Changes in Requirements Management when Introducing RUP to a Legacy System
– a Case Study at Volvo Information Technology

Åsa Nordin
Degree project for master degree in Software engineering

Changes in Requirements Management when Introducing RUP to a Legacy System
- a Case Study at Volvo Information Technology

Åsa Nordin

Examiner:
Lektor Stefan Mankefors
Department of Informatics and Mathematics

Supervisor:
Lektor Stefan Mankefors
Department of Informatics and Mathematics

Trollhättan, 2003
Changes in Requirements Management when Introducing RUP to a Legacy System
- a Case Study at Volvo Information Technology

Åsa Nordin

Sammanfattning
Innehållsförteckning

Sammanfattning ........................................................................................................... ii

Bilagor
1  Overview of RUP ..................................................................................................... 1
2  Overview of the SID system ................................................................................... 2
3  Overview of the scope in SID 5.0 ........................................................................... 3
4  The Development Case of SID 5.0 ......................................................................... 4
Abstract

The Rational Unified Process, RUP, is a development process that is developed and used for green-field development. Since most of the systems that are developed are legacy systems it would be useful to be able to implement RUP in legacy systems. The department 9271 at Volvo IT is probably one of the first to start the implementation of RUP in legacy systems. According to Rational Software Nordic AB have no other company introduced RUP in legacy systems, at least no company that they have knowledge about. Sweden, and the other Nordic countries, are the ones that have come most far in this area. This degree thesis is a case study of the project SID 5.0 at the time the project and the implementation of RUP began. The problems and benefits that may appear are discussed and also how RUP should be introduced in a legacy system. The project was postponed because of the lack of key persons.

1. Introduction

Requirements engineering is not only about identifying the purpose of a software system but also about the context in which the software will be used [1]. Requirements engineering is also about finding the stakeholders and their needs and document them in a way that is useful during the whole software process [2].

The purpose is to find out what services are required and the constraints on the system operation and development. The process of finding out, analyzing, documenting and checking these services is the requirements engineering. Requirements engineering are a critical stage in the software process. It is important to find the requirements because mistakes at this stage inevitably lead to problems during the design and implementation of the system [3].

A lot of the problems in software engineering arise from an imprecise requirements specification. The system developer interprets ambiguous requirements to make the implementation easier. This may cause problem, because the requirements can be different from what the customer wants. New requirements and changes may appear during the whole developing process and that may delay the delivery and increase the cost [3].

When an existing system is going to be further developed problem arises. The existing system must always have the highest priority. If a bug appears, that must be taken care of immediately. That may result in delays for the project. The department 9271 at Volvo IT maintains legacy systems and that makes it difficult to run development projects with priority “one”. There is always risk for delays.

The customer, Volvo Parts, wants two releases each year, which decreases the complexity, makes the installations more manageable and increases the quality.

This case study discusses how Rational Unified Process (RUP) is introduced in an existing application, a legacy system that was not developed with RUP from the beginning. The study is made by following the project SID 5.0 by participating in meetings and workshops, both at Rational Software Nordic AB and at Volvo IT, and by using information in books and on the web.

The purpose was to investigate what problems the introduction of RUP in a legacy system may cause the project and what measures that has to be taken to avoid the pitfalls.

2. Requirements management

Brainstorming, conceptual prototyping, questionnaires and interviews are some techniques to find the requirements. If the software is going to be a product to be sold at the market place the end-user should be replaced by the companies marketing people who have made a competitive analysis [4].

To build an application that will meet the customer’s needs, the project team must define the problem the application is supposed to solve. The project team must also identify the stakeholders and agree with them on a set of application features. The software requirements have to be written in a way that both the customers and the development team understand them. The detailed software requirements are used as input for the system design specifications and for the test plans and the procedures needed for the validation and implementation. To facilitate this, the project team should discuss and agree on some essential matters [4]:

- agree on a common vocabulary for the project
- develop a vision of the system that describes the problem to be solved by the system, as well as its primary features
- elicit stakeholders needs in at least five important areas: functionality, usability, reliability, performance and supportability
determine what requirements types to use
select attributes and values for each requirements type
choose the format in which requirements are described
identify team members who will author, contribute to or simply view one or more types of requirements.
decide what traceability is needed
establish a procedure to propose, review and resolve changes to requirements
develop a mechanism to track requirements history
create progress and status reports for team members and management

These requirements management activities are independent of tools, such as methodology tools, requirements tools, development tools or industry tools.

2.1 Requirements management problems

A lot of problems may arise during the requirements management. Requirements are not always obvious; therefore they may be difficult to find. The requirements can be difficult to express in a proper way in words. There can be too many requirements, they can be time-sensitive and they can change. It is important to understand the customer’s needs. People with end-user experience should be included in the development phase to guarantee that the software fulfills the needs. Requirements may appear from many sources, they all have interest in the outcome of the project. It is important to make a correct evaluation of the different sources of the requirements. Who have the most significant requirements [4]?

Another problem that may appear during the requirements management is the trouble getting access to the persons with key roles; these persons are often involved in other projects at the same time. Resources as time and money are also problematic limitations.

The requirements will inevitable change, no matter how careful the requirements are defined. This fact may be both good and bad. The good thing is that the team is engaging the stakeholders that can contribute to successful projects. The bad thing is that it can be hard to manage changes, especially when they appear late in the project and have impact on other requirements or if the changes are too many [4]. However, a changed requirement means that time has been spent on implementing a particular feature and the requirement may have an impact on other requirements. Activities such as keeping track of the history of each requirements, establishing a baseline, maintaining version control and determine which dependencies are important to trace are included in the management of requirements change [4].

2.2 Requirements concepts

It is important that everyone in the project understands certain requirements management concepts. The concepts include requirements types, cross-functional teams, traceability, multi-dimensional attributes and change history [4]. The requirements type is a class of requirements. By identifying types of requirements they can be organized into more manageable groups that help the project members to handle the requirements. The larger and more complicated the application is, the more different types of requirements appear. A high-level requirement can be divided into other types, for example business roles of vision statement.

A Cross-functional team means that requirements management should involve everyone that can contribute their expertise to the development process. That includes members from the customer and people with different specialties from the developers.

Traceability is a difficult concept but it is essential to be able to adjust changes, because requirements may be dependent on other requirements. The project team has to determine the impact the changes may have and make sure that the traceability relationships are well documented understood and maintained [4].

Multi-dimensional attributes mean that every type of requirements has attributes and every individual requirement has different attribute values.

Changes are inevitable in both individual and in a collection of requirements and that is why change history is important. The history of changes is valuable to keep track of the changes, what changed, when did it change and who authorised the change. A collection of requirements may be associated with a particular version of the application, understanding this allows one to manage change incrementally, reduce risk and improve the probability of meeting milestones [4].
2.3 Different types of requirements

There are three different types of requirements defined by the purpose and who is going to use them [3]. The first two are used by the system end-users. The user requirements are written in natural language and include diagrams that show the services and constraints. Clients and contractors who do not have the technical knowledge of the system use this type of specification. System requirements, also called functional specification, is a more detailed document and people from both the clients and from the contractor use it, for example project managers and senior technical staff. It is often used as a contract between the software developer and the buyer. The third type of requirements is software design specification. It is an abstract description of the software design. The software design specification is a base for detailed design and implementation and it adds more information to the system requirements specification. The software engineers who will develop the system use this type [3].

2.4 Functional and non-functional requirements

Requirements can be classified as functional, non-functional and domain requirements.

Functional requirements – describes the functionality or services that the system is supposed to provide, how the system should respond to different inputs and behave in different situations. They can be written in different levels of detail. Functional requirements can also tell what the system should not do. They are often described in a general but functional way when described as user requirements. Examples of functional requirements are that an ATM machine must be able to give a person that have logged in successfully his or hers money as long as it is enough money on the account or that the ATM machine will confiscate the card if a person have used the wrong PIN code too many times. When they are described as system requirements they are more detailed with its inputs, outputs and exceptions for example. It is important that the functional requirements are both complete and consistent. This means that all the services that the users require should be defined and that the requirements not should have conflicting definitions. It must be taken under consideration that it is almost impossible to achieve consistency and completeness in large and complex systems. In a large system there is different point of views and they often have inconsistent needs that may not be obvious from the beginning [3].

Non-functional requirements – can be of different types and arises from the customer needs. It can be described as requirements that not directly have reference to functions of the system but relate to properties of the system. These properties can concern product requirements such as reliability and portability, organisational requirements such as delivery and implementation and external requirements such as interoperability and ethical requirements. It is common that the non-functional requirements relate to the system as a unit rather than to individual features. As a result of this, the non-functional requirements often are more critical than individual functional requirements. If an individual functional requirement fails, only a part of the system will be affected but if a non-functional requirement fails it may be a problem for the whole system.

