results of a survey. The survey was designed to find out whether Norwegian IT-companies used incremental development, and in a lesser extent to find out what the industry meant by the term incremental development.

The survey was implemented in two stages: first a series of interviews with 5 companies, followed by a questionnaire distributed to a larger group of entities. The first round of interviews was designed to gather knowledge about the field of incremental software development, and to find qualitative data. The second part aimed at finding quantitative data to prove that incremental development was an advantage for software development. Due to a low response rate on the questionnaire, the study became qualitative, and the results show that incremental development is a sort of best practice in the parts of the industry that responded to the survey.

The conclusion to the project is that qualitative evidence indicates that there are advantages to using incremental software development over non-incremental software development.
Preface

This project was carried out at the Software Engineering group at the Department of Computer and Information Science (IDI) at NTNU. The authors are fifth year students at IDI, and the project is part of their curriculum. The overall context of the project is a larger research project called INCO - Incremental and Component-based Development. INCO is a joint project between IDI, NTNU and the Department of Informatics (Ifi) at the University of Oslo (UiO), with the support of NFR (The Research Council of Norway).

We would like to thank our advisor at the Software Engineering group, Reidar Conradi, for help and advices throughout the project. Also, we would like to thank Magne Jørgensen at Ifi, UiO and NTNU PhD student Tore Dybå who have contributed to our project. Lastly, we want to give our thanks to those companies and individuals who participated in interviews and answered the questionnaire, and also PROFIT and IKT-Norge for supplying e-mail addresses.

Trondheim, November 2001

Per Christian Nødtvedt and Sunniva Finsvik Andersen
Contents

1 Introduction .................................................. 3
  1.1 Report structure ......................................... 4

2 Project context ............................................. 5

3 Software lifecycle models ................................. 7
  3.1 The waterfall model ..................................... 8
  3.2 The V-model ........................................... 8
  3.3 Prototyping ............................................ 9
  3.4 The Spiral model ...................................... 10
  3.5 Cleanroom ............................................ 11

4 Incremental development .................................. 13
  4.1 Incremental development in general .................. 13
    4.1.1 Definition ....................................... 13
    4.1.2 Terminology ..................................... 14
  4.2 Using Incremental Development ....................... 15
    4.2.1 Types of increments .............................. 15
    4.2.2 Phase span of increments ....................... 15
    4.2.3 Length and scheduling of increments .......... 17
    4.2.4 Organization of increments ..................... 18
  4.3 Advantages of incremental development ............. 20
  4.4 Extreme programming ................................. 20
  4.5 RUP - Rational Unified Process ..................... 21
    4.5.1 Phases in the Rational Unified Process ....... 23
    4.5.2 RUP process model .............................. 24
    4.5.3 Process framework .............................. 25
  4.6 DSDM - Dynamic System Development Method ........ 25
    4.6.1 RAD .............................................. 25
    4.6.2 RAD and DSDM ................................... 25
    4.6.3 DSDM principles ................................ 26
    4.6.4 DSDM lifecycle ................................... 26
  4.7 Catalysis ............................................... 27
    4.7.1 Levels of description ............................ 27
    4.7.2 Modelling constructs ............................ 27
    4.7.3 Underlying principles ............................ 28
Chapter 1

Introduction

In the last few decades of software development, various development methods have been used to formalize the development process. One of the predominant methods has been the waterfall model, largely because of its simplicity. But as the history of software development shows, this has not been all successful. The industry has seen projects overrun both time and budget, and projects have been failing to meet the users’ needs. According to [Gra92], the traditional problem areas of software development are:

**Estimation**: it is difficult to make an accurate estimation of cost, effort, time or size of large systems.

**Changing requirements**: the customers do not know what they want in the future at the beginning of a project

**Testing and maintenance**: 40% of the software development effort is typically spent on testing, normally at the end of the project.

**Customer confidence**: the customer loses confidence in the project as the product is not visible and often delayed.

There are several examples of failed software projects. These are from the Norwegian history of software development [SU]:

**TRESS** Project at Rikstrygdeverket - delayed by 5 years and loss of more than 1,2 billion NOK

**Norwegian Railways** Spent 200 MNOK on non-functioning ticket system in 1995.

**SMART ticket system** In 1995, a ticket system in Norwegian post offices, collapsed when 50,000 teenagers wanted concert tickets.

**Kreditkassen bank** Lost 320 mill. NOK in 1996 on new EDP system.

With projects overrunning both time and budget and products not meeting the needs of the user, one can draw the conclusion that the models and processes used in the last decades of software development have had some serious flaws. New models and processes have been developed to overcome the problems of the traditional development methods.
Today, much of the industry is going away from the traditional waterfall model and starting to develop software iteratively in increments. Although the idea of incremental development has been in the literature for several years, it is only the last few years that it has become popular in industry, for instance through process products such as RUP from Rational Software.

1.1 Report structure

Chapter 2 presents the angle chosen on this project and the wider context, i.e. the INCO project.

Chapter 3 of this document will give a brief summary of some well-known software lifecycle models.

Chapter 4 gives a general overview of incremental development and alternatives for using it in projects. It also gives an introduction to some of the incremental development models.

Chapter 5 describes the survey carried out. The first part of the survey was an interview of five companies about their use of incremental development, the second part was a questionnaire distributed to a larger number of companies in Norwegian software industry. The background for this survey were the following hypotheses: 'There are no advantages to using incremental development' and 'Incremental development does not require more/other customer/developer competence than non-incremental development'.

Chapter 6 gives the conclusion of the investigation carried out in this project and presents some interesting points of further research.
Chapter 2

Project context

This project is part of a larger research project called INCO - Incremental and Component-based Development. This is a joint project between the Department of Computer and Information Science (IDI), NTNU and the Department of Informatics (Ifi) at the University in Oslo (UiO), with the support of NFR (The Research Council of Norway).

The background for INCO is the high costs and high failure rate of software projects today. Some of the proposed solutions to these problems include incremental development and component-based development, as these methods can help reduce development time and cost and increase the quality of the developed software. Although incremental development and component-based development have been known in the industry for some years, these fields of software development are still immature. The main problems lie in the area of planning and cost/risk estimation. The overall goals of the INCO project [INC] are therefore:

1. Advancing the state-of-the-art of software engineering, focusing on technologies for incremental and component-based software development.

2. Advancing the state-of-the-practice in software-intensive industry and for own students, focusing on technologies for incremental and component-based software development.

3. Building up a national competence base around these themes.

4. Disseminating and exchanging the knowledge gained.

In our project, the focus will be on goal 1, with a focus on technologies for incremental development in Norwegian software industry. The INCO project defines some open issues that can serve as basis for this incremental development project:

- Length of increments, number of phases involved
- How to manage time-to-market vs. traditional qualities? How to manage the risks in general?
- Estimation models for this, based on Use Cases?
- Coupling to COTS?
• What projects are most well-suited for this?

• What training is required from developers and managers?

• What of this works in practice?

Based on this, the focus of this project will be on how incremental development is used in practice in Norwegian industry.
Chapter 3

Software lifecycle models

The definition of a software lifecycle model is a that it is a prescriptive model of the development process from the first ideas about the system until the system is taken out of use [McC96]. However, most of the models only focus on the period from the first ideas about the system until the first release of the system.

In this project, a lifecycle model is viewed as a model for how a software development process should be carried out. Since the beginning of software development, software lifecycle models have been sought, to help [Bal98]:

- Find the stages in the software development process
- Find and organize the activities to be carried out
- Define the parts of the product
- Find completion criteria for the parts of the product
- Find responsibilities and competences
- Find the standards, methods and tools to be used

In contrast to a lifecycle model, there is the so called ’code and fix’ method. This was predominant in the beginning of the software development history. This method consists of two steps:

1. Write a program
2. Find and correct errors in the program

As this method proved to be inefficient, the software development industry sought other ways to develop software.

This section will discuss some of the tried and tested software development methods. We will start with the waterfall model, followed by the V-model. Then we will briefly discuss prototyping, the spiral model and the cleanroom approach. In chapter 4, a discussion of incremental development will be followed by the presentation of some lifecycle models with focus on incremental development. There, we will look at extreme programming, followed by Rational Unified Process, Dynamic System Development Method, Catalysis, Select and Staged Delivery.
3.1 The waterfall model

The waterfall model is the classic development model for software projects. It consists of isolated phases linked together in a chain, where one phase will be completed before the next one can be started. It was introduced in [Roy70] and it is still in widespread use today. This probably stems from its simplicity, although its weaknesses have been pointed out by many sources.

Figure 3.1 shows that the development process consists of five main phases: requirement analysis, design, implementation, testing and maintenance. One would normally complete one phase before moving on to the next, although it is possible to remount the flow to correct earlier mistakes. This means that fundamental changes done late in the process are more expensive than changes at the beginning of the project.

The waterfall model works nicely in an environment where the requirements are quite static. In systems where the demands for reliability and safety are stringent, the verification of each phase demanded by the model will provide valuable input to the development. In systems where the requirements are fluid, setting a timeline can be demanding. Missing schedules are a normal problem when using this model on a project unsuited for this kind of development.

3.2 The V-model

As a response to the weaknesses in the waterfall model, the V-model was introduced. It emphasizes testing, and forces the developers to write test plans during each phase of the process.

Figure 3.2 shows that the phases in the waterfall model been slightly modified. The requirements phase remains unchanged, but the design phase has been divided into two. This gives one phase of overall design, and another with detailed design. Implementation remains unchanged too, while testing gets split into three. These three test phases
correspond each to one of the phases leading to the implementation, giving the V-shape. The advantage of the V-model is a more detailed approach to testing. Defining the tests at different levels in the design process forces the developers to take the concrete test cases into account. It inherits many weaknesses from the waterfall model, in that it aims at completing a phase completely before going on to the next. This means that the model will be useful in the same cases as the waterfall model.

3.3 Prototyping

The aim of the prototyping model is to make a runnable version (prototype) of the system at an early stage in the development.

Prototyping is useful for:

- Collecting, detailing and understanding requirements
- Collecting practical experiences with the system
- Specifying user interfaces
- User training before the final system is delivered

Figure 3.3 shows the stages in a prototyping model. It consists of making a plan for the prototype to be developed, what the prototype should do, developing it and running the prototype so that it can be evaluated.

In [PRO] a distinction is made between two main types of prototyping, evolutionary prototyping and throw-away prototyping.

**Evolutionary prototyping** This type of prototyping aims to develop a running system to the customer. The development starts with the requirements that are well understood.
**Throw-away prototyping** The goal of throw-away prototyping is to collect and validate the requirements of the system. In contrast to evolutionary prototyping, it starts with the requirements that are poorly understood.

When making a prototype, several techniques can be used. These include:

- Executable specification languages
- Application generators and 4GLs
- Composition of reusable components

According to [Bal98] the advantages of prototyping are as follows:

- Reduction of development risks
- It can be used in combination with other development models
- A prototype can be developed quickly
- Prototyping encourages creativity

And the disadvantages:

- It causes higher development costs
- The limitations of the prototype are often not known

### 3.4 The Spiral model

In the late eighties, the spiral model was introduced. It consists of four consecutive phases that lead back to the "starting point". This means that an iteration of the spiral will complete a partial product, and this intermediate result will provide the basis for the next iteration of the spiral. The metaphor of an outward going spiral was possibly chosen to show that subsequent layers are added to an existing product, resulting in a larger and hopefully more complete product in the later iterations.

The four phases in the original spiral model introduced by [Boe88] are planning, risk analysis, engineering and evaluation. As mentioned above, the result of the evaluation phase is whether one should take another round or release the product as is. Spiralling is
closely related to prototyping (see section 3.3), and especially evolutionary prototyping, where one builds a small functional version of the system before extending it into a full version.

The spiral model will excel in an environment where the requirements of the system change. The different increments allows developers to concentrate on certain aspects of the project, since other aspects can always be introduced in later increments. Projects on a tight schedule will also profit from the incremental approach, as priorities can be set. If a feature has low priority, it will be implemented in an increment late in the spiral. A disadvantage of the model is that it demands motivated customers to give feedback in the evaluation process.

3.5 Cleanroom

Cleanroom development is based on an analogy with semi-conductor fabrication, where the products are developed in a sterile and controllable environment. The model is based on the notion of static verification and increments.

Figure 3.5 shows a proposed cleanroom process as envisioned by [Lin94]. A formal specification of the system is developed, before the product is scheduled into increments. Each increment is implemented by one team before it is formally verified by another team to find defects in the code. The programming is structured, meaning that only a limited number of constructs and abstractions are used, facilitating inspection.

The 'traditional' approach to cleanroom development prohibits unit testing of components. Programmers are in some cases not even allowed to compile their code before it is sent to verification, forcing them to thoroughly think through the problem while coding rather than hacking and testing. After code verification, the system is tested using statistical methods to see if it meets the criteria of the requirements specification. These tests will in most cases find the errors that slipped through the verification process.
Experiments have shown that cleanroom development results in systems with few errors at no significantly larger cost than conventional development methods. This stems largely from the fact that errors are noticed early in the process, and costly reworkings late in the project are avoided. Examples of successful implementations of the cleanroom process can be found in [LM88]. A drawback to the model is that it relies on the skill and commitment of very good developers. It has also been criticized for ignoring advances in testing methodology, in that it eschews accepted testing principles without refuting the underlying theory [Bei97].
Chapter 4

Incremental development

In this project we will concentrate on the field of incremental development, which is the software development method that much of the software industry is shifting towards today. Before carrying on a discussion in the field of incremental development, it is important to clarify some of the terms used, as they tend to be many and confusing. According to Karlsson [Kar00], the term incremental development can be used for all the following terms; evolutionary delivery, concurrent software engineering, time boxing, spiral model, iterative development, phased development, versioned implementation and short interval scheduling. An overall term for these type of models is lifecycle models. In this project, the term incremental development will be used to cover all these concepts, although the spiral model has been considered separately. This chapter details the general incremental development. The last sections of this chapter will focus on the incremental development models. First, we will look at extreme programming, followed by Rational Unified Process, Dynamic System Development Method, Catalysis, Select and Staged Delivery.

4.1 Incremental development in general

This section aims to clarify the terminology used in incremental development.

4.1.1 Definition

The definitions of incremental development are many, and vary to some degree. The background for the following description is found in [Bal98].

Incremental development is a stepwise development of a product. At the beginning of a project, the requirements are gathered and modelled as far as possible. A chosen part of the requirements is then designed and implemented. After a short development time, this partial product (increment) can be tested by the customer. Based on the product model and the experiences gathered by the customer, the product can be further developed. This process is repeated until the product is finished.

To look at incremental development more closely, it is useful to compare it with the waterfall model (see section 3.1). The upper part of figure 4.1 shows the waterfall model, that begins with the requirements analysis and design part of the project, continues
with the implementation and ends with testing. The lower part of the figure shows examples of two incremental processes, where the development is divided into increments. Each increment can be said to follow the traditional waterfall model, as seen in the figure, in that each increment is developed through the phases requirements analysis, design, implementation and testing. The first process is overlapping, that means that one increment begins before the other is completed, while the other is sequential, i.e. there is no overlap between the increments. This is further discussed in section 4.2.3.

### 4.1.2 Terminology

These two definitions of important terms used in incremental development are found in [Gra92].

**Increment:** a self-contained functional unit of software, together with all supporting material (e.g. requirements specification, design documentation).

**Incremental delivery:** delivery of increments to the customer at intervals throughout the timescale.
4.2 Using Incremental Development

For a given project, incremental development can be carried out in different ways. For example, it can differ with respect to type of increment, the phases that the increments consist of, length of the increments in time and allocation of people to increments. In this subsection, we will take a look at different alternatives for incremental development.

4.2.1 Types of increments

There are different types of increments, depending on what type of functionality is added in each increment. There are different ways to classify these types, this classification is according to [Kar00]:

- Separate user functions are added in different increments. In [Kar00], this is called feature increments.
- Normal/error increments: first simple normal cases are developed, then the more complex cases and the error handling is added in these cases.
- System function increments: Separate system functionality is added in separate increments.

These types of increments can be used in combination with each other. For instance, feature increments can be used on an overall level and within each feature increment, there can be a division of normal/error increments, and within these there can be a further division in system function increments.

4.2.2 Phase span of increments

In the literature, three different types of incremental development are mentioned. They differ at which phases are included in the incremental process. According to [Gre96], the different types vary with respect to the risk of using them:

**Incremental build (and test)** In this variation of incremental development, only the building (implementing) and testing phases of the development are incremental. That is, the incremental development starts in the coding phase (see figure 4.3). Many developers practice this informally, by coding and testing in several steps. This type of development assumes that the requirements analysis and design are already completed.

**Incremental design** Incremental design also assumes complete requirements, but in contrast to incremental build, the design is also developed in increments. (See figure 4.4)

**Incremental requirements analysis** Using this variation, every phase is part of the increment. This means that requirements are not necessarily complete when starting the development. (See figure 4.5)
Figure 4.2: Waterfall model

Figure 4.3: Incremental build

Figure 4.4: Incremental design

Figure 4.5: Incremental requirements
4.2. USING INCREMENTAL DEVELOPMENT

These types of incremental development can be seen in contrast to the waterfall model, where every phase is finished before the next one begins, as seen in figure 4.2.

Types of incremental development are further elaborated in [Kar00], where the concept of splitting and merging is introduced. Splitting is dividing the process in increments, and merging is when incremental development is ended and a whole product is delivered to the next phase. According to [Kar00], splitting and merging can be introduced at any point in the development, not only at those point mentioned earlier. In addition, several splitting and merging points are possible.

With respect to the possible need for changes, incremental requirements analysis is most risky of the three variations. Since the requirements are not complete when starting the design phase, the risk that the design would have to be changed at a later point is high. But this can also be said to be the great advantage of this variation, as it is flexible with respect to changing requirements from the customer. Incremental build, on the other hand, is closer to the waterfall model, and the advantages of incremental development is therefore not fully exploited. As with types of increments, the splitting and merging points have to be chosen for each project, as there is no right answer.

4.2.3 Length and scheduling of increments

[Kar00] mentions three questions concerning length and scheduling of increments:

1. How long should the increments last?
2. Should the increments be of fixed or variable length?
3. Should the increments be scheduled to run in parallel or sequentially?

All these questions are important to answer when planning an incremental development project. There are several alternative answers to these questions:

Length This is of course connected with which phases we let the increments span. But independent of which phases the increment span, the increments can be chosen to be short or long. At each extreme, very short increments can lead the developers to only focus on coding, not having time for documentation. Using very long increments, on the other hand, the project might end up using the waterfall model. In the industry, the increments can vary from hours (Extreme programming, see section 4.4) and days, to weeks and months. According to [Kar00], the most common is increments of three to six weeks if the phases spanned are implementation and testing, four to twenty weeks if all the phases are included.

Fixed or variable If the increments are of fixed length, the schedule is fixed for the whole development process. This way, everyone knows when the increments should be finished and integrated. Alternatively, using variable length can make it easier to divide the increments with respect to functionality, but this requires better managing.

Scheduling As mentioned, the scheduling of increments can be either parallel or sequential. Using sequential scheduling, everyone will be working on one increment. Using
parallel scheduling, the work with different increments overlaps. This requires more coordination than the sequential alternative, but more time can be spent on each increment.

Figure 4.6 shows examples for how a project can be divided in increments. The increments span requirements analysis, design, implementation and test, and the first requirements analysis phase and the last test phase considers the entire system. The first variant has eight increments, which are carried out two and two in sequence, which again are parallel. The increments have variable duration. This variant has short increments and the overlap causes the total length of the project to be shorter. On the other hand, this project requires more planning with respect to how to divide the project in increments and more coordination since it is highly parallel and contain variable length increments. The second variant has three long increments and less overlap, and therefore the total length is somewhat longer. The last variant has four increments and the height of the increments indicates that more people are allocated to each increment than in the first and second variant. Like for the second variant, the increment length is fixed, which makes planning easier. The project only overlaps in the testing phase, which makes feedback from one increment to the next possible. Due to the lack of overlap, this project variant is longer.

4.2.4 Organization of increments

A challenge when using incremental development is how to assign the people to the increments. There are two major ways to do this [Kar00]:

By design item: In this strategy, people are assigned to either a design item or a development phase. The developers have to focus on the functionality of each increment.

By increments: Using this strategy, people are designed to increments. The developers then have to focus on the functionality in each design item that this increment affects.

Which of these two strategies are more appropriate depends on the type of increments. If the increment is of feature type (see section 4.2.1), allocation by increment is more appropriate, to set the focus on the functionality of the increments. If the increment is of normal/error type on the other hand, allocation by design item might be more appropriate, as you then can have the same people developing the normal and error cases for the design item.
Dark grey represents the phases requirements analysis, design and implementation, light grey represents testing.

Figure 4.6: Length and scheduling of increments
4.3 Advantages of incremental development

There are many advantages of incremental development, for both developers and customers, found in the literature [Gra92]:

For developers:
- Better team morale
- Implementation problems can be solved early
- Rework with changing requirements is reduced
- Risks (e.g. of overrunning time schedules) are reduced
- Easier maintenance of developed products
- Productivity is measured

For customers:
- A useful product is received early
- Confidence in the developer is increased
- The software developed is of better quality
- User acceptance is increased
- Users can learn the system gradually as it is developed
- It is easier to understand the requirements
- The real needs for the system are met, not the old needs

Continuing on this theme, the next few sections will detail several different incremental development methodologies. We will take a look at extreme programming, the Rational Unified Process, the Dynamic System Development Method, Catalysis, Select Perspective and Staged Delivery.

4.4 Extreme programming

The methodology to be discussed in this section was proposed by [Bec99]. It is based on a very strict usage of basic software development techniques.

**Continuous planning.** Planning in an extreme programming project is done by the two abstract groups of business and development. Decisions are taken after a dialogue between the two units, where certain dependencies between the two groups have been formalised. Business is responsible for setting priorities, for example, while development will give estimate on the releases. This shares the responsibility of a project between management and developers.
Unit testing. Any feature to be added to the system in an extreme programming project will always have a complete test case written in advance. This leads to every feature being completely tested from its creation, and all the test cases are integrated into a test suite so that complete functionality always can be verified.

Pair programming. A controversial point in this methodology is pair programming. Developers do not have dedicated machines, but write code and test cases together so that one will review the code while the other writes it. These roles are interchangeable, and the pairs are dynamic. Two persons working together in the morning must not necessarily continue together on the tasks they undertake in the afternoon.

Simplicity. The extreme programming methodology aims at the simplest solution possible. This means that any feature that does not add something to the system immediately is left out. This translates into 'designing for today', rather than 'designing for tomorrow'. Simple design also makes refactoring simpler, and supports shared responsibility.

Shared responsibility. Every person on an extreme programming team has equal responsibility for the code. This means that anyone have the right to improve any part of the code provided that it does not break the test suite and that it will improve the system.

Refactoring. When a new feature has been added, and the complete set of test cases for the system is running, extreme programming developers should ask themselves: 'Can the system be made simpler?'. If this is possible without breaking the test suite, this improvement should be done immediately. Modifying and throwing away other people’s code demands courage, but the combined effect of shared ownership and pair programming will make the barrier lower as the other person can provide critical insight and motivation.