An example of this is if a real-time system that controls a nuclear power plant does not fulfil the reliability requirements, and then it can not be used at all. It is a common problem that the non-functional requirements relate to the system as a unit rather than to individual features. As a result of this, the non-functional requirements often are more critical than individual functional requirements. If an individual functional requirement fails, only a part of the system will be affected but if a non-functional requirement fails it may be a problem for the whole system.

Domain requirements – can be functional or non-functional. This kind of requirements comes from the application domain rather than from the needs of the user of the system. This type of requirements is important because if they are not fulfilled, the system will not work
as desired. Copyright laws and implementation of databases in specific ways are examples of domain requirements. A major problem is that they are written in a language that is specific to the application domain; this may lead to misunderstanding because the system developers do not understand it [3].

2.5 Requirements from the user and from the system

When to develop a software system the requirements must be found. A user does not have the same requirements as the system. The purpose with the user requirements is to help people who are involved in the developing process but does not have the technical knowledge to understand the system. The user requirements should be written in natural language and easy to understand. It could also contain diagrams and tables. It should also contain both the functional and the non-functional requirements and it should only cover the external behaviour of the system. The design characteristics should be avoided in this document. Some problem may arise by using a natural language because it lacks clarity and it may lead to misunderstandings or requirements that are complicated to understand. It is also important to define the notions to eliminate indistinctness. Another problem is that the requirements may be difficult to separate when using a natural language. A user requirement that includes too much information limits the system developer. It makes it difficult for him/her to use new and innovative solutions, and it is also much more difficult for the user to understand. It could be a good idea to make some sort of standard when writing user requirements to make it easier to read and to avoid misunderstandings. Requirements have different priority and it is appropriate to mark that with shall and should, depending on how important they are [3]. Key parts can be marked in a specific way, for example with bold or italic typeface and keep the language as easy to read for the non-technical reader as possible. To facilitate for the non-technical reader the user requirements can be separated from the more technical system requirements in the requirements document [3].

The system requirements are a detailed description of the user requirements and the purpose is to serve as a basis for the contract for the implementation of the system a specification of the whole system. It is important that the system requirements are complete and consistent. The document should be made in a way that makes it useful both for the customer and for the developer. It could include models of the system, for example a data-flow model or an object model. The system requirements should, in an ideal situation, state what the system should do; not how it should be implemented. But it is often impossible to exclude that information; the reason could be that the system must interoperate with other existing systems [3]. Even in system requirements specifications natural language can be used. But problems may arise in this document as well as in the user requirements specification. A system requirements specification is a more detailed document and that lead to new problems to take care of. The reader and the writer of the document do not always have the same frame of reference and that may cause problems because they use the same concept in different ways. This may lead to expensive misunderstandings between them, because it is often not discovered until later in the process and that make them harder to amend. Another problem that may appear when using the natural language is the difficulty to find all related requirements and to discover the influence a change may have. It is not possible to take a look at a group of related requirements; instead every requirement must be studied [3].

There are different ways to avoid the problems with the natural language. One way is to use a structured natural language. The advantage is that it is uniformed and therefore is help in avoiding misunderstandings; it is also easy to learn. The disadvantage is that it may limit the terminology. Another way of solving the problem is to use a Program Description Language (PDL). A PDL is a language with origin in programming language, like Ada or Java. The advantage with PDL is that it is possible to check semantically and syntactically by software tools. The disadvantage is that the result is very detailed and too close to the implementation [3].

3. What is Rational Unified Process?

Rational Unified Process is a collection of experiences gathered from a huge number of successful projects. RUP can be considered as a smorgasbord, it is unwise to eat everything, eat only the favourite dishes, the ones that make sense for the specific project [5].

RUP advocates an iterative approach to software development, meaning that a project is
divided into small projects that are made one after another. All iterations that are made have objectivity’s and is completed by delivering an executable, every executable is a step closer to the final system than the last one. Elements of requirements management, analysis and design, implementation, integration and test are used in the iterations that are made [5].

RUP is an industry-wide process platform with six best practices. These best practices are the culmination of 20 years experience from working with customers, partners and other industry leaders. The aim was to find activities that are common in successful results of the projects. This resulted in six best practises and they are fast becoming the de facto industry standard. [6].

Volvo Information Technology has chosen Rational Unified Process, RUP, to be their common application process. RUP was chosen because it was an objective for Volvo IT to use a known, tried and tested standard process that supports modern application development practices. The criterion that was used in the choice of the supplier was that the process must be configurable, it must be iterative, the supplier must have global training and support and they must be well-established [7].

RUP is a customisable framework; it can be adapted to different ways of working. RUP can be described as a Web-enabled software engineering process that enhances team productivity and delivers software best practices to team members, by providing extensible guidelines, templates and examples for all critical e-development activities [7].

RUP is tightly integrated with Rational’s tools; the purpose is to allow development teams to gain full benefits from Rational’s products, Unified Modelling Language (UML) and other industry best practises. This enables the entire team to take full advantage of the six best practices:

1. Develop software iteratively – it is not possible to sequentially first define the entire problem, design, solution, build the software and then test the product at the end with today’s sophisticated software systems. An iterative software process is necessary to allow an increasing understanding of the problem through refinements and to incrementally build the software solution by multiple iteration. Using iterations is a way of handling risks and changes during the development process. The risks should be brought up as early as possible [8]. An iterative approach makes the project lifecycle consisting of several iterations. Every iteration is time-constrained and is ended by a specific milestone of the progress. An iterative approach helps one finding and mitigating risks early, unlike a linear method or a waterfall model [6]. The waterfall model means that every work product and activity must be finished in the phase before moving to the next phase. There is also a modified Waterfall model, which allows iteration to the previous phase [9].

2. Manage requirements – requirements management is an approach, made systematically to find, organise, document and track the changed requirements of a system. The traceability to other requirements and project artefacts must be ensured. Managing the requirements is a help to make sure that the system fulfils the needs of the users and to be able to handle the changes [6]. The comments of use cases and scenarios described in the process are good ways of capturing the functional requirements. They are easy to understand for the users; both the users and the analysts can identify holes in the use case. A use case can almost be seen as a section of the future user manual for the system. Use cases enable to work close to the requirements and thereby ensure that the user requirements are not only documented but also delivered [5].

3. Use component-based architectures – components are groups of code or subsystems that are developed to handle a specific function. A component can be reused. The use of components often makes the software more robust and resilient. The component architecture should be implemented early in the project to minimise the risks and reducing the development costs [6]. The focus is on the early development. The purpose is to make a robust and executable architecture. These best practices describe how to design a flexible architecture that can handle changes, is easy to understand and make effective software reuse possible. RUP supports component-based software development and provides a systematic approach to defining an architecture using new and existing components [8].
4. Visually model software – a best practice that is a tool that shows how to visually model software to capture the structure and behaviour of components and architecture. It is an aid to see how the elements in the system fit together. UML, Unified Modelling Language, is a visual modelling tool that is created by Rational Software; UML has become the industry standard [8]. Modelling visual means that a notation which is graphical, semantically rich and use textual design notations is used. UML is such a notation. The use of a notation language makes it easier to understand for the stakeholders [6].

5. Verify software quality – quality should be reviewed to the requirements based on reliability, functionality and performance of application and system. RUP is useful in planning, design, implementation, execution and evaluation of these test types. Quality is taken under consideration by every developer during the whole development process [8]. If the quality of the software is not verified the result may be poor performance and reliability. The quality should be considered during the whole project lifecycle, each iteration should be tested. The traditional approach was to do the testing of integrated software late in the project [6].

6. Control changes to software – describes how to control, track and monitor changes. Changes are inevitable in an iterative development process. It is possible to establish a workspace for each developer by providing isolation from changes made in other workspaces. Changes of all software artefacts also ensure a more secure development [8]. Co-ordinating the development of software-intensive systems is a big challenge when several developers are involved and they may be located in different places. The problems may be that the developers can work on multiple iterations, releases, platforms or products. To avoid chaos and to be able to use the resources efficiently it is important to establish repeatable procedures for managing the changes that may appear to the software and the artefacts that are involved [6].

3.1 The four phases in RUP

RUP divides a project into four phases, the objectivity’s of them are: the Inception phase, the understanding of what to build, the Elaboration phase, the understanding of how to build it, the Construction phase, the building of a beta version of the system and the Transition phase, the building of the final version of the product. Each phase contains one or more iterations. These iterations focus on the producing of technical deliverables that are needed to achieve the business objectivities of the phase [5].

The iterations could also be described as:
- Inception – the first phase only have one iteration and the iteration is preliminary, which means that it does not result in an executable release. The focus in the Inception phase is on requirements and project management [10]. In a legacy system the Vision Document, Business Case and Initial Development Case, specifying the needed artefacts, should be produced in the Inception phase. Some artefacts, mainly requirements and architecture, should be in the process of reverse engineering. The reason is to enable the choice of an appropriate evolution strategy and to estimate the cost [11].
- Elaboration – next phase is the Elaboration phase and it may consist of one or more iterations. The iteration results in executable release and that is why they are called real iterations according to the definition in RUP. Analysing the domain and reducing risks by producing a stable executable architecture is the focus in the Elaboration phase [10]. The RUP baseline and the few needed artefacts will be completed in one short iteration unless there are too many architectural changes involved. Compared to a green-field development the Elaboration phase already has several artefacts, code in particular, to manage. The Elaboration phase may be the dominant phase and in that case there will not be much to do the Construction and Transition phases [11]. A lot of risks in the project are associated with the architecture, which is why the primary focus in the Elaboration phase is to get the architecture right. Not only the design and architecture but also the implementation and test have to be designed [5].
- Construction – the third phase consists of multiple iterations. These iterations result in executable releases. Completing the system by adding functionality is the focus in this phase [10]. This phase is not very different from any other RUP
The elements are documented, re-engineered and redesigned or translated to another language [11].

- **Transition** – the last phase consists of several iterations, including beta-releases, general availability releases, bug-fix and enhancement releases. The focus is on the activities required to place the software into the hand of the user [10].

The phases have a well-defined ending point where critical decisions have to be made and goals achieved. These ending points are called milestones. If the milestone in the Inception phase, called the Lifecycle Objectives, fails it is decided if the project is going to be cancelled or re-thought. The milestone in the Elaboration phase, Lifecycle Architecture, is the point where the detailed system objectives, the architecture, the scope and the decision about the risks are inspected. The project can still be terminated. The third milestone, the Initial Operational Capability, is the end of the Construction phase. At this milestone it is decided if the users and the software are ready to go operational with what usually are called the “beta” release. If this milestone is not fulfilled satisfyingly the transition phase may have to be postponed one release. The last milestone, the Product Release, ends the Transition phase. This milestone is the point where the decision about if the objectives were met, it is also decided if another development cycle should be started. This milestone sometimes occurs at the same time as the Inception phase is started for the next phase [12].

### 3.2 Disciplines in RUP

RUP consists of several disciplines [A1]. The disciplines are [13]:

- **Business Modelling** - this discipline is not only used to ensure that everyone involved in the process have an understanding of the organisation but also to derive the requirements and understanding the dynamics of the system.

- **Requirements** – the aim is to reach an agreement together with the customer about what the system should do and to be a possibility for the developers to understand the requirements as a basis for the further development. The user-interface should be defined.

- **Analysis & Design** – the requirements are going to be evolved to the design of the system and a robust architecture should be developed. The design should be adjusted to fit the implementation environment and the performance should be taken under consideration.

- **Implementation** – the subsystem should be organised in layers to define the code. Classes and object should be implemented as components. The result of the implementer’s work should be integrated into a complete system.

- **Test** – the purpose is to make sure that all requirements have been implemented and to verify that the interactions between the components are serving as supposed.

- **Deployment** – the system shall be produced and delivered to the customers.

- **Configuration & Change Management** – the purpose is to avoid problems like limited notification, simultaneously updates and multiple versions by controlling the artefacts in the project.

- **Project Management** – the purpose is to handle the framework for managing software intensive projects and for managing the risks. The purpose is also to handle the guidelines for the staff and planning for example.

- **Environment** – the purpose is to support the development team with both the process and the tools. The activities are needed to develop guidelines as a support for the project.

### 3.3 Different roles in RUP

Volvo has adapted the different roles in RUP to fit the needs better. Every role has different artefacts that the role is responsible for. The roles are:

- **Analyst role** – includes the system analyst, the use-case specifier and the tester. Coordinating and leading requirements elicitation and use-case modelling by outlining the systems functionality and delimiting the system is the role of the system analyst. The use-case specifier details the specification of part of the systems functionality, and may also be responsible for the use-case package and the contained use cases and actors. Describing the requirements aspects of one or more use cases accomplish this. Execution of the system is the
responsibility of the system tester. The analyst use these documents [14]:
  • glossary
  • requirements management plan
  • software requirements specification in combination with use-case modelling
  • software requirements specification
  • stakeholder request
  • supplementary specifications
  • use case specification
  • vision
• Developer - this role includes the designer and the user-interface designer. The designer defines responsibilities, relationships, attributes and operations for classes. The designers are also responsible for the determination of how the classes should be adjusted to the implementation environment. The responsibility for design packages and design subsystems, including the classes in them, also belongs to the designer. The user-interface design, the leading and co-ordination of the prototyping and design is the responsibility of the user-interface designer. The user-interface designers capture requirements, both on the user-interface and the usability requirements. They also build prototypes on the user-interface, involve stakeholders in usability reviews and use test sessions. Reviewing and providing feedback for the final implementation of the user-interface are the user-interface designer responsibilities. Artefacts that the analyst use [15]:
  • use case realisation specification
• Project manager - the role project manager can also be a deployment manager. The deployment manager is responsible for the planning and documenting of the plan to transition the product to the user community in the deployment plan. The project manager has the overall responsibility for keeping the project team focused on the goal; he or she allocates resources, co-ordinates interactions with the customers and users and shape priorities. Another responsibility for the project manager is to ensure the quality and integrity of the artefacts in the project [16]. Artefacts that the project manager use
  • business case
  • iteration assessment
  • iteration plan
  • measurement plan
  • problem resolution plan
• product acceptance plan
• quality assurance plan
• risk list
• risk management plan
• software development plan
• status assessment
• bill of materials
• deployment plan
• release notes
• Business analyst - the business designer defines the responsibilities, relationships, attributes and operations of business workers and business entities. A business designer specifies both the business entities and the business entities needed to realise a business use case, the business designer also distributes the behaviour of the business use case to them. He or she also details the specification of a part of the organisation by describing the workflow of business use cases. A business-process analyst co-ordinates and leads business use-case modelling by outlining and delimiting the organisation that is being modelled. Artefacts that the business analyst use [17]:
  • software architecture document
  • business architecture document
  • business glossary
  • business rules
  • business use-case realisation
  • business use-case realisation specification
  • business vision
  • supplementary business specification
  • target-organisation assessment
• Architect - the role of the architect is to co-ordinate and lead technical artefacts and activities throughout the project. The architect also establishes the overall structure for each architectural view, such as the grouping of elements, decomposition of the view and the interfaces between these groups [18].
• Implementer - developing and testing components is the responsibility of the implementer. The developing and testing shall be done in accordance with the projects adopted standard, which enables integration into larger subsystems. When a test component must be created to support testing, the implementer must also develop and test the test components and corresponding subsystems. Drivers and stubs are examples of test components [19].
• Test designer - planning, design, implementation and evaluation of testing is
responsibilities of the test designer. The test designer generates a test plan, a test model and an implementation of the test procedures. He or she also evaluates effectiveness, test coverage and test results. Artefacts that the test designer use [20]:
- test evaluation summary
- test plan
- Process engineer - software development, including configuring the project in the project start and continuously improving the process is the responsibility of the process engineer [21].
- Configuration manager - change control manager and system integrator are similar tasks. Setting up the product structure in the configuration management system, defining and allocating workspaces for developers and integration is the responsibility of the configuration manager. A configuration manager extracts the appropriate status and metrics reports for the project manager. Artefacts that the test designer use [22]:
  - integration build plan
  - configuration management plan
- Tool smith - the responsibility of a tool smith is to develop tools to support special needs and to provide additional automation of tedious or error prone tasks. The tool smith shall also provide better integration between tools [23].

3.4 Software requirements specification, SRS

The software requirements specification is a document in RUP that tells the system developers what is required. This document is also called the software requirements document. It should be made in a way that makes it useful to both system developers and system users. Therefore both the user requirements and the system requirements should be included. The users of the software requirements specification are described like this [3]:
- System customers – specify the requirements and read them to check that they meet their needs. The customers specify changes to the requirements.
- Managers – use the requirements document to plan a bid for the system and to plan the system development process.
- System engineers – use the requirements to develop validation tests for the system.
- System maintenance engineers – use the requirements to help understand the system and the relationships between its parts.