As extreme programming is a relatively new methodology, it has not been thoroughly discussed in the literature yet. Experiments conducted with students as developers [MT01, KHH+00] points out that the methodology demands highly skilled programmers, and that the process does not scale well beyond 10 persons. This last limitation is also pointed out by [Bec99]. The strengths of the methodology pointed out by the same studies are that code is clearer as it is continually reviewed in pair programming, in addition to faster development cycles. The experience of the users of extreme programming is that unit testing, with the tests written in advance, is a great advantage.

Due to the scaling problems, many projects will not profit from extreme programming. On smaller projects with changing requirements and tight schedules, this methodology can certainly add value provided that the developers are motivated and experienced.

4.5 RUP - Rational Unified Process

In 1995, a process started that in 1998 resulted in the Rational Unified Process, developed by Rational Software. It is a process framework used by thousands of companies
worldwide, in the areas of telecommunications, transportation, aerospace, defence, manufacturing, financial services and systems integrator [RUP]. According to [Kru99] the Rational Unified Process achieves an improvement in the software development process as it is based on best practices, that is, methods that are proven to be effective through use in the software industry. The six most important best practices are:

- Develop software iteratively
- Manage requirements
- Use component based architectures
- Visually model software
- Verify software quality
- Control changes to software

RUP aims to help the development organization to produce high-quality software that meets the needs of its end users within a predictable schedule and budget. RUP is a process framework that can be used by the organization.

Figure 4.7 shows the architecture of RUP. For each of the phases, inception, elaboration, construction and transition, there are a number of iterations, consisting of planning, business modelling, stating requirements, analysis, design, implementation, test and evaluation. The iterations are carried out until the wanted results for that phase are achieved.
4.5.1 Phases in the Rational Unified Process

The different phases focus on different parts of the process, and after the completion of each phase a new milestone is reached. Figure 4.8 shows graphically the emphasis of each phase in the project.

Inception phase The main objective of this phase is to define project goals, in cooperation with the stakeholders. As shown in figure 4.8, the main emphasis is on business modelling and planning. The conclusion of the inception phase is reaching the Life-Cycle Objective milestone. This milestone includes that the scope definition, cost and schedule estimates are agreed on, and that the requirements represented by use cases, priorities and risks are agreed on.

Elaboration phase As seen in figure 4.8, this phase continues the business modelling, requirements modelling and analysis and design started in the inception phase, in addition to starting the implementation and testing. When the Life-Cycle Architecture milestone is reached at the end of this phase, the most important results are that a use case model of the system is nearly complete, non-functional requirements are established, the software architecture is described and an executable prototype exists.

Construction phase As the name implies, the main focus of this phase is on analysis and design and implementation and testing, but as shown in figure 4.8, management activities like configuration and change management and project management are carried out, in parallel with continued business and requirements modelling. The result of this phase is reaching the Initial Operational Capability milestone, when having completed component development and testing, so that the product is ready
to put in the hand of the users. This means that the product must be integrated on the users platform and the user manuals must be ready.

**Transition phase** This phase mainly consists of doing the required changes that arises after introducing the software to the users. The milestone for this phase is Product Release, when all the required changes are made and the objectives of the project are met, the product is ready for the market.

**Iterations and milestones** Each phase is carried out in the necessary number of iterations to reach the milestone for each phase. Each milestone consists of a number of evaluation criteria for the phase that has to be met before continuing to the next phase.

### 4.5.2 RUP process model

A process model should describe who is doing what, how and when. In RUP, this is done through workers, activities, artifacts and disciplines (called workflows in earlier versions of RUP).

**Workers** A worker defines 'who' in a process. That is, the behaviours and responsibilities of an individual or a team. In RUP, the term worker means the role that defines how an individual should work. Each individual may be a different type of worker at the same time.

**Activities** Activities define the behaviour of a worker, that is 'how'. Each activity has a specific purpose and a relatively short time-span (a few hours to a few days). The worker performs the activity. Each activity is broken into steps, that fall in the following categories: thinking (understanding the activity), performing (doing the activity) and reviewing (inspecting the results of the activity).

**Artifacts** Artifacts are what the worker develops and works with in an activity, that is the responsibilities of the worker, the 'what'. Artifacts are the actual products of a process: models, documents, code etc. The artifacts are divided into sets: management (related to the management of the project), requirement (requirement and business models), design (design and test model, architecture description) implementation (source code and data files) and deployment (installation script, user documentation and training material) set.

**Disciplines** A discipline defines the sequence of activities, that is when. The disciplines in RUP are divided into nine core disciplines, as seen in the vertical dimension of figure 4.8 e.g. the business modelling discipline and the requirements discipline.

One of the main advantages with RUP is that it is based on best practices proven to be successful in the industry. One of the disadvantages is that it is a very general process model, that has to be tailored to meet the needs of each organization. Also, the use of this model requires some reorganization of the traditional project work in software development, for instance with the use of workers and activities. This can result in organizations not taking full advantage of the RUP framework.
4.6. **DSDM - Dynamic System Development Method**

As compared to other lifecycle models, one of RUP’s strengths is that it is a process product. That is, it is a software tool, not just a set of manuals. Also, RUP can be integrated with Rational’s UML product, to offer a more complete software development tool.

4.5.3 **Process framework**

In addition to the workers, activities, artifacts and disciplines, RUP contains guidelines, concepts and tool mentors to assist the software development process. This together makes RUP a process framework for developing software.

4.6 **DSDM - Dynamic System Development Method**

DSDM is a method for iterative development. It was developed by a consortium established in 1994, which goal was to "develop and evolve continuously a public-domain method for rapid application development" [MS95]. The method Rapid Application Development - RAD, occurred in the beginning of the 1990s.

4.6.1 **RAD**

At the beginning of the 1990s, the sequential waterfall model (section 3.1) was seen as insufficient in a fast-moving business environment. RAD was a step away from the waterfall model, describing a process to deliver a fit-for-purpose system faster [Her95].

The main approach in RAD is to use [Mor95]:

- Small teams
- Fourth generation tools
- Fast development
- Tight time-scales and resource constraints
- Iterative, evolutionary and participative prototyping
- User involvement

4.6.2 **RAD and DSDM**

The problem with RAD was that no standards and rules for the method was developed, until the need for this resulted in the Dynamic System Development Method Consortium. Based on the needs that resulted in RAD, DSDM is meant to provide a framework process for developing software faster. The main difference between DSDM and traditional lifecycle models is that traditionally, the requirements of the project has been the fixed part, while time and resources have been variable. DSDM projects aim to have time and schedule as the fixed part, while the requirements should be variable.
4.6.3 DSDM principles

DSDM is based on some underlying principles that must be fulfilled to get a successfully running project:

- Focusing on the business use of the product
- Strong user involvement throughout the project
- Collaboration and cooperation between all stakeholders
- Fulltime involvement of the project team
- User interface is an important part of the project
- Cleanroom is used for the project group
- Time-boxing is used to lock the time used
- Frequent delivery of products
- Iterative process

4.6.4 DSDM lifecycle

The DSDM lifecycle is iterative and incremental. The lifecycle consists of the following phases:

**Feasibility study** Checks if the project is suitable for a DSDM approach

**Business study** Makes the basis for the project: requirements, system architecture

**Functional model iteration** Refining the business aspects of the system

**Design and build iteration** Testing the system

**Implementation** Delivering the product to the customer

DSDM also contains recommendations in the following fields:

- Project management
- Personnel
- Tools and techniques
- Quality assurance
- Software procurement
There are several advantages to DSDM. Most importantly, it creates a formal definition of RAD. Also, it focuses on testing as integrated in the development process. In comparison with the waterfall model, DSDM is faster, as experience shows that project deliveries are more often delivered on time [DSD]. On the other hand, DSDM is very project management oriented. It lacks the integration with a development tool that for instance RUP has through UML. Some critics also claims that the short timeboxes and the lack of a complete set of functional requirements can be a threat to the quality of the software.

4.7 Catalysis

Catalysis [CAT] is a process model for object oriented and component based design. It is based on UML, and it has been evolving since 1992. The process is open and non-proprietary, and it is has been used by several Fortune 500 companies.

The principles of catalysis is based on three axes: the levels of description, the modelling constructs and the underlying principles. These three axes have again been split into three elements. These sub elements will be described in the following sections.

4.7.1 Levels of description

Catalysis operates with three levels of description. These cover different granularities of system design, and provides a method of decomposing a problem into smaller parts. The levels are as follows.

Domain/Business This level identifies the problem at hand. Any component in the system will have a context, and the domain model developed at this level gives a clear definition of this context.

Component specification At this level, the problems solution is specified. To reach this goal, a detailed description of a components external behavior is given.

Internal design Here, an implementation of the superior level is given. This will be a description of a component’s internal design using UML.

These levels of description allows a software component to be completely defined at different levels. A component can be reused, so that domain/business descriptions can compose top-level designs without thought of the component’s underlying structure.

4.7.2 Modelling constructs

Catalysis offers three different constructs for modelling. These specify groups of components, how they cooperate and how to flesh out a description.

Type A black-box description of component. This will only specify its external behaviour while completely ignoring its implementation.
Collaboration  Specifies the behaviour of a group of components. A collaboration is an architectural package that contains several interconnected types.

Refinement  A relation between two descriptions of the same phenomenon at two different levels of abstraction. If we have two models where one model is at a finer level of granularity than the other, the one at the finest level will be a refinement of the coarser one.

Using these three constructs, the designer can give detailed descriptions of inter-component interaction in addition to specifications of each separate component. Formalising the refinement process is also an important strength of the Catalysis method.

4.7.3 Underlying principles

In addition to the levels of description and the modelling constructs, Catalysis has three underlying ‘philosophical’ principles. These are meant as an aid to the designer in how one should think. The three principles are:

Abstraction  Focus on the essential aspects of the problem. Any unnecessary details should be taken out of the overlying design.

Precision  Find inconsistencies early and provide clear abstract models. Avoid unclear definitions in the high-level models.

Pluggable parts  Always strive for reuse. A system should be constructed from existing parts if possible. If an existing design provides wanted functionality, disassemble its design until a useful component can be obtained.

4.8 Select Perspective

Select Perspective is a process-modelling tool that came from Select Software Tools, but is now a part of Aonix. Select is a set of tools to support software development. The underlying process is Select Perspective, which is a "collection of industry best-practice modelling techniques that are applied and adapted using process templates within an architectural framework across a wide range of developments in a component-based setting". In summary, Select Perspective is [Han01]:

- A process
- A component-based architecture
- Guidelines for carrying out projects

Select is based on Rumbaugh’s OMT (Object Modelling Technique), supplemented with use cases, component-based architecture and UML. Since the process is based on making systems consisting of several components, the overall process is divided into two processes, the Solution process and the Component process, which will be described in sections 4.8.1 and 4.8.2.

The Select process should help:
• Identify and divide the work to be done
• Follow-up of the development
• Allocate work appropriately
• Find the need for physical resources
• Estimate time and cost
• Identify the necessary conditions for starting a new activity
• Find products and results from each activity
• Decide what techniques should be used
• Draw experiences from earlier work

The Select Perspective is based on [AF98]:

• RAD
• Iteration
• Focus on products
• Incremental development
• Evolutionary development
• Constant validation
• Positive politics
• Process refinement

4.8.1 Solution process

The aim of this process is to develop a solution that can be used after short time.

Figure 4.9 shows the solution process. The purpose of the steps in the process are:

Feasibility Find possible solutions (preferably through reuse) and plan the project.

Analysis As far as possible, find requirements to use a basis for an incremental delivery plan.

Prototype Get the user requirements by help of screenshots. Iterate with analysis.

Plan increment Plan production of services by incremental delivery.

Design and build Construct, build and test an increment. Reuse.
Figure 4.9: The solution process.

**User acceptance** Get acceptance of each increment from the users. Iterate with design and build.

**Roll out** Install an increment in the user environment.

As seen, this process focuses on the general solutions of the system, and how to develop incrementally.

### 4.8.2 Component process

The component process aims to create components that can be common for several systems, and that can be used by a third party.

Figure 4.10 shows the component process. The purpose of the steps are:

**Assessment** Evaluate the need for reusable components.

**Plan services** Make a plan for the use of components and increments.

**Design and build** Construct, build and test components.

**Acceptance** Get accept before components are installed.

**Roll out** Install a set of components.

Each of these steps focus on how to take advantage of the use of components in the project.
4.9 Staged Delivery

Staged Delivery is an incremental development model described in [McC98]. In Staged Delivery, increments are called stages. It aims to force the development teams to bring software into releasable states faster, through developing and delivering the product in stages. An outline of the Staged Delivery process can be seen in figure 4.11.

According to [McC98], the key to software project success is planning. The Staged Delivery model is based on several plans: software development plan, project estimates, quality assurance plan and staged delivery plan. The software development plan is the overall planning of project, creating the guidelines for the project. The staged delivery plan on the other hand, is a detailed planning of each stage in the project. This will be described in the next subsection.

4.9.1 Staged delivery plan

The stage plan contains milestones, schedules and task assignments for the following activities:

- Requirements updates
- Detailed design
- Construction
- Test case creation
- User documentation updates
- Technical reviews
Figure 4.11: Staged Delivery.

- Defect corrections
- Technical coordination
- Risk management
- Project tracking
- Integration and release
- End-of-stage wrap-up

Each of these activities is performed at each stage, at the detail level adequate for that stage in the development.

4.9.2 Aspects of Staged Delivery

There are several advantages to using Staged Delivery:

- Most important functionality is delivered first
- Planning and risk management is emphasized
- Problems become evident early in the project
- Less time is spent on making progress reports
- The possibility of estimation errors is reduced
### Table 4.1: A comparison of software lifecycle models.

<table>
<thead>
<tr>
<th>Model</th>
<th>Incremental</th>
<th>Phase testing</th>
<th>User interaction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Waterfall</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>V-model</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Prototyping</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Spiral model</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Cleanroom</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Extreme programming</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>RUP</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>DSDM</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Catalysis</td>
<td>Yes</td>
<td>Not specified</td>
<td>Not specified</td>
</tr>
<tr>
<td>Select</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Staged Delivery</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

- The time between stages can be used to make changes to the software

There are also some disadvantages worth mentioning:

- There is an increased project overhead because of the many releases
- Much time is spent on planning releases
- Parts of the software is repeatedly tested in the different stages.

The last point can be seen as an advantage, as through repeated testing throughout the project, much testing is avoided at the end of the project.

## 4.10 Summary of software lifecycle models

Table 4.1 recapitulates some properties of the different models discussed in chapter 3 and chapter 4. We have looked on the amount of testing, whether the model is incremental and if the user is involved in the development process. The values in the table should be self explanatory.
Chapter 5

Investigation of incremental development

The main part of this project was to be a survey of the use of incremental development in Norwegian IT industry. The project started out with a literature study to provide the background of the survey. The survey was carried out in two stages, first with a round of interviews, thereafter a questionnaire distributed to a larger set of companies. In this chapter, we will first look at the method of the project, thereafter we will give the results of the study before discussing these results in the last section.

5.1 Method

The project consisted of several phases, and this section will describe the process we followed while working. Our work went through the following phases:

1. Literature study
2. Prestudy design
3. Interpreting and analysing the pre-study results
4. Main study design
5. Main study interpretation and analysis

The literature study provided us with a general background in the field, and gave us enough material to make and analyse the pre-study. Based on the lessons learned in the pre-study we were more confident when we shaped and interpreted the main study. Our main motivation for using this method was to optimise our learning curve, and to build our knowledge gradually.

5.1.1 Literature study

At the start of the project, we read up on available material on incremental development. We used several different sources to compile an image of the state of the art of incremental software development.
The literature study raised questions about how widespread incremental development was. We decided to design some questions for software development businesses to learn more about the subject.

5.1.2 Prestudy design

As a result of the literature study, we decided to perform a round of structured interviews. We wanted to pose a set of questions to several software development companies as a qualitative study. The questions were formed after the hypothesis that 'Incremental development is used, but in an ad-hoc manner, in Norwegian IT industry.' This hypothesis is partly based on [Lik00, Kar00], where a survey was carried out that asked nine companies about the use of incremental development.

The goal was to interview eight companies, but time constraints only allowed five. The interview’s main intention was to get an overview of how many companies used incremental development, how they use the term incremental development and for what kind of projects incremental development is used. The length of the interviews was decided to be one and a half hour. We wanted to interview project managers, as they would have the overview over the different projects in the company, in addition to the way they are carried out. All the companies were given the questions in advance. It was also attempted to get a variation of companies, in the fields of consulting, electronics and software development in larger organizations. See appendix A for the list of questions that were used in this phase of the project.

5.1.3 Prestudy: method for interpreting and analysing the results

The results of the interviews were used to refine the hypothesis given above, before moving on to the next step in the project. When we evaluated the results of the interviews, we realized that our hypothesis had been poorly phrased. We concluded that it was impossible to prove or refute the hypothesis used in this part of the study, and based on the lessons learned from this error, we were more stringent when formulating the new hypotheses.

During the course of the interviews, we realized that a quantitative study of the same domain could provide interesting insights. This lead us to the conclusion that a larger sample was needed, and that attempting to interview a significantly larger number of companies would be infeasible. The next logical step was to perform a study with a questionnaire that could reach a larger number of companies.

5.1.4 Main study design

Based on the results from the round of structured interviews, we developed two new hypotheses that refined the ideas from the first part of the project. Our impression was that most of the companies that develop software use some form of incremental development, and we wanted to state formally that this was an advantage. In addition, we wanted to see if the companies felt that incremental development was more demanding on the customers and the personnel.
These hypotheses were used as a basis for a first questionnaire. This questionnaire was evaluated and criticized by professors at NTNU’s and UiO’s institutes for computer science. This criticism uncovered weaknesses in both the formulation of the hypotheses and the form and content of the questions themselves. Based on this criticism, we formulated two new hypotheses, and proceeded to design a new questionnaire. The refined hypotheses are as follows.

\( \mathcal{H}_1 \) There are advantages to using incremental development.

\( \mathcal{H}_2 \) Incremental development requires more/other customer/developer competence than non-incremental development.

The resulting null hypotheses to be rejected are as follows:

\( \mathcal{H}_{10} \) There are no advantages to using incremental development.

\( \mathcal{H}_{20} \) Incremental development does not require more/other customer/developer competence than non-incremental development.

Drawing on the experiences gathered by [Lik00, Dyb01], we decided that the use of a rating scale was the most appropriate approach to making the questionnaire. Full sentence answers would have been too hard to synthesize, so the problem was to find what kind of rating scale to use. [Dyb01] concludes that if too few scaling points are used, the answers will be inaccurate, and too many scaling points make it difficult for the respondent to make a discrimination between the points. This leads to the conclusion that a five-point scale yields the most reliable result. This was used in this survey, where the respondents were asked if they

1 totally disagree
2 partly disagree
3 indifferent
4 partly agree
5 totally agree

with a statement. As the main intent of the survey was to study incremental development, the introductory question was if the company used incremental development or not. Six questions were designed for companies who did not use incremental development, asking for reasons why the company did not use it. The remaining questions were for companies who used incremental development. The results of the interview were considered when making the questions. From the interviews, section 5.2.1, it was known that incremental development is in use in many Norwegian companies. Therefore, the focus of the questionnaire was on how incremental development is used, not why it is not used. From the interviews, important aspects of incremental development were also known:

• Close cooperation with the customer
• Reduction of project risks
• Requires management discipline
• Requires some form of training or mentoring
• A process product is not used in all companies
• Length of increments and projects vary
• Project group size vary

Based on this, several statements were made to test the hypotheses. To test $H1$ the respondents were asked to evaluate the following statements:

• A strength with incremental development is that the project group has more control over the requirements.
• It is an advantage that incremental development encourages close communication with the customer.
• Staged delivery of the product is important for the customers satisfaction with the result of incremental development projects.
• Incremental development leads to a better internal prioritising of tasks.
• Reduction of risks is a central attribute of incremental development projects.

And to test $H2$ these statements were presented to the respondents:

• Incremental development projects requires more skills from the customer than non-incremental development projects.
• Incremental development projects requires more skills from the developers than non-incremental development projects.
• Incremental development projects requires more skills from the management than non-incremental development projects.
• Formal education of developers is central in carrying out a successful incremental development project.
• Informal developer support (e.g. mentors and "coffee machines") are crucial in carrying out a successful incremental development project.
• The project manager’s role is more important in a incremental development project than in a non-incremental development-project.
In addition, an intention of the questionnaire was to get an overview over how incremental development is used in the software industry. Questions about length of increments, sequential or overlapping increments and project group sizes were therefore also included. To get an overview of what kind of companies and employees were answering the questionnaire, a set of general questions about the company and employees was included at the end. See appendix B for the complete questionnaire.

The questionnaire was distributed by e-mail to a number of Norwegian software development companies. The e-mail addresses were found through several sources. 29 e-mail addresses were found from an IDI contact list, 9 e-mail addresses came through the PROFIT participant list and 600 were part of IKT Norge’s e-mail list. It was requested that the questionnaire was answered by a project manager, method responsible, quality manager or similar. The companies were given one and a half week to answer the questionnaire.

5.1.5 Main study: method for interpretation and analysis

The last part of the project was related to treating and analysing the data gathered in the survey. We attempted to create a quantitative study, but due to reasons that will be discussed in section 5.3.2, this proved to be harder than foreseen.

- Of the 9 companies on the PROFIT list, 3 answered. On the list we received from IDI, we successfully contacted 29 companies. Out of these 29, 3 responded that they did not want to participate in the study, while 14 filled out the questionnaire. This gave us a respectable response rate between 30% and close to 50% on these two channels. It is worth noting that 4 of the companies answering the questionnaire also participated in the interview round.

- The IKT Norge list, on the other hand, proved to be a fiasco. Sources responsible for the administration of the list claimed that it contained approximately 600 companies. Out of these 600, 4 answered, leading to a rather morose response rate of less than 1%. As a result of this, we have excluded these four responses from the final presentation of the results, although the raw data is presented in appendix C.

When analysing the results of the questionnaire, we discovered that our rating scale was too finely grained. Each of the five ratings had an average of \( \frac{24}{5} \approx 4 \) answers, so generalizing on the data proved to be difficult. When treating and presenting the results, we collapsed the scale into 3 points. The values 1 and 2 (totally disagree and partly disagree) were merged into a value called disagree, while the values 4 and 5 (partly agree and totally agree) were merged into a value called agree. We left the indifferent value untouched.

The data obtained from the questionnaire were treated and used to draw conclusion on the state of incremental development in Norwegian IT-industry today. As the number of responses was relatively low, we have not drawn any quantitative conclusions as was the original goal. Rather, we have used the data to produce a qualitative conclusion.
5.2 Results

This section will give the results of the two parts of the survey, the interviews and the questionnaire.