Six requirements are also mentioned, that should be included in the software requirements specification. The document should:
1. Specify only external requirements
2. Specify constraints on the implementation
3. Be easy to change
4. Serve as a reference tool for system maintainers
5. Record forethought about the life cycle of the system
6. Characterise acceptable responses to undesired events

A widely known standard for software requirements specification is the IEEE Std 830-1998 developed by the Institute of Electrical and Electronics Engineers (IEEE). The IEEE Recommended Practice for Software Requirements Specifications describes the approach for the specification of software requirements. According to IEEE the standard for SRS is based on a model that makes the software requirements specification an unambiguous and complete specification document. There are different purposes with this document. One is to help the software customers to describe what they want the system to accomplish and the software suppliers to make sure that they have understood the customer’s needs. On the individual level, a standard software requirements specification (SRS) outline should be developed to fit their organisation. The format should be defined and a quality check list or a SRS writer’s handbook should be developed as a support [24].

3.4.1 Benefits of a good SRS

A good software requirements specification provides several benefits such as [24]:
- Establish the basis for agreement between customer and the supplier of what the software is to do - the customer has a chance to determine whether the software will meet their needs or if the software specification has to be modified.
- Reduce the development effort - if the requirements have been reviewed carefully misunderstandings, omissions and inconsistencies will be discovered easily in the development process.
• Provide a basis for estimating costs and schedules – with a more precise description of the software system the cost is easier to estimate.
• Provide a baseline for validation and verification - the SRS provides a baseline against which compliance can be measured.
• Facilitate transfer – a good SRS facilitates the transfer of the software to new user or new machines.
• Serve as a basis for enhancement – the SRS is about the product and not the project and that makes it possible to use the SRS as a basis for later enhancement.

The SRS often needs to be evolved during the software product process. It is often impossible to specify all details at the time the project is initiated. New details will appear and changes must be done. Therefore the requirements should be specified as completely as possible at the time even if evolutionary revisions probably will appear. It is important to note if a requirement is incomplete. The project changes must be caught by a formal change process to be able to identify, control, track and report them [24].

3.4.2 What characterises a good SRS?

The SRS should be written by one or more persons from the software supplier, one or more persons from the customer, or by both [24]. It is preferred that persons from both the supplier and from the customer participate in the writing of the SRS.

The basic topics are:
• Functionality – what does the customer need?
• External interface – how does the software interact with hardware and with other software?
• Performance – what is the response time, availability and speed etc?
• Attributes – what are the security, portability, and maintainability considerations?
• Design constraints imposed on an implementation – are there any policies for database integrity, implementation language, resource limit or standards in effect?

Design and project requirements should be avoided in the SRS. The design and implementation details should be described in the design stage of the project. Additional constraints of the software should be specified in other documents, for example in the software quality assurance plan.

The software may be a part of a larger system and that means that the rest of the system must be taken under careful consideration. The software can also contain all the functionality of the project. In either case the system requirements must be written in the SRS [24].

A good SRS should be:
• Correct – means that every requirement in the SRS must be ones that the software shall meet.
• Unambiguous – every requirement in the SRS shall have only one interpretation.
• Complete – every significant requirement, regardless of what they are relating to, should be treated. The response to valid and invalid input values shall be specified. It must also have full labels and references to all tables, figures and diagrams. Terms and units of measure must be declared.
• Consistent – the SRS must agree with higher-level document, such as system requirements specification. The requirements must for example state the same thing. Standard terminology and definitions must be used in a standardised way.
• Ranked for importance and/or stability – the importance and/or stability of the requirements must be ranked because some may be desired and others may be essential. The customer must decide this.
• Verifiable – every requirement must be stated in a way that makes them verifiable. It must be able to check if the product meets the requirements. If the requirements can not be stated in a way that makes it possible to measure it, the requirements should be removed from the SRS.
• Modifiable – it must be easy to make changes to the requirements. Each requirement should be expressed separately, no redundant information should exist and the SRS should be easy to use and change.
• Traceable – the SRS must be easy to read and it should be both backward and forward traceable.

The project requirements represents an understanding between the supplier and the customer and it should include cost, delivery schedule, reporting procedure, software development methods, quality assurance, validation and verification criteria and acceptance procedures. These project requirements are not specified in the SRS;
instead they are specified in a software development plan, a software quality assurance plan or a statement of work. The Recommended Practice for Software Requirements Specifications, IEEE Std 830-1998, can be used to create a SRS without modifications but it can also be used as model for a company specific standard [24].

The IEEE standard is not perfect in every situation but it is a template that could be adjusted to fit the specific purpose. When the evolutionary [25] approach is used the requirements document will be less extensive, the designers and the programmers will have to use their judgement to meet the outline user requirements. When the software is a part of a large system the requirements document is very detailed [3].

3.5 The software architecture

The architecture consists of the systems most important building blocks and their interfaces or expressed in other words, the subsystems, their interfaces and the most important components with their interfaces. The architecture provides the skeleton structure in the system, approximately 10 to 20 percent of the code. The architecture also consists of architectural mechanisms, which are common solutions to common problems, for example, how to deal with the garbage collection [5]. The architecture is associated with a lot of risks, which is why the architecture is important. To design, implement and test early in the project is essential to a successful project. These activities are the primary objectives in the Elaboration phase. Many of the most difficult parts have been dealt with when the architecture is in place. This makes it easier to introduce less-experienced team members to the project [5].

3.6 Benefits of using RUP

One important benefit that was mentioned at the workshop, held at Rational Software Nordic AB, was that the use of RUP makes the risks appear earlier than they would have with traditional methods. The fact that the risks appear early can be misunderstood, some believe that RUP bring more risks to the process. RUP is a tool for the system architect to find the problems in the architecture and bring up the risks; it is better to face them in an early stage in the development process. That is why it is important to include the system architect from the beginning of the development process.

There are fundamental principles in RUP, these principles are not specific to green-field development and can therefore be used even in the development of a legacy system. These principles are [11]:

- early risk mitigation
- iterative development
- progress assessment based on concrete, measurable evidence
- organisation around small, empowered teams
- verifying quality continuously
- scope management
- producing only the artefacts that are needed

Probably one of the largest benefits by using an iterative development, compared to the waterfall development, is the possibility to test much earlier that is in the Elaboration phase. The benefit with testing in an early stage is that the function of the architecture can be verified early. This opportunity to feedback can save a lot of time and costs. It is not surprisingly that a common reaction for the teams that have introduced RUP is to claim that the quality has increased. [5].

4. Is it possible to apply RUP in the development of a legacy system?

Since RUP was created to be used in green-field development the possibility to use it in legacy system is discussed. One opinion is that most of RUP can be used in legacy systems but some adjustments must be made to fit the specific system. The factors for the adjustments may depend on how much information that is available, the complexity of the evaluation or the degree of risk that is acceptable. To get the best conditions multiplying of risks should be avoided, for instances do not introduce a new process and a new tool when introducing RUP to a legacy system. These should instead be introduced in an earlier project to give the chance to the developers to get familiar with RUP and the tool before the major legacy evolution [11].

RUP does not contain guidelines and activities for design recovery database conversion or reverse engineering; this will be noticeable when creating the development case. The reason is that these techniques are hard to generalize because they are dependent on the
state the legacy system and the technology that was used [11].

The deployment of a new system are often more delicate than with a green-field development, how much more difficult depends on the scope. A strategy must be chosen if the scope is a new architecture or if a significant part if it is redeveloped. The strategies could be to change the entire system at the same time or to use a phased strategy. Another approach is to have both systems working in parallel until the new system has proved to be reliable [11]. Which alternative that is best must be decided in the specific case. Configuration are required earlier in the process when RUP is used in a legacy system, the reason is that many artefacts exists from the beginning [11].

**4.1 Establish a RUP baseline**

A baseline is used to establish a starting point by identifying a minimal set of artefacts that describes the legacy system. These artefacts may be requirements, architecture and design, tests and user documentation, which ones that are needed is decided by the scope [11]. Then is reverse engineering required, the purpose is to try recreating enough information to proceed as if the system had been developed with RUP from the beginning. When it comes to requirements the most important issue is to provide a minimum specification for the new system, use the existing information, and identify the key use cases. The key use cases can often be found in the user’s manual. The only use cases that need to be detailed are the ones that have to be changed. The installation and marketing documentation often contains information about the non-functional requirements [11]. The configuration management should be introduced in the architecture and design phases. The legacy system does not necessarily have to be redesigned completely with the object-oriented technique. Minimal architectural information is needed and can be accomplished with the Software Architecture Document. The description of interfaces and the scenarios of how these interfaces are performed are critical. The subsystems that are not affected by the evolution will be identified later. [11]. The tests are performed almost the same way as in green-field development. The user documentation for the legacy system is valuable as a base for the new system. The RUP templates can be used but they will probably need to be tailored first to avoid the pitfall a too extensive documentation may be, do not document the elements you do not need [11]. A glossary is a valuable artefact and it should be used during the whole process.