5.2.1 Prestudy results

<table>
<thead>
<tr>
<th>PQ#</th>
<th>Company1</th>
<th>Company2</th>
<th>Company3</th>
<th>Company4</th>
<th>Company5</th>
</tr>
</thead>
<tbody>
<tr>
<td>PQ1</td>
<td>#Employees</td>
<td>90</td>
<td>50</td>
<td>25</td>
<td>35</td>
</tr>
<tr>
<td>PQ2</td>
<td>#IT-emp.</td>
<td>45</td>
<td>40</td>
<td>24</td>
<td>34</td>
</tr>
<tr>
<td>PQ3</td>
<td>Segment</td>
<td>BFI</td>
<td>BFI</td>
<td>BFI</td>
<td>ISW</td>
</tr>
<tr>
<td>PQ9</td>
<td>Devel. model</td>
<td>IAD/RUP</td>
<td>RUP</td>
<td>Own</td>
<td>Staged Delivery</td>
</tr>
<tr>
<td>PQ10</td>
<td>Formalized Process</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>PQ11</td>
<td>Experience</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
</tr>
<tr>
<td>PQ12</td>
<td>How Long</td>
<td>5-6 years</td>
<td>3 years</td>
<td>3-4 years</td>
<td>4 years</td>
</tr>
<tr>
<td>PQ13</td>
<td>Old Model</td>
<td>Waterfall</td>
<td>Waterfall</td>
<td>Waterfall + inc.del</td>
<td>N/A</td>
</tr>
<tr>
<td>PQ15</td>
<td>Motivation</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
</tr>
<tr>
<td>PQ16</td>
<td>Training</td>
<td>Internal Formal</td>
<td>External Formal</td>
<td>Internal Informal</td>
<td>Internal Informal</td>
</tr>
<tr>
<td>PQ17</td>
<td>Length tot/ increment</td>
<td>4-12 months/ 2-4 weeks</td>
<td>3-12 months/ 3-8 weeks</td>
<td>6-12 months/ 4 weeks</td>
<td>3-6 months/ 2-3 weeks</td>
</tr>
<tr>
<td>PQ18</td>
<td>Project group size</td>
<td>2-20</td>
<td>2-12</td>
<td>4</td>
<td>5-6</td>
</tr>
</tbody>
</table>

Table 5.1: Summary of interview responses.

BFI - Bank/Finance/Insurance. ISW - Internet/Software development.

Table 5.1 gives a summary of the different companies’ responses to the interview. The numbering of the questions refer to the list of questions in appendix A. This is an extremely concentrated version of the data gathered during the interviews, and should be taken as a roadmap to the results gleaned from the pre-study. Some points of interest will be enumerated below.

All the companies used incremental development, although the phraseology and lifecycle model varied. Two companies used the Rational Unified Process, one used Staged Delivery combined with elements from Extreme Programming, one used a proprietary process based on RAD called Iterative Application Delivery (IAD), and the last used other internally developed project process models. Of the five, the ones using RUP and IAD had formalized their processes. The tools used were quite diverse, but the companies using RUP naturally depended on Rational products.
Most of the companies interviewed were quite young, and two had been using incremental development since they started 3 to 4 years ago. The three remaining had earlier experience with other methods, but introduced incremental development 2 to 6 years ago. The models used before this change varied, but most had been using versions of the waterfall method, albeit with slight modifications. One company in particular had been using incremental delivery in combination with the waterfall method for 10 to 15 years before incremental development was introduced.

Incremental development had normally been introduced in the companies by technical people. Often though, the managers responsible for the decision had technical backgrounds. An exception to this was a large multinational consulting firm where the central quality office introduced incremental development. All the interviewees had personnel that were motivated for using incremental development.

The training related to incremental development varied greatly. One of the interviewed companies was a Rational partner, with certified RUP personnel, and the users of IAD had internal training in their method. The other companies had less formal education, but everyone had a mentor policy. The responses whether incremental development demanded more competent technical personnel varied from those that meant yes definitely, via those who meant no but they have to be more disciplined, to those who meant no, but management has to be stricter.

Incremental development projects usually lasted between 3 and 12 months, with increments ranging from 2 to 8 weeks. The project groups were on average around 5-6 people, but the different companies had experience with groups ranging from 2 to 20 persons. Groups becoming larger than this were normally split into smaller groups.

The companies used incremental development for all their larger projects, although very short projects spanning over just a few weeks were normally managed as waterfalls.

**Interesting quotes from the interviewees.**

During the course of the interviews, some interesting quotations were noted down. Some of them have been gathered here, and they serve as a pointer to what parts of the industry mean about incremental development.

- Internet Software developer, 35 employees: *The only way of doing software development is by using incremental development.*

- Internet Software developer, 35 employees: *We have always been using incremental development with extreme testing.*

- Consulting firm, 90 employees: *Having educated customers is equally important to having competent developers when using incremental development.*

- Consulting firm, 50 employees: *Incremental development’s most valuable property is risk reduction.*

- Consulting firm, 50 employees: *The biggest challenge when introducing incremental development is convincing customers of its added value.*
<table>
<thead>
<tr>
<th>Q#</th>
<th>Disagree</th>
<th>Agree</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q1</td>
<td>Never considered</td>
<td>1</td>
</tr>
<tr>
<td>Q2</td>
<td>Projects too short</td>
<td>2</td>
</tr>
<tr>
<td>Q3</td>
<td>Too expensive</td>
<td>3</td>
</tr>
<tr>
<td>Q4</td>
<td>Demands re-œd.</td>
<td>3</td>
</tr>
<tr>
<td>Q5</td>
<td>Company too small</td>
<td>3</td>
</tr>
<tr>
<td>Q6</td>
<td>Customer lacks understanding</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 5.2: Answers to questions 1 through 6 from appendix B.

- Consulting firm, 100 employees: *Incremental development demands high levels of cross-disciplinary competence.*

- Consulting firm, 25 employees: *A great advantage of incremental development is that the customer can see the product evolve according to the specification.*

- Consulting firm, 25 employees: *We have no intentions of introducing RUP as the process demands far too much involvement.*

These statements are subjective, and should be taken as such.

### 5.2.2 Main study results

Due to the low response rate on the questionnaire, the results will not be statistically valid. This section will instead give a summary of the results. Tables detailing each companies’ answers are found in appendix C, while the numbering refers to the questions in appendix B.

**Questions 1 to 17** Tables 5.2 and 5.3 summarises the answers to the first 17 questions in the survey. Questions 1 through 6 are only relevant to businesses not using incremental development. Of the 17 valid respondents to the survey, three answered these questions. As none of the subjects were indifferent to any of the statements put forth, this column has been removed from table 5.2. Table 5.3 summarizes the results of questions 7 through 17. These questions were answered by the 14 subjects that used incremental development.

In the two tables, the numbers signify how many respondents agreed, were indifferent or disagreed to a statement. As mentioned in 5.1.5, the rating scale used in the survey had 5 points, but the rating scale of the results has been collapsed into three points. These are named disagree, indifferent and agree, respectively.

**Questions 18 to 24** These questions were also answered by the 14 companies using incremental development. The questions asked for a short answer or number.

- Question 18: 10 of the companies used RUP, either completely or partly. One used it in combination with XP and Staged Delivery. The remaining 3 companies used varying methods.
<table>
<thead>
<tr>
<th>Q#</th>
<th>Description</th>
<th>Disagree</th>
<th>Indifferent</th>
<th>Agree</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q7</td>
<td>Better requirement control</td>
<td>1</td>
<td>1</td>
<td>12</td>
</tr>
<tr>
<td>Q8</td>
<td>Better communication</td>
<td>2</td>
<td>1</td>
<td>11</td>
</tr>
<tr>
<td>Q9</td>
<td>Staged delivery is important</td>
<td>2</td>
<td>0</td>
<td>12</td>
</tr>
<tr>
<td>Q10</td>
<td>Better prioritizing of tasks</td>
<td>0</td>
<td>3</td>
<td>11</td>
</tr>
<tr>
<td>Q11</td>
<td>Reduced risk</td>
<td>2</td>
<td>1</td>
<td>11</td>
</tr>
<tr>
<td>Q12</td>
<td>More customer skill</td>
<td>8</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>Q13</td>
<td>More developer skill</td>
<td>4</td>
<td>2</td>
<td>8</td>
</tr>
<tr>
<td>Q14</td>
<td>More management skill</td>
<td>2</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>Q15</td>
<td>Formal education</td>
<td>5</td>
<td>2</td>
<td>7</td>
</tr>
<tr>
<td>Q16</td>
<td>Informal support</td>
<td>4</td>
<td>3</td>
<td>7</td>
</tr>
<tr>
<td>Q17</td>
<td>Mgr more important</td>
<td>5</td>
<td>2</td>
<td>7</td>
</tr>
</tbody>
</table>

Table 5.3: Answers to questions 7 through 17 from appendix B.
5.2. RESULTS

- Question 19: 4 companies used both sequential and overlapping increments. 7 used sequential and the remaining 3 used overlapping increments.
- Question 20: 6 companies answered that the total length of a project was less than or equal to 5 months, 4 answered more than 5 months, one answered unknown and the rest had varying project lengths.
- Question 21: 7 companies answered that the total length of their increments was less than or equal to 4 weeks. Of these, one company sometimes had increments down to hours and days. 5 had increments of more than 4 weeks, the longest one being 16 weeks. One answered unknown. The last company was not able to give an average size, as the increment lengths varied too much.
- Question 22: 4 companies had project groups with less than or equal to 5 members, 5 used project groups with between 5 and 10 members and the rest had project groups with 10 or more members.
- Question 23: 10 companies had project sizes of less than 10 man years, 2 had 10 or more man years, one answered unknown and one varying.
- Question 24: Most of the companies gave advice about the use of incremental development. These are reproduced in the next section.

Questions 25 to 32 The general questions about the companies were answered by 16 of the 17 companies answering the questionnaire. The characteristics of the companies and respondents are shown below, as a summary of table C.4 in appendix C.

From table 5.4, it is clear that a majority of the companies in the survey had less than one hundred employees. No market segment in particular dominated the survey, although Electronics / Telecommunications distinguish itself somewhat. Twelve companies worked for external customers, mostly operating as consulting firms.

Table 5.5 shows that most of the respondents were at managerial level in the company, as for instance project or quality managers. Also, a majority held a Master of Science degree or equivalent. The average years of software experience of the respondents was around eleven.

Interesting quotes from the respondents

We have compiled some of the responses to question 24 in this section. They have been translated from Norwegian, and they represent some of the most interesting qualitative results that surfaced during the main study. The remaining answers can be found in appendix C.

- Consulting firm, 100 employees: RUP can be used without all the project members having experience, but:
  - Some central members need to have experience
  - As many as possible need knowledge, through courses etc.
  - One RUP-mentor should be working full-time on the project
<table>
<thead>
<tr>
<th>Characteristics</th>
<th>Frequency</th>
<th>Percent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q25: Number of employees</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Less than or equal to 100</td>
<td>6</td>
<td>37,5%</td>
</tr>
<tr>
<td>Between 100 and 500</td>
<td>5</td>
<td>31,25%</td>
</tr>
<tr>
<td>More than or equal to 500</td>
<td>5</td>
<td>31,25%</td>
</tr>
<tr>
<td>Q26: Number of SW developers</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Less than or equal to 30</td>
<td>3</td>
<td>18,75%</td>
</tr>
<tr>
<td>Between 30 and 200</td>
<td>5</td>
<td>31,25%</td>
</tr>
<tr>
<td>More than or equal to 200</td>
<td>7</td>
<td>43,75%</td>
</tr>
<tr>
<td>Unknown</td>
<td>1</td>
<td>6,25%</td>
</tr>
<tr>
<td>Q27: Primary market segment</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Public sector</td>
<td>0</td>
<td>0%</td>
</tr>
<tr>
<td>Bank / Finance / Insurance</td>
<td>3</td>
<td>18,75%</td>
</tr>
<tr>
<td>Industry</td>
<td>2</td>
<td>12,5%</td>
</tr>
<tr>
<td>Electronics/Telecommunications</td>
<td>6</td>
<td>37,5%</td>
</tr>
<tr>
<td>All</td>
<td>1</td>
<td>6,25%</td>
</tr>
<tr>
<td>Other</td>
<td>4</td>
<td>25%</td>
</tr>
<tr>
<td>Q28: Role of SW development projects</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Development of Off-the-shelf products</td>
<td>2</td>
<td>12,5%</td>
</tr>
<tr>
<td>Development for external customers</td>
<td>12</td>
<td>75%</td>
</tr>
<tr>
<td>Development for internal customers</td>
<td>2</td>
<td>12,5%</td>
</tr>
</tbody>
</table>

Table 5.4: Summary of answers to questions 25 through 28.

<table>
<thead>
<tr>
<th>Characteristics</th>
<th>Frequency</th>
<th>Percent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q29: Job title</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Project manager</td>
<td>2</td>
<td>12,5%</td>
</tr>
<tr>
<td>Quality manager</td>
<td>3</td>
<td>18,75%</td>
</tr>
<tr>
<td>General manager</td>
<td>4</td>
<td>25%</td>
</tr>
<tr>
<td>Consultant</td>
<td>1</td>
<td>6,25%</td>
</tr>
<tr>
<td>Other</td>
<td>3</td>
<td>18,75%</td>
</tr>
<tr>
<td>Unknown</td>
<td>3</td>
<td>18,75%</td>
</tr>
<tr>
<td>Q30: Highest education</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Engineer/bachelor</td>
<td>3</td>
<td>18,75%</td>
</tr>
<tr>
<td>MsC or equivalent</td>
<td>10</td>
<td>62,5%</td>
</tr>
<tr>
<td>PhD</td>
<td>3</td>
<td>18,75%</td>
</tr>
<tr>
<td>Q31: Average years of SW dev. experience</td>
<td>11,1</td>
<td></td>
</tr>
<tr>
<td>Q32: Average years worked in company</td>
<td>6,6</td>
<td></td>
</tr>
</tbody>
</table>

Table 5.5: Summary of answers to questions 29 through 32
The project leader must be fully devoted to this method.

- Consulting firm, 300 employees: Use your head. Much of this (incremental development) is no magic, just regular management theory.
- Consulting firm, 50 employees: Introduce a new process gradually. Use mentors.
- Telecommunication company, 1000 employees: Take into consideration that it demands more detailed planning than a waterfall project.
- Telecommunication company, 1000 employees: Requirement management becomes more complex - a requirement database is recommended.
- System development company, 30 employees: Make a light-weight method with the methods suitable for your company, instead of forcing the company into a pattern like RUP.
- Financial system development company, 15 employees: Extreme programming uses extremely short increments of maximum 10 hours. This way of planning can be recommended, as the process is easily controllable and gives good communication and fast feedback.
- Internet software company, 80 employees: Spend time explaining the customer his role, with a focus on design and test.
- Internet software company, 75 employees: Find out what the problems of the software development process are and use elements of RUP to solve these problems.
- Consulting firm, 700 employees: You need skilful, experienced people on analysis, design and implementation.
- Consulting firm, 18 employees: Take extra care to find the customer’s need.

5.3 Discussion

5.3.1 A discussion on the results of the prestudy

As can be seen in section 5.2.1, the different interview subjects had different interpretations of incremental development. The consensus was that incremental development consists of gradually building and delivering a product to the customer. This means that the customer can watch the product take form, and it allows criticism underway. Each increment could be a result of one or more iterations of for example the spiral model. One company reversed this terminology, and called the iterations of these models increments, but the end result was the same.

The general experiences regarding incremental development were good. Important aspects pointed out were that the customers could see the gradual development of the products, and that the models reduced the risk in the projects. Incremental development did demand mature customers though, and the customers have to be brought into the
process for it to succeed. It was also pointed out that the incremental development methodologies are more demanding for the management. One company thought that incremental development was the only way of doing software development.

Based on the results from the interviews, one can be tempted to call incremental development a best practice. All the companies in the sample agreed that incremental development is useful, and it is interesting to note that the introduction of the process was usually bottom up. It is also surprising to see that different flavours of incremental development have been in use for a relatively long time.

From table 5.1, we note that all the companies claimed that their personnel were motivated for using incremental development. This might be because of the notion of best practice mentioned above, and also because the process have been introduced by the personnel themselves. This might be an important factor for motivation, as the employees might want to 'prove' the value of their contribution to the company. In any case, employee satisfaction should be a crucial point in any organization, and incremental development may be a factor to evaluate when building morale.

Another interesting point that surfaced during the interviews was the length of the increments. It was surprising to see the average length of the increments estimated to around 4 weeks, indicating that the projects were very flexible and allowed adjustment of objectives during the course of the development.

**Summary of prestudy results**

The discussion of the interviews can be summarised as follows:

- The interviewed companies had good experiences with using incremental development
- Company personnel were motivated for using incremental development
- Incremental development can be called a best practice in the software industry.
- Average length of increments was about 4 weeks.

## 5.3.2 A discussion on the results of the main study

**Questionnaire response rate**

As mentioned in section 5.1.5, the response rate was very low. We received 20 valid answers, 17 of which answered the questionnaire, and 3 who answered that they were not able to participate in the survey. Four of the 20 answers came from different divisions of the same company. As these were different, we chose to include them as different elements. Of the companies that did not want to participate, one said that they did not know the meaning of 'incremental'. Another company was a developer of integrated circuits, and did not feel that the questionnaire was relevant. The third company chose not to participate of unknown reasons. There are several possible explanations to the low response rate.

Due to a misunderstanding, the questionnaire to the IKT Norge's e-mail list was distributed one and a half week later than the others, giving the companies little time
to fill out the questionnaire. This was crucial for the survey, as this could have been our main source of answers. Additionally, the original list of IDI contacts contained more than one hundred e-mail addresses, but as many of these were old, more than seventy of them were useless.

We did receive four responses from companies on the IKT-Norge list, but we have chosen not to analyze them as they are not a representative sample of the population. Their responses have been included in the summaries of the results in appendix C, and will not be commented further.

Another reason for the low response rate could be that the people at manager level are generally very busy, and they were the people we were trying to reach. Lastly, the term incremental development might not be as widely known as we thought, i.e. companies might use incremental development without using the term. This could have resulted in people not knowing what we asked for and therefore not answering the questionnaire. We received complaints about the lack of a definition of incremental development, but at the time of making the questionnaire we felt this was unnecessary. We received most answers from companies using incremental development. The simple answer to this might be that companies not using incremental development also have not heard of it. This is an evident weakness of the distribution of the questionnaire.

At the end of the project, we tried to get a better response rate through calling some of the companies that had not answered. A calling round to 20 companies resulted in 3 more answers. The rest were either unavailable or not interested in participating in the survey. The reasons given for not participating in the survey were that they did not have time, were out traveling or not interested.

**Answers from companies not using incremental development**

As seen in table 5.2, only three companies who answered the questionnaire did not use incremental development. This is such a small data set that there is not much to say about it, except noting that the three companies did not agree that the reasons for not using incremental development was that it required too many resources, too much re-education or that the company was too small.

The following sections will delve deeper into the material, starting with an informal data-analysis and continuing with hypothesis testing. We finish with a summary of the main study results.

**Answers to questions 18 to 23**

This section looks at the answers to the general questions asked about how the companies use incremental development. An interesting question was how long increments the companies used. According to [Kar00], increments from three to six weeks spanning the phases implementation to test and increments from four to 20 weeks spanning analysis to test are the most common. This is in accordance with the results in our study, where 12 of the 14 respondents used increments of less than 16 weeks. Of these, 7 companies used increments of less than or equal to 4 weeks. Which phases were included in the increments was not a question in the questionnaire.
<table>
<thead>
<tr>
<th>Company</th>
<th>Number of employees</th>
<th>Market segment</th>
<th>Model</th>
<th>Type of increment</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1000</td>
<td>ET</td>
<td>RUP and own</td>
<td>Sequential</td>
</tr>
<tr>
<td>2</td>
<td>1000</td>
<td>ET</td>
<td>RUP</td>
<td>Sequential</td>
</tr>
<tr>
<td>3</td>
<td>1000</td>
<td>ET</td>
<td>RUP</td>
<td>Sequential</td>
</tr>
<tr>
<td>4</td>
<td>700</td>
<td>I</td>
<td>RUP</td>
<td>Overlapping</td>
</tr>
<tr>
<td>5</td>
<td>450</td>
<td>I</td>
<td>Own model</td>
<td>Both</td>
</tr>
<tr>
<td>6</td>
<td>350</td>
<td>BFF</td>
<td>RUP</td>
<td>Sequential</td>
</tr>
<tr>
<td>7</td>
<td>300</td>
<td>All</td>
<td>Own model</td>
<td>Both</td>
</tr>
<tr>
<td>8</td>
<td>175</td>
<td>IE</td>
<td>Staged Delivery, RUP and XP</td>
<td>Both</td>
</tr>
<tr>
<td>9</td>
<td>100</td>
<td>BFF</td>
<td>RUP</td>
<td>Sequential</td>
</tr>
<tr>
<td>10</td>
<td>80</td>
<td>ET</td>
<td>XP</td>
<td>Sequential</td>
</tr>
<tr>
<td>11</td>
<td>50</td>
<td>ET</td>
<td>RUP</td>
<td>Overlapping</td>
</tr>
<tr>
<td>12</td>
<td>20</td>
<td>Other</td>
<td>Unknown</td>
<td>Varying</td>
</tr>
<tr>
<td>13</td>
<td>18</td>
<td>BFF, ET</td>
<td>DSDM</td>
<td>Both</td>
</tr>
</tbody>
</table>

Table 5.6: Size and market segment of company compared to incremental development method and types of increments used. BFF: Bank/finance/insurance, I: Industry, ET: Electronics/telecom, IE: Internet

To analyse the general information we got further, we looked to see if there were any connections between the type of company, the lifecycle model and types of increments they use. Table 5.6 gives an overview of this.

Table 5.6 is sorted from the largest to the smallest company. Company 1, 2 and 3 are actually different divisions of the same company and will therefore be considered as one company in this discussion. One company did not answer the company information questions and is therefore not in this table. To look at the connections between model and size of companies, these categories can be made:

**Small** Less than or equal to 30 employees

**Medium** Between 30 and 200 employees

**Large** More than or equal to 200

Looking at this table, it is hard to find similarities within the different categories with respect to development model and type of increment. There are also no evident similarities between the companies in the same market segments with respect to model and type of increments. Perhaps on a larger data sample, such connections would become more evident, if they exist. But it is easy to see that RUP is the most widely used model in all size categories, either alone or in combination with other models. Also, extreme programming was only used by the medium sized and small companies.
5.3. DISCUSSION

5.3.3 Hypothesis testing

As stated earlier, the response rate on the questionnaire is so low that we cannot draw any statistically valid conclusions. Below, the external, internal and concept validity of the results are evaluated. These concepts have been taken from [WRH00].

External validity of the data seems to be in order, as both the size and field of the companies are distributed on the entire scale. The results are also similar to the distribution of companies found in [Dyb01], which further supports external validity.

Internal validity There are two threats to the internal validity of the results:

Selection is a threat because of the fact that the respondents are more motivated to answering than the population (those who received the questionnaire) as a whole.

Instrumentation This is a threat because the designers of the questionnaire were not experienced in this field.