A guideline when it comes to the artefacts is that if there are any doubt about the artefact should be used or not, do not use it [5].

**4.2 Implementing RUP to an existing system**

It is not an easy problem to implement RUP in a legacy system that was not developed with RUP from the beginning since RUP is developed to be used in green-field development. It is difficult to find the requirements and there are no rules to use when using RUP in a legacy system. The development team has to compromise and choose parts of RUP and adjust them to fit the purpose. But the parts can not be adjusted too much; in that case the whole idea with RUP will get lost. It is important to have people in the team that can decide which parts of RUP that shall be used and how they are going to be adjusted. When these persons have made the decisions every one will have to accept that, it is a way of avoiding discussions and misunderstandings. The development team must be aware of that some parts of the development process were better with the traditional method, but everyone must follow RUP anyway to be able to succeed. It is a problem if the project leader does not manage to convince the team members not to make small changes because they are easy to make when they find them. These small changes that are made can cause problem later in the development process, RUP must be followed completely. It takes experience and ability to be able to see the whole picture.

There are three questions that must be answered before one can determine whether a legacy system should be evolved with RUP or not [11]. These questions are:

1. What characterizes the legacy system? A legacy system is often characterized as an old and large system and, in this context, a system that was not developed with RUP from the beginning. The system is most likely not developed with an object-oriented technology. This means that the artefacts are not in created in the RUP form and they also have different names, if they exist at all. The artefacts could also be old and not maintained; therefore they can often not be trusted. Usually, the older the system, the harder to grasp
[11]. A legacy system although have some benefits when it comes to the implementation of RUP, they are a valuable source of information and one can easily identify requirements and business rules, primary use cases, users wishes, behaviours and priorities. What is architecturally significant can also be identified in a legacy system [11].

2. How is the legacy system going to be evolved? There are several ways of evolving the system. It could be a cosmetic makeover, for example a new GUI (Graphical User Interface), extension, some functionality or feature that is going to be added. It could also be migration, the system relies on technologies that are too expensive to keep alive or no longer maintained, or redevelopment, used for example if the system is impossible to scale up, relies on an obsolete hardware or is too hard to evolve. The legacy system is a valuable source of information and should be used to understand important aspects of the system that is going to be evolved. There are situations when the evolving of the system also could be a combination of all these ways, that is a challenging and risky approach, but it is still possible [11].

3. Is the plan a sound business decision? In most cases it is a good decision to keep the legacy system and evolve it, it is often less costly to evolve it than to start a green-field development [11].

4.3 How far has Volvo IT come in the use of RUP in legacy system?

Volvo IT is probably one of the first companies to start using RUP in legacy systems according to Rational Software Nordic AB. The department 9271 (9221 until May 2003) have not used the entire RUP before in legacy systems, but they have used parts of RUP, like use cases and iterations. There have also been a couple of projects that have implemented own modified versions of RUP.

The group manager at department 9271 believes that RUP is very useful in green-field development. He also believes that RUP has the potential to become a very useful tool even in the development of legacy systems that is important since most of the development at the department is development of legacy systems. That is the reason for why the department 9271 is one of the first to implement RUP on legacy systems.

4.4 Do other companies use RUP in legacy systems?

According to Rational Software Nordic AB have no other company introduced RUP in legacy systems, at least no company that they have knowledge about. Sweden, and the other Nordic countries, is the ones that have come most far in this area. Several companies in Sweden are about to begin to use RUP in this way but have not started yet. This opinion is supported by the investigations made during this degree thesis. Most of the companies do not use RUP at all. Some companies, like WM-Data Consulting AB, used RUP for the green-field development at the customers. One company, EDS, did use an own process, called Object & Component Engineering (OCE), which had some similarities to RUP. They have considered beginning using RUP but in that case they are going to introduce RUP in the development of new systems, not in legacy systems.

5. How does Volvo IT introduce RUP in legacy systems?

The material for this degree thesis was gathered by following the project SID 5.0 by attending in meetings and workshops. These workshops and meetings have been internal, together with the customer or led by people from Rational Software Nordic AB.

The purpose was not only to investigate how RUP is implemented in legacy systems and how far the implementation has come but also which problems that may appear.

5.1 VCADS Pro

Volvo Computer Aided Diagnostic System Professional, VCADS Pro, was together with SID 4.0 the first to implement RUP at Volvo IT.

VCADS Pro is a tool for service on excavators, buses and trucks. VCADS Pro makes it possible to test and adjust functions on the machines to make them able to handle specific tasks [26]. To be able to use the big amount of information from the electronic system Volvo has developed this powerful PC-tool. Different control units can be programmed
to optimise the truck for example. The truck registers and stores fault codes in the control unit, this is a valuable help in the search for faults and diagnoses. The mechanics use the PC tool to collect, analyse and program the electrical system [27].

The representative from VCADS Pro explained that VCADS are constantly developed, in the nearest future Mack and Renault are going to be integrated in the system. Earlier, when the waterfall method was used, they had problems with the release dates that were decided in advance, the releases take place three times a year. The waterfall method also resulted in problems with the quality; the customer did not always have the features that they wanted.

Volvo IT and the customer, Volvo Parts, started the discussion, in the autumn 2001, about using a new development process. Volvo IT suggested that RUP should be used and the customer accepted. They decided to choose an adjusted version of RUP called VCUP, VCADS Pro Customised Unified Process. The development of what came to be VCUP started in the project Sebra; the aim was to develop a process that would be suited for the development of VCADS Pro. VCUP could be described as a part of RUP complemented with a small number of artefacts with the aim to suit a legacy system. Today, only one project has been using VCUP. Before VCUP was introduced the project team had been using use cases. Volvo IT had an own method called OOA/OOD, Object Oriented Analysis/Object Oriented Design but the use cases in OOA/OOD was at a lower level.

The transition to VCUP was done by using the old method to a point and then the implementation of VCUP took place in a specific project. This resulted in some problems; new artefacts had to be created before the development of the system could take place. They also tried to create artefacts of the whole system but it took too much time and resources, therefore they chose to take the parts that they needed instead. They are continuously working to document the whole system.

The project team had been prepared to the implementation of VCUP by using small parts of RUP, for example risk list and iteration plan, in the project before VCUP was implemented. Every one involved in the project also had a two days course before the project was started.

Every project has a development case in which all artefacts for the project are gathered. The artefacts that have been used are for example the vision document, stakeholder request, use case model and design model.

In the opinion of the representative from VCADS Pro they have a lot to do before the new process works perfectly but he believes that problems have been discovered earlier in the process and the quality have been raised. Another benefit is that Volvo IT and the customer have increased the understanding of each others situation now when they are working closer together. Before VCUP the customer placed the order and then, after a certain time, Volvo IT delivered the solution, more or less. The use of VCUP involves the customer in the development process in a way that the Waterfall method did not. This interaction between the developers and the customers have led to that the customers requirements are more legible.

The mistakes that were made have been documented and they have investigated how the pitfalls can be avoided in the next projects. Both the way of working and the method are therefore constantly improved. One disadvantage that appears when a new process is introduced, whether it is RUP or not, is the big overhead cost in the first project for the administration since the artefacts does not exist.

Afterwards it is easy to see that some things could have been done different, for example the fact that the process is new to everyone should have been taken under bigger consideration. The scope should have been decided earlier in the project and too much time was spent to set rough time estimation in the beginning of the Inception phase, this should have been done when the requirements were defined better. Another problem was that the use cases should not have been delivered to the implementers; they should have delivered parts of the use cases instead. Now it was difficult for the implementers to understand what was new. The reason is that most of the use cases already were implemented in the legacy system.

The conclusions the VCADS project have done is that the implementation of VCUP is on the right track but it will probably take a year or two before the process is fully implemented and adjusted. The lesson they have learned is the importance of not underestimating the time, costs and difficulties that the change of a new process brings, especially when the process is going to be implemented in a legacy system. They consider that the quality has been raised since VCUP was implemented and that the method is a useful tool in the development of
legacy systems. The tool will be better adjusted for every project.

5.2 Service Information Database, SID

Service Information Database, SID, is a collection of applications that creates methods, handles service information and builds structures [A2]. Volvo Parts is the owner of SID.