Construct validity seems to be in order on some, but not all, questions. On the questions linked to $H_1$, the responses are clearly divided into disagree or agree, but on the questions designed for $H_2$, the responses have a larger spread. Many of the respondents are indifferent to the statements, indicating that the statements are poorly phrased.

In section 5.1.4, we have put forth two hypotheses, and a set of questions designed to accept the corresponding null-hypotheses. The hypotheses are rewritten below.

$H_1$ There are advantages to using incremental development.

$H_2$ Incremental development requires more/other customer/developer competence than non-incremental development.

When we consider the material in table 5.3, we note that the results of questions 7 through 11 show a tendency that the subjects accept the statements. These 5 questions are all designed so that if a high number of subjects agree, $H_{1o}$ can be rejected. As close to 70% of the subjects agree on all of these questions, we note a tendency towards acceptance of $H_1$.

Questions 12 through 17 in table 5.3 are designed to reject $H_{2o}$. We note that the responses to these questions are much more scattered than the previous 5 questions, thereby suggesting that we could not accept $H_2$. One should note that a high percentage of the respondents are indifferent to several statements, thereby weakening any conclusion drawn on the data.

To conclude the discussion on the hypothesis we can state the following tendencies:

- $H_1$ seem to be acceptable.
- $H_2$ seem to be unacceptable.
5.3.4 Summary of the main study

- Incremental development has been in widespread use in Norwegian IT industry for several years.
  - Four of the five interviewees had used incremental development for three years or more.
  - A majority of the respondents to the questionnaire used incremental development.
- There are indications that hypothesis $H_1$: 'There are advantages to using incremental development.' may be accepted.
- The same indications does not exist for hypothesis $H_2$: 'Incremental development requires more/other customer/developer competence than non-incremental development.'
- There are no evident connections between the size and domain of software companies and the development model they use.
  - In the table comparing company size and domain with development model and increment scheduling, no evident patterns surface.
  - None of the development models dominated the industry, but RUP was in widespread use.
- Norwegian IT-industry is generally positive to incremental development.
  - In the interviews, the companies proved to have good experiences with incremental development.
  - Companies giving advice in the questionnaire showed a positive attitude to incremental development.
Chapter 6

Conclusion and further work

6.1 Conclusion

To summarise the results of the study, we can state that the results indicate that hypothesis $H1$: 'There are advantages to using incremental development.' may be accepted, while hypothesis $H2$: 'Incremental development requires more/other customer/developer competence than non-incremental development.' may not.

Incremental development is in widespread use in Norwegian IT industry today, and the companies generally have positive experiences with this type of development.

6.2 Further work

Since incremental development is relatively new in software development, there are still many points available for further research. With this project as a starting point, it would be interesting to look closer at the first hypothesis: 'There are no advantages to using incremental development'. As mentioned in chapter 6.1, the results of this project points toward a rejection of this hypothesis. To prove this hypothesis, a larger, statistically valid survey would have to be carried out. This could be done based on the questionnaire in this project, and with consideration to the lessons learned here. Further work on the second hypothesis may be to rephrase the questions to receive less indifferent responses. To achieve this goal, the questions will have to be clearer and more detailed.

As the main impression of this project and [Lik00] is that incremental development is widely used in Norwegian software industry, it might also be interesting to focus more on specific aspects of incremental development in Norway. This can be done in several ways:

- No investigation has been done about incremental development in the public sector in Norway. After all, this is where many software project failures have taken place.

- Many of the companies in this investigation were relatively small (less than hundred employees). It might be interesting to look specifically of some of the larger companies use of incremental development, and perhaps compare this to the use in smaller companies, for instance with respect to methodology used.
Since the big picture of the use of incremental development is becoming clearer, a case study of methods used (RUP, DSDM, XP, Staged Delivery) could be of interest. This could be done through follow up on specific projects, to answer questions like:

- How closely is the methodology followed up?
- How project members experience incremental development?
- How do customers experience incremental development?
- Are some methods more effective than others?

These are questions that a survey cannot answer.

Only three companies that answered this survey did not use incremental development. It could be interesting to look closer at why some companies do not use incremental development, if it is only due to lack of knowledge of it or if it is a deliberate choice.

Evidently, this field is ripe for research, and both academia and industry should take the challenge to extend the knowledge about incremental development.
Appendix A

Interview Questions

The following questions were posed to the five companies that participated in the interviews. The questions have been translated from Norwegian.

The first part, questions 1–8, concerning general information about the company have been taken from [Dyb01].

1. How many employees are working in the company?

2. How many of these work with software development?

3. What is the primary market segment of the company? (choose one)
   - Bank / finance / insurance
   - Public sector
   - Industry
   - Electronics / telecommunications
   - Other:

4. What is the role of software development projects in your company? (choose one)
   - Development/sale of off-the-shelf products
   - Development for external customers
   - Development for internal customers

5. What is your job title?

6. What is your highest completed education? (choose one)
   - Engineering school/cand.mag or equivalent.
   - MSc or equivalent.
   - PhD
   - Other:

7. How many years of software development experience have you got?
8. How many years have you been working in this company?
For the last part of the interview there was an introductory question: Does the company use incremental development? The companies that responded positively to this were asked the following questions. Since all the interviewees used incremental development, the questions about not using incremental development were not used.

9. What does the company mean by incremental development?
- Does the company use the expressions Rapid Application Development, evolutionary development and iterative development?

10. What processes/development models are in use?
- Are the processes formalized?
- Are there any specific tools in use?

11. What are the company’s general experiences with incremental development?

12. For how long has the company been using incremental development?

13. What was used before incremental development was introduced?

14. Who took the initiative to start with incremental development?
- Management or technical people?
- Are the technological personnel motivated to use the methods?
- What kind of competence, and what kind of education have been given?
- Does incremental development demand more competence than earlier methods?

15. Are there any routines for gathering experience from use of processes?
- Are they in use?
- What kind of results have been gleaned from the experiences?

16. How big are the project groups for incremental development projects?
- What are the time frames for these projects (increments and totally)?

17. For what kind of projects does the company use incremental development?
Appendix B

Questionnaire

This questionnaire was distributed during the second part of the project. The questionnaire has been translated from Norwegian.

Introductory question: Does the company use incremental development? (yes/no):
For companies not using incremental development: Answer question 1 through 6 and 25 through 32. Questions 1 through 6 regards the motivation for not using incremental development.
Please rate the statements according to the following key:

1 - totally disagree
2 - partly disagree
3 - indifferent
4 - partly agree
5 - totally agree

1. Incremental development has never been considered used in the company.
2. Our projects are too short for incremental development to give any advantages.
3. Incremental development requires too many resources.
4. Incremental development requires too much re-education of developers and managers.
5. The company is too small to start using incremental development.

6. The customer lacks an understanding for an incremental development process.
For companies using incremental development: Answer question 7 through 32. Questions 7 through 24 regards the motivation for using incremental development.
Please rate the statements according to the following key:

1 - totally disagree
2 - partly disagree
3 - indifferent
4 - partly agree
5 - totally agree

7. A strength with incremental development is that the project group has more control over the requirements.

8. It is an advantage that incremental development encourages close communication with the customer.

9. Staged delivery of the product is important for the customers satisfaction with the result of incremental development projects.

10. Incremental development leads to a better internal prioritising of tasks.

11. Reduction of risks is a central attribute of incremental development projects.

12. Incremental development projects requires more skills from the customer than non-incremental development projects.

13. Incremental development projects requires more skills from the developers than non-incremental development projects.

14. Incremental development projects requires more skills from the management than non-incremental development projects.

15. Formal education of developers is central in carrying out a successful incremental development project.

16. Informal developer support (e.g. mentors and "coffee machines") are crucial in carrying out a successful incremental development project.

17. The project managers role is more important in an incremental development project than in a non-incremental development-project.

Some relevant questions of incremental development: Please answer questions 18 through 24 with a short sentence or a numerical value.

18. Which incremental development model is used in the company (e.g. RUP, DSDM, RAD etc).

19. Are the increments overlapping or sequential?

20. What is the average total length of an incremental development project (in months)?

21. What is the average length of an increment in an incremental development project (in weeks)?

22. What is the average size of a project group in an incremental development project (in number of full-time persons)?
23. How big is a regular incremental development project (in man-years)?
24. Which advice would you give others wanting to use incremental development?
   General information about the company: These questions have been taken from [Dyb01].
25. How many employees are working in the company?
26. How many of these work with software development?
27. What is the primary market segment of the company? (choose one)
   - Bank / finance / insurance
   - Public sector
   - Industry
   - Electronics / telecommunications
   - Other:
28. What is the role of software development projects in your company? (choose one)
   - Development/sale of off-the-shelf products
   - Development for external customers
   - Development for internal customers
29. What is your job title?
30. What is your highest completed education? (choose one)
   - Engineering school/cand.mag or equivalent.
   - MSc or equivalent.
   - PhD
   - Other:
31. How many years of software development experience have you got?
32. How many years have you been working in this company?
Appendix C

Answers to questionnaire

These are the answers received for the questionnaire. The questions are found in appendix B.

C.1 Questions 1 through 23

<table>
<thead>
<tr>
<th></th>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
<th>Q5</th>
<th>Q6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Navia Aviation</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>Ericsson</td>
<td>5</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>HolteProsjekt</td>
<td>4</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>4</td>
</tr>
</tbody>
</table>

Table C.1: Answers questions 1 through 6, companies who do not use incremental development.
<table>
<thead>
<tr>
<th>Company</th>
<th>Q7</th>
<th>Q8</th>
<th>Q9</th>
<th>Q10</th>
<th>Q11</th>
<th>Q12</th>
<th>Q13</th>
<th>Q14</th>
<th>Q15</th>
<th>Q16</th>
<th>Q17</th>
</tr>
</thead>
<tbody>
<tr>
<td>FAST</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>5</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Mogul Technology</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>2</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>Edbasa</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>4</td>
<td>4</td>
<td>5</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>ConsultIT</td>
<td>5</td>
<td>4</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>5</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>Ericsson -utvikling</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>1</td>
<td>1</td>
<td>5</td>
<td>1</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>Ericsson -ETO</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>3</td>
<td>2</td>
<td>4</td>
<td>4</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>Ericsson -ETO</td>
<td>5</td>
<td>4</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>Statoil IT</td>
<td>4</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>5</td>
<td>1</td>
<td>5</td>
<td>3</td>
<td>4</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>SevenMountains Software</td>
<td>4</td>
<td>4</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>4</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>Kognita</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>Finale</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>4</td>
<td>5</td>
<td>5</td>
<td>4</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>Antares</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>5</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Icon Medialab</td>
<td>4</td>
<td>4</td>
<td>5</td>
<td>3</td>
<td>5</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>2</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>Kjoberg Programware</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>3</td>
<td>5</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Tieto Enator</td>
<td>4</td>
<td>4</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>Coetron</td>
<td>4</td>
<td>5</td>
<td>4</td>
<td>5</td>
<td>5</td>
<td>1</td>
<td>4</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>Gjensidige</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>Evita</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>2</td>
</tr>
</tbody>
</table>