5.3 SID 4.0

In the end of 2001 the development of SID 4.0 started. The aim was to develop the release with RUP. It was a big challenge because RUP had only been used on a legacy system in one project before at Volvo IT, in the system VCADS. The project team did not have much experience from RUP and therefore got help from two people outside Volvo IT. The consultant’s approach was too theoretical for the project members; this caused disagreements between them. SID 4.0 did not get a good starting point. The project team experienced a lot of problems. The handling of the documentation was a difficult issue. Some of the problems were that they could not agree on what was specification and what was a requirement in SID was. They lost, in the opinion of one project member, valuable time when they tried to re-create that information. It also took a lot of time to maintain the old documentation. It was necessary to maintain the documentation to be able to use the information in SID 4.0. The artefacts that were produced were at a very high-level, anyone from the street would have understood them and that made them less useful for the project team, since they got a lot of information that they did not needed. That made the artefacts difficult to manage for the project team. The analysis parts also were a problem; there was no system analyst involved. The design people and the requirement people both thought that the other part should do the analyse part; this led to a gap in the process. There was no actual place for the design solutions and that caused problems, it led to that artefacts were moved from the Supplement Specification. The importance of the Supplement Specification was not understood and therefore not used in a proper way. There were a lot of discussions about what type of artefacts that did belong in Supplement Specification and where the ones that did not belong there should be. Iterations are not equals to a release but that was the case in SID 4.0 and that made it difficult to use the feedback from the iteration. The project team had also problems with the converting of the database; it was converted too late. After a while they discovered that the wrong flows had been chosen. They were missing support for an iterative development and that resulted in iterations that were not separated from each other.

The project also had too much focus on the customer and their wishes, which led to reduced focus on the architectural issues.

During the work with the first delivery the team realised that the project SID 4.0 could not be continued with RUP, this was discussed during the workshop at Rational. They did not have the knowledge to fulfil the whole project 4.0 in time and with a high quality if they were going to continue the use of RUP. That is why the decision was made to use RUP in the first delivery and then use the traditional method. It is important to point out that SID 4.0 was a success; the only thing that did not succeed was the using of RUP.

5.4 SID 4.1

SID 4.1 is a project that had a higher priority than SID 5.0. That was the main reason for why SID 4.1 had access to the critical resources, the key persons that SID 5.0 was promised to be able to use, at least for part time. The lack of, for example, the system architect with the overall view of SID and the knowledge of what 4.1 was going to accomplish, were risks for the SID 5.0 project. This problem may appear when two ore more projects are running, more or less, at the same time. SID 4.1 has been delayed but is probably going to be finished in the summer 2003. The aim with the project SID 4.0 is to improve the process used when documents are created.

5.5 SID 5.0

The project SID 5.0 began in spring 2003. The decision to make a new attempt to use RUP had been taken. This time the project team was much better prepared and the problems during the development of SID 4.0 had been taken under consideration. To allow everyone in the project team to discuss what went wrong with
the implementation of RUP in SID 4.0 and to make sure that everyone had the opportunity to discuss how their parts in the development process should be handled, a two days workshop at Rational Software Nordic was arranged.

5.6 The scope of SID 5.0

A set of requirements in a project is called a scope. The scope of a project is a continuous activity that requires iterative or incremental development. The scope of a project is divided into smaller pieces that make them easier to manage. A useful technique for managing the scope is to use requirement attributes, such as risk, priority and effort as a basis for negotiating the inclusion. Changes of the scope that are impossible to handle due to different resources, such as key persons, time and money should be refused. There should be a responsible representative from both the customer and in the project that have the authority to deny such changes to the scope [4].

The question in the SID 5.0 project was how big the scope could be and still be able to be managed within the time limit of the project. The SID 5.0 project also had to take the possibility that the scope consisted of a chain of features that could not be broken under consideration. It is important to find the scope and the delimitation in the beginning of the process.

6. SID 5.0 workshop at Rational

A lot of issues were discussed during the workshop at Rational. How to find use cases in an existing system and how many use cases that could be found were discussed. There are different ways of finding use cases in a legacy system, which way that is the best depends on the system and the situation. In a small system it probably is the easiest way to find them all from the beginning. The use cases could be found during the time, or they could be found when they are needed, this was discussed at the workshop. In SID 5.0 the decision was taken to find the use cases when they were needed. The number of use cases is dependent on how big the project team decided that a use case should be. A rule of thumb to know if the use cases are on the right level is if the users, they who make the design and those who perform the tests are satisfied with the size of the use case when they have studied them. It is important for them that the use cases are useful in their roles. When, for example, the first use case is found it can be studied by the design- and test managers to make sure that the level is correct. A Business Change Request, BCR, is a request for changes in the system from the customer. A BCR must be able to be traced in the existing SID system via the documentation and the functions that are concerned must be able to be found. The use cases can be found from the concerned functions. Another important question that was discussed is where to draw the line for how small changes that is going to be documented in this way.

The fact that SID is not documented the way that it should be, from now on, is an issue that was discussed. How much time and money can be used to document an existing function in SID? The project SID 5.0 could not document the whole system; there were no resources for that.

The mistake with the delivery between the requirements and the design that was made in SID 4.0 was not going to be repeated. Designers were going to be involved in the work with the use case specification in SID 5.0.

The length of iterations had to be taken under consideration because it takes three to five days to move a release from the development environment to the acceptance test environment.

The Vision document was at a very good level for the SID 5.0 project but the level was a bit too low for a traditional vision document. It is possible to make a vision document for the whole SID if desired. Class diagram was going to be used; they were going to be created in Rational Rose.

Efforts should be made to make Supplementary Specification to a valuable artefact, used for the non-functional requirements. A questionnaire can be used to capture these requirements early in the process. The Software Requirements Specification artefact can be used to get a better overview to the Supplement Specification and the use cases.

It was decided that the glossary made in the project SID 4.0 should underlie the glossary for SID 5.0.

The sequence of work that was suggested was to recreate use cases from the features. These use cases does not have to be complete, a name and the main flow is enough. Features and use cases must be discussed with the customer to capture their knowledge about the existing system and to find their actual needs. The features could be placed in these use cases as alternative flows for example.
To solve the problems with what information that should be placed in which artefacts that appeared in 4.0 it was decided that a group of people should be responsible for these decisions.

One issue that was discussed during the workshop was who will have the responsibility to maintain the documentation, and foremost the use cases, produced in 4.1 when the project is completed? Today, no one have that responsibility. Are the artefacts going to be maintained or shall they be re-created when they are needed? Who is going to maintain the artefacts created in 5.0? These questions must be considered.

6.1 RUP workshop in SID 5.0

The project team agreed on the importance to decide early in the project how to work and which artefacts that should be used. One of the problems in the project SID 4.0 was that the consultants were too theoretical, they wanted to settle the whole working procedure from the beginning, the procedure was supposed to be a template for other projects. This resulted in information that was lost or misplaced. One use case was made from the beginning to the end and the others were made at a too high level. It was discussed that a use case diagram for the whole SID should be created, it is difficult to select parts of the system and make use cases from these parts. The use case model can be quite rough in the beginning; parts can be refined when they are needed. But SID 5.0 did not have the resources to do a whole use case model.

The artefacts in RUP are made for green-field development and have to be adjusted according to the consultant from Rational. It is almost impossible to use them in a legacy project without modifications. A unique situation appears when RUP is going to be implemented in a legacy system. How much time and effort is it wise and possible to spend on the development of artefacts according to RUP?

SID 5.0 had a quite special situation when the project did not have access to the key person who has a unique knowledge of the system; this brought the risk that the project could be postponed. The goal that the project team had was to continue working without loosing too much speed and to keep the quality as high as possible. The question was how. One suggestion that was discussed was that they would focus on documenting the requirements without doing any use cases. The risk with this approach is that the use case modelling may not be done at all. The decision was taken to continue to work with use cases.

One of the biggest issues was how much the RUP artefacts should be adjusted. An extreme opinion is to use the artefacts as they are, without modifications; another opinion is that the artefacts must be modified to be useful in a legacy project. But somewhere there is a limit for how much the artefacts can be adjusted and still be called RUP. If the artefacts are modified too much the project team would end up in a difficult situation. No one outside the project would have the ability to help them and it will be hard to find help at Rational’s website. The problem with the modifications is not a new issue, the artefacts have to be adjusted even in green-field development but the problem is more difficult when it comes to legacy systems. One opinion was that changes to the artefacts have to be documented carefully; the changes could be described in the development case or in the artefact that are modified. A risk when RUP is used in legacy systems is that the documentation ends up in a much lower level than when it comes to green-field development project. It is important that the Vision document still is useful for the architect.

It is important to find a way of using RUP that everyone at the department can accept; otherwise it will be hard to use RUP in an efficient way with a high quality.

6.2 Risks in the project SID 5.0 discussed under the meetings.

Since RUP has not been developed for legacy systems and since two parallel projects have not existed in this way before, a lot of new risks appear. The risks that may occur in the project can also be more traditional ones. One of the most serious risks was the lack of the system architect that has the whole overview of the SID system. A lot of information and knowledge is hard to document in a proper way and is therefore unique for the persons that have been involved in the SID projects before. The decision was taken to do the Inception phase with the available resources, otherwise the basis for the decision on how to continue the project will be too inferior. If the project will be delayed too much, the customer may want some other functionality. The risk that the Inception phase develops in the wrong direction was a risk that had to be taken; when the key persons are available they must revise the work done in the Inception phase. Until that point the project had
to find other persons with the knowledge that
the project members lacked.

Another risk was that features that should
have been made in 4.1 could be moved to 5.0.
In worst case these features could be transferred
late in the project.

Since customers are not always available
when they are needed, it may lead to
requirements that are misunderstood by the
project members because they do no always
understand the problem. The priority of the
change requests made by the project team can
be incorrect. The fact that RUP is not developed
to be used in legacy systems resulted in
discussions about how to use RUP and how the
artefacts should be created, these discussions
takes extra time and the result may not be of as
high quality as desired.

Since 5.0 have five actors, TST, GD, VOSP,
VCADS Pro and IMPACT, to deal with, the
boundaries between SID and these actors had to
be taken under consideration. A possible risk
was that issues that belonged to the actors ended
up in the scope of SID 5.0.

The conditions, with the lack of key persons
and with a parallel project, were not an ideal
situation. In fact, it was a difficult task for the
project leader to manage

6.3 The sequence of work in SID 5.0

In the beginning of the process was a vision
document for the project made by the project
team. The vision document is an overall vision
over the system, used to make sure that the
customer and the supplier have the same vision
for the system. The vision document is used to
get an overview of the scope for the project
leader. There is no vision document for the
whole SID system and that was an issue that has
been discussed at the department 9271. A vision
document is usually on a higher level than in
SID 5.0, because when the vision document is
used in a legacy system the features are already
known.

One big issue was to find the use cases and to
make a rough use case model. The question is
how to minimise the risk of a bad architecture,
which use cases are the most important to get a
good architecture? The answer that the project
team had was to find the architecturally
significant use cases; they cover the main tasks
or functions the system is to accomplish. There
may be more than one architecturally significant
use case because SID is a complex system; in
that case they have to be ranked. Time and
effort was spent to define the use cases, when
that was done they had to be compared to the
use cases in 4.1 to see if there were any use
cases that could be used. How far should the
identification of the use cases go in 5.0? Maybe
it is enough to identify them in an overarching
way before they are compared with the ones in
4.1. That may save some time and effort. A use
case that is a pilot for the process was chosen.
The most difficult use case also was chosen. If a
project is impossible to carry out, it is better to
find out from the beginning. The discussion in
the project team was that beginning with the
most difficult use case could result in that they
had made a use case that could not be used as a
support for the rest of the use cases. These use
cases was a test for the process. That is why it
was important to choose the right use cases to
begin with.

It was also discussed if tasks in the
development process that only had to be done
once should be a part of the use case model. It
was decided that they should be named
Activities and placed in a separate artefact.

The priority of the change requests was
different depending on which stakeholders that
were asked. That was a problem because the
priority is important, if the scope must be
reduced for example, the project team must
know which feature that can be removed. A way
of solving this problem may be to make a list in
Microsoft Excel for example and let all the
stakeholders make their own priority. Then the
list should be discussed with the stakeholder
that has the authority to take the final decision.
Other ways to rank the change requests can be
to arrange them in a list with the most important
as number one, and/or to mark them in different
colours, for example number 1 – 7 are red, 8 –
15 are yellow and the least important 16 – 25
are green.

The change requests must also be arranged in
groups to be able to find ones that belong to the
same use case. A way of doing this might be to
look at SID as a box. The different actors and
change requests should be placed in the picture;
this makes it possible to get an overview of the
change requests and their connections [A3].

One person, responsible for the test, was
involved in the process from the beginning to be
able to verify the quality.

6.4 SID 5.0 was postponed

The project SID 5.0 has had different
problems. The worst problems probably were
the lack of the key persons and the limited time
the customer could spend on the project. Both
the key persons and the customer were involved in the project SID 4.1 that have been delayed, this fact have led to that the project team in 5.0 can not continue the Inception phase. The risk that the result not would be worth the effort was too high and therefore the decision to postpone the project was taken together with the customer. The project SID 4.1 had a higher priority and therefore the key persons were focused on that project. When the project 5.0 is going to be completed is not decided, but RUP is going to be used even the next time. The work that already has been done in SID 5.0 is going to be preserved. The purpose is to use the artefacts and the knowledge that have been accomplished so far when the project is continued.

As a support for the continuing work with RUP in the department 9271 a standard use case is going to be completed. A standard use case is also called a “process use case”; a typical example of a use case that can be used in the rest of the project. SID 5.0 will have access to one of the key persons in SID 4.1 and people from Rational Software Nordic AB when the process use case is going to be made. The purpose is to find and document the knowledge and the artefacts that have been made. The purpose is also to document a process use case in an adequate level that can be used in the further development process. The legacy specific issues that have appeared should be documented, they are a valuable help in the continuing work of implementing RUP in legacy systems.

The work that has been done so far is documented in the development case [A4]. The Construction and Transition phases are removed from the development case since it is an artefact that should be updated at least in every iteration. The process is iterative and these two phases does not need to be in the artefact at this moment.

7. The key to success

A recommendation when a company is going to implement RUP, or any other development process, in a legacy system is to choose a suitable project. There are no perfect projects but the project should be as simple as possible, the more complex the project is, the more difficult the implementation will get.

Another factor that is important is to have access to the resources, it is necessary to be able to use key persons with knowledge about the system and the actors that will interact with the system. It is also important that the project team have knowledge about RUP; it is valuable if the team have used RUP before, in green-field development projects.

The level of the documentation in the artefacts is an important decision to take. The artefacts must be adjusted to fit the needs of the legacy system because the level is not suitable for a legacy system. The artefacts must be at a level where the different roles in the development process get the maximum benefits by using them. Therefore persons from the different roles should take part in the decision of the documentation level.

The knowledge of the legacy system should be used as a valuable source of information, instead of something that makes the development more difficult.

Workshops together with people with great knowledge of RUP and mentors that supports the implementation of RUP is valuable to be able to succeed.

How the implementation should be done depends on the legacy system and the project team. In a small system and with a project team that have used RUP in several projects before it might be possible to implement the whole RUP at the same time. Otherwise it might be easier to implement parts of RUP in the first project and then gradually use more of RUP in the following projects.

8. The customer’s opinion about the use of RUP in SID

The customer, Volvo Parts, considers the use of RUP as a positive initiative. The job to find a development process that is adjusted to the changeable world that we live in today is seen as something worth fighting for. The method must also be transferred to Volvo Parts, otherwise the customer might create an own version of the method and some of the benefits will be lost. If Volvo IT succeed to make RUP into a functional method with a reasonable level of ambition Volvo Parts believes that RUP can be a useful tool.

The risk that Volvo Parts points out is if not everyone at Volvo IT, and to a certain degree Volvo Parts, understands the method and the principles. The customer also believes that they have to prepare their instructions to make them useful for Volvo IT. Since the final customer, the retailers, does not understand the method, there is a risk that Volvo IT might miss the point, to develop for the final customers, with the method.
The customer have not seen the whole process in RUP and therefore miss some parts, for example they want describing pictures, that they would have considered as a help for them to understand the process. Volvo Parts also wants a closer co-operation with Volvo IT in the documentation of the requirements. A guideline for how the requirements should be handled at Volvo Parts is missing, they are aware of that this fact may cause problems to Volvo IT. The priority in Volvo Parts requirements could be better defined from the beginning in their own opinion. A question that has occurred is how the interface is going to be handled in the process.

9. Result

A common problem in software development is to find and fulfil the requirements. It is important that the requirements from the customer are taken under consideration. But the software developers must be aware of the fact that the customer does not always know exactly what problems the software system is supposed to solve and which demands they should have on the system. This may lead to a system that is not used in the proper way, or in worst case, not used at all!