Table C.2: Answers to questions 7 through 17, companies who use incremental development.
<table>
<thead>
<tr>
<th>Company</th>
<th>Q18</th>
<th>Q19</th>
<th>Q20</th>
<th>Q21</th>
<th>Q22</th>
<th>Q23</th>
</tr>
</thead>
<tbody>
<tr>
<td>FAST</td>
<td>Staged delivery, XP og RUP</td>
<td>Sekvensielle/parallel</td>
<td>3</td>
<td>Timer/dager - en måned</td>
<td>3-5</td>
<td>ca. 1</td>
</tr>
<tr>
<td>Mogul Technology</td>
<td>RUP</td>
<td>Sekvensielle</td>
<td>6</td>
<td>6</td>
<td>8</td>
<td>2-4</td>
</tr>
<tr>
<td>Edbasa</td>
<td>Det meste</td>
<td>Både og</td>
<td>1-12</td>
<td>1-2</td>
<td>1-14</td>
<td>Ukjent</td>
</tr>
<tr>
<td>ConsultIT</td>
<td>RUP</td>
<td>Delvis overlappende</td>
<td>5</td>
<td>4</td>
<td>8</td>
<td>5</td>
</tr>
<tr>
<td>Ericsson -utvikling</td>
<td>RUP og egendefinerte</td>
<td>Mest sekvensielt</td>
<td>10-12</td>
<td>16</td>
<td>60</td>
<td>50</td>
</tr>
<tr>
<td>Ericsson -ETO</td>
<td>RUP</td>
<td>Sekvensielle</td>
<td>24</td>
<td>9</td>
<td>25</td>
<td>40</td>
</tr>
<tr>
<td>Statoil</td>
<td>Egenutviklet</td>
<td>Begge deler</td>
<td>Ukjent</td>
<td>Ukjent</td>
<td>10</td>
<td>7</td>
</tr>
<tr>
<td>SevenMountains Software</td>
<td>Elementer av RUP/XP</td>
<td>Begge deler</td>
<td>6</td>
<td>4-6</td>
<td>5-10</td>
<td>5</td>
</tr>
<tr>
<td>Kognita</td>
<td>Vet ikke</td>
<td>Overlappende</td>
<td>1</td>
<td>4</td>
<td>2</td>
<td>0,1</td>
</tr>
<tr>
<td>Finale</td>
<td>XP</td>
<td>Sekvensielle</td>
<td>5-8</td>
<td>2-3</td>
<td>7-8</td>
<td>3</td>
</tr>
<tr>
<td>Antares</td>
<td>XP</td>
<td>Sekvensielle</td>
<td>6</td>
<td>2,5</td>
<td>6</td>
<td>2,8</td>
</tr>
<tr>
<td>Icon Medialab</td>
<td>RUP og egenutv. tilegg</td>
<td>Sekvensielle</td>
<td>4-6</td>
<td>1</td>
<td>5-6</td>
<td>1</td>
</tr>
<tr>
<td>Kjoberg Programvare</td>
<td>Ingen bestemt</td>
<td>Noe overlapp</td>
<td>4</td>
<td>4</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>Tieto Enator</td>
<td>RUP</td>
<td>Overlappende</td>
<td>3</td>
<td>4-5</td>
<td>7</td>
<td>4-6</td>
</tr>
<tr>
<td>Gjensidige</td>
<td>RUP</td>
<td>Sekvensielle</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>Evita</td>
<td>DSDM</td>
<td>Både og</td>
<td>2-3</td>
<td>2-4</td>
<td>2-4</td>
<td>1</td>
</tr>
</tbody>
</table>

Table C.3: Answers to questions 18 through 23, companies who use incremental development.
C.2 Question 24

Answers to question 24:

Fast Just do it!

Mogul Sett i gang! Ved bruk av RUP er prosessen i seg selv til enorm hjelp for selve utførelsen av arbeidet, felles forståelse for krav, prioriteringer og tidsplan, og for kvaliteten i arbeidet. Opplæring er vesentlig for å lykkes, det går an å starte et RUP-prosjekt uten at alle har erfaring med RUP, men:

- noen sentrale aktører må ha erfaring,
- flest mulig må ha kunnskap, i form av kurs, etc.
- det bør være en RUP-mentor som arbeider full tid i prosjektet,
- prosjektleder bør være fullt innstilt på at man skal bruke denne metodikken.

Edbasa Bruk hodet. Mye av dette er ikke hekseri men vanlig ledelsesteori

ConsultIT Innfør ny prosess gradvis. Bruk mentorer med praktisk erfaring.

Ericsson utvikling Ta hensyn til at det krever mer detaljert planlegging og oppfølging enn vannfallsprosjekter. Det er en utfordring å sette rammer for prosjektet hvis man skal basere seg på at krav defineres per inkrement. Kravhåndtering blir mer komplisert og krever mer resurser. En kravdatabase er å anbefale.

Ericsson ETO Råd for RUP:

- Begynn med å få iterasjons-mekanismen på plass.
- Fokuser på å automatisere bygge- og integrasjonsprosessen
- Få til automatisk test

Statoil IT Ikke bli 'religiøse'. Dette er bare en av mange metodikker som en profesjonell prosjektgruppe bør håndtere.

Seven Mountains Software Lag en lettvektmetode med litt av det beste fra fra det beste som passer til din bedrift, heller enn å prøve å tvinge bedriften inn i ett fast mønster som f.eks RUP e.l. Legg gjerne XP i 'bunn'.

miljøet oppe! Lek på jobben! Når vi er ferdig med en iterasjon får vi et kinderegg eller en ispinne. Ved releaser kan det være større premier. Ut på en bedre middag etc."

**Antares** Bruk lang tid på å forklare kunden vedkommendes rolle (vektelegg design og test)

**Icon Medialab** Finn ut hvor skoen trykker mest. Plukk elementer fra RUP som kan løse dette problemet. Utvid så med flere deler av prosessen.

**Kjoberg Programvare** Anbefales prøvd.

**TietoEnator** Tilse god forankring og legitimitet for prosjektet i hele organisasjonen.
Styrk kompetansen rundt prosjektstyring, utviklingsprosess og endringshåndtering.
I tillegg til å ha skikkelig gode, erfarne folk på analyse, design og implementasjon.

**Evita** Vær ekstra god på avdekking av kundens behov.
### C.3 Questions 25 through 32

<table>
<thead>
<tr>
<th></th>
<th>Q25</th>
<th>Q26</th>
<th>Q27</th>
<th>Q28</th>
<th>Q29</th>
<th>Q30</th>
<th>Q31</th>
<th>Q32</th>
</tr>
</thead>
<tbody>
<tr>
<td>FAST</td>
<td>175</td>
<td>80</td>
<td>IE</td>
<td>USH</td>
<td>Project manager</td>
<td>SH</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>Mogul Technology</td>
<td>100</td>
<td>80</td>
<td>BFF</td>
<td>UEK</td>
<td></td>
<td>SH</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>Edbasa</td>
<td>300</td>
<td>200</td>
<td>Alle</td>
<td>UEK</td>
<td>Regiondirektør</td>
<td>SH</td>
<td>30</td>
<td>5</td>
</tr>
<tr>
<td>ConsultIT</td>
<td>50</td>
<td>40</td>
<td>ET</td>
<td>UEK</td>
<td></td>
<td>SH</td>
<td>8</td>
<td>4</td>
</tr>
<tr>
<td>Ericsson -utvikling</td>
<td>1000</td>
<td>600</td>
<td>ET</td>
<td>UEK</td>
<td>Kvalitetskoordinator</td>
<td>SH</td>
<td>7</td>
<td>6,5</td>
</tr>
<tr>
<td>Ericsson -ETO</td>
<td>1000</td>
<td>750</td>
<td>ET</td>
<td>UEK</td>
<td>Process manager</td>
<td>D</td>
<td>11</td>
<td>6</td>
</tr>
<tr>
<td>Ericsson -ETO</td>
<td>1000</td>
<td>300</td>
<td>ET</td>
<td>USH, UEK</td>
<td>Kvalitetsjef</td>
<td>SH</td>
<td>10</td>
<td>19</td>
</tr>
<tr>
<td>Statoil IT</td>
<td>450</td>
<td>200</td>
<td>I</td>
<td>UIB</td>
<td>Fagleder</td>
<td>D</td>
<td>10</td>
<td>5</td>
</tr>
<tr>
<td>SevenMountains</td>
<td>30</td>
<td>13</td>
<td>Alle</td>
<td>USH</td>
<td>Utviklingssjef</td>
<td>SH</td>
<td>4</td>
<td>1,5</td>
</tr>
<tr>
<td>Kognita</td>
<td>9</td>
<td>3</td>
<td>I</td>
<td>UEK</td>
<td>Utviklingssjef</td>
<td>SH</td>
<td>7</td>
<td>1,5</td>
</tr>
<tr>
<td>Finale</td>
<td>15</td>
<td>10</td>
<td>BFF</td>
<td>USH, UEK</td>
<td>Systemutvikler</td>
<td>IC</td>
<td>5</td>
<td>3,5</td>
</tr>
<tr>
<td>Antares</td>
<td>80</td>
<td>55</td>
<td>ET</td>
<td>UEK</td>
<td>Prosjektleder</td>
<td>SH</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>Kjøberg</td>
<td>1</td>
<td>1</td>
<td>A</td>
<td>UEK</td>
<td>IT-konsulent</td>
<td>SH</td>
<td>20</td>
<td>10</td>
</tr>
<tr>
<td>Tieto Enator</td>
<td>700</td>
<td></td>
<td></td>
<td>UEK</td>
<td>Seniorkonsulent</td>
<td>IC</td>
<td>6</td>
<td>4,5</td>
</tr>
<tr>
<td>Cœtron</td>
<td>20</td>
<td>10</td>
<td>A</td>
<td>UEK, USH</td>
<td>Vice President</td>
<td>D</td>
<td>14</td>
<td>4</td>
</tr>
<tr>
<td>Gjensidige</td>
<td>350</td>
<td>40</td>
<td>BFF</td>
<td>UIB</td>
<td>Avd.dir.</td>
<td>IC</td>
<td>25</td>
<td>15</td>
</tr>
<tr>
<td>Navia Aviation</td>
<td>150</td>
<td>40</td>
<td>A</td>
<td>UEK</td>
<td></td>
<td>SH</td>
<td>10</td>
<td>8</td>
</tr>
<tr>
<td>Ericsson</td>
<td>800</td>
<td>ca.800</td>
<td>ET</td>
<td>UEK</td>
<td>Ingeniør(QA/CM)</td>
<td>IC</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>HolteProsjekt</td>
<td>35</td>
<td>20</td>
<td>A</td>
<td>UEK</td>
<td>Daglig leder</td>
<td>SH</td>
<td>11</td>
<td>10</td>
</tr>
<tr>
<td>Evita</td>
<td>18</td>
<td>18</td>
<td>BFF, ET</td>
<td>UEK</td>
<td>Daglig leder</td>
<td>SH</td>
<td>12</td>
<td>2,5</td>
</tr>
</tbody>
</table>

Glossary

COTS Components Off the Shelf: developing a system using COTS means using third-party software in the system.

DSDM Dynamic System Development Method: method for iterative development, based on RAD.

IDI Institutt for Datateknikk og Informasjonsvitenskap - Department of Computer and Information Science at NTNU

Ifi Institututt for informatikk - Department of Informatics at UiO

IKT Norge IKT-næringens Interesseorganisasjon - Organisation for Norwegian ICT-industry.

INCO Incremental and Component-based Development: a joint project between NTNU and UiO on how to improve software development

NFR Norges Forskningsråd - The Research Council of Norway

NTNU Norwegian University of Science and Technology, Trondheim

OMT Object Modelling Technique: notation for object oriented modelling proposed by James Rumbaugh.

PROFIT PROcess Improvement for IT industry: PROFIT focuses on developing methods and guidelines for process improvement with emphasis on measurement and reuse of experiences and knowledge, especially for companies in rapidly changing markets.

RAD Rapid Application development: a method to deliver a fit-for-purpose system fast.


UiO Universitetet i Oslo - University of Oslo

UML Unified Modelling Language: Modelling language for visualizing object oriented systems, developed by James Rumbaugh, Grady Booch and Ivar Jacobson.
References


REFERENCES