An important step in the understanding of requirements management is to use a vocabulary that is common and known among the involved persons.

When a new method is going to be introduced, education and understanding for why a new method is going to be used are probably the most important issues.

To succeed with the implementation of RUP in a legacy system it is important to keep things as simple as possible, do not create artefacts and execute activities that are unnecessary [11].

The implementation of RUP can be made in different ways. One way is to do the whole implementation at once; another is to begin the implementation by choosing one phase of RUP. A third way is to start the implementation by taking pieces from all phases. Which alternative is the best? That depends on several factors, for example the specific project, the available resources and the knowledge of the legacy system and of RUP in the development team.

One important conclusion from this work is that it is possible to try to implement RUP in a legacy system; if it does not work one can always return to the method that was used before. Learn from the mistakes and be better prepared the next time a new method, for example RUP, is implemented. Changes are not the enemy; unmanaged changes are!

10. Future work

A suggestion for complementary work would be to follow the project SID 5.0 in the continuing process. Which artefacts are chosen and how they are used in the project is also suggestions for future work. RUP could also be compared to other development processes to find the advantages and the disadvantages the different processes have. The implementation of a legacy system developed with RUP is a subject that may suit as a complementary work.

11. Acknowledgements

I would like to thank Kurt Andersson, the group manager for the department 9271, who made it possible for me to do my degree thesis in Volvo IT and Anders Ödman, the project leader in SID 5.0, for helping me to specify the subject and letting me follow the project at close quarters. Many thanks to the members in the project team, Lennart Östlund, Mikael Ljunggren, Anna Ortmark and Björn Olson, they are angels and I have learned a lot from them. I would like to thank the representatives from the customer Petra Rikås and Andreas Hanson for letting me participate in the meetings and for the information they have given me. Finally, I would also like to thank Anders Jonsson, at Rational Software Nordic AB, Håkan Elvingson at Volvo IT, Stefan Mankefors, the supervisor and examiner, and everyone else that have helped me during this degree thesis. I have really appreciated my time in Volvo IT!
12. References

http://www.re01.org/


http://www.rational.com/media/whitepapers/app
rmuc.pdf

http://www.therationaledge.com/content/dec_01 
/>spiritOfTheRUP_pk.html

http://www.rational.com/corpinfo/practices.jsp

http://rup_volvo.se/stra/index.htm

http://rup_volvo.se/prod/index.htm


http://maps_volvo.se/rup/users%20guide/phases/default.htm

http://www.therationaledge.com/content/ent_0 
>May_legacy_pk.html

http://www.rational.com/media/whitepapers/rupt 
>bestpractices.pdf?SMSESSION=NO

http://maps_volvo.se/rup/users%20guide/discipli 
>nes/default.htm

http://maps_volvo.se/rup/users%20guide/roles/a 
>nalyst/default.htm

http://maps_volvo.se/rup/users%20guide/roles/d 
eveloper/default.htm

http://maps_volvo.se/rup/users%20guide/roles/pr 
>ject%20manager/default.htm

http://maps_volvo.se/rup/users%20guide/roles/b 
>usiness%20analyst/default.htm

http://maps_volvo.se/rup/users%20guide/roles/ar 
>chitect/default.htm

http://maps_volvo.se/rup/users%20guide/roles/i 
>mplementer/default.htm


Appendix 1

Overview of RUP

This picture from Rational Software Corporation shows an overview of RUP.
Appendix 2

Overview of the SID system

This is an overview of the SID system as it is today. In SID 5.0 it will be possible to export information to the IMPACT system when it is needed. The systems Guided Diagnostics, TST, VCADS Pro and VOSP will be included to this overview.
Appendix 3

Overview of the scope in SID 5.0

This picture is made by Mikael Ljunggren, one of the members in the project SID 5.0. The picture shows for example the actors, the Change Requests (CR).
Appendix 4

The Development Case of SID 5.0

This Development Case is made for the project SID 5.0.
SID 5.0

Document name: Development Case – SID 5.0
Issued by: Anders Jonsson
Date: 2003-03-14
Reg. no:
Description:
Approved:
Version:
Replaces:
Reviewed:
Distribution:
RUP Version 2002.05.00

Revision History

<table>
<thead>
<tr>
<th>Date</th>
<th>Version</th>
<th>Description</th>
<th>Author</th>
</tr>
</thead>
<tbody>
<tr>
<td>2003-04-13</td>
<td>0.8</td>
<td>Created</td>
<td>Anders Jonsson</td>
</tr>
</tbody>
</table>
Table of Contents

Sammanfattning ........................................................................................................ ii
Revision History ........................................................................................................ 2
Table of Contents ...................................................................................................... 3
Development Case ................................................................................................... 6
Introduction ............................................................................................................... 6
  Purpose ................................................................................................................... 6
  Scope ....................................................................................................................... 6
  Definitions, Acronyms, and Abbreviations ......................................................... 6
  Abbreviations ........................................................................................................ 6
  References .............................................................................................................. 7
Overview of the Development Case ........................................................................ 7
  Lifecycle Model .................................................................................................... 7
  Disciplines ........................................................................................................... 8
  Discipline Configuration ....................................................................................... 9
  Other Issues ......................................................................................................... 9
Disciplines ............................................................................................................... 10
  Business Modeling ............................................................................................ 10
    Discipline .......................................................................................................... 10
    Artifacts ............................................................................................................ 11
    Notes on the Reports ......................................................................................... 11
    Additional Review Procedures ......................................................................... 11
    Other Issues ..................................................................................................... 11
    Configuring the Discipline ............................................................................. 11
Requirements ........................................................................................................... 12
  Activity Overview .............................................................................................. 12
  Artifacts .............................................................................................................. 12
  Notes on the Artifacts ......................................................................................... 13
  Reports ............................................................................................................... 13
  Notes on the Reports ......................................................................................... 13
  Additional Review Procedures .......................................................................... 13
  Other Issues ..................................................................................................... 13
  Configuring the Discipline ............................................................................. 13
Analysis & Design .................................................................................................. 14
  Activity Overview .............................................................................................. 14
  Notes on the Artifacts ......................................................................................... 14
  Reports ............................................................................................................... 14
  Notes on the Reports ......................................................................................... 15
  Additional Review Procedures .......................................................................... 15
  Other Issues ..................................................................................................... 15
  Configuring the Discipline ............................................................................. 15
Implementation .................................................................................................................................................. 15
Activity Overview .............................................................................................................................................. 15
Artifacts .......................................................................................................................................................... 15
Notes on the Artifacts .................................................................................................................................. 15
Reports .......................................................................................................................................................... 16
Notes on the Reports .................................................................................................................................... 16
Additional Review Procedures ..................................................................................................................... 16
Other Issues .................................................................................................................................................. 16
Configuring the Discipline .......................................................................................................................... 16

Test 16 ............................................................................................................................................................ 16
Activity Overview .............................................................................................................................................. 16
Artifacts .......................................................................................................................................................... 17
Notes on the Artifacts .................................................................................................................................... 17
Reports .......................................................................................................................................................... 17
Notes on the Reports .................................................................................................................................... 18
Additional Review Procedures ..................................................................................................................... 18
Other Issues .................................................................................................................................................. 18
Configuring the Discipline .......................................................................................................................... 18

Deployment ................................................................................................................................................... 18
Activity Overview .............................................................................................................................................. 18
Artifacts .......................................................................................................................................................... 19
Notes on the Artifacts .................................................................................................................................... 19
Reports .......................................................................................................................................................... 19
Notes on the Reports .................................................................................................................................... 19
Additional Review Procedures ..................................................................................................................... 19
Other Issues .................................................................................................................................................. 19
Configuring the Discipline .......................................................................................................................... 19

Configuration & Change Management ........................................................................................................ 19
Activity Overview .............................................................................................................................................. 19
Artifacts .......................................................................................................................................................... 19
Notes on the Artifacts .................................................................................................................................... 19
Reports .......................................................................................................................................................... 20
Notes on the Reports .................................................................................................................................... 20
Additional Review Procedures ..................................................................................................................... 20
Other Issues .................................................................................................................................................. 20
Configuring the Discipline .......................................................................................................................... 20

Project Management ................................................................................................................................. 20
Activity Overview .............................................................................................................................................. 20
Artifacts .......................................................................................................................................................... 20
Notes on the Artifacts .................................................................................................................................... 20
Reports .......................................................................................................................................................... 21
Notes on the Reports .................................................................................................................................... 21
Additional Review Procedures ..................................................................................................................... 21
Other Issues .................................................................................................................................................. 21
Configuring the Discipline .......................................................................................................................... 21

Environment ................................................................................................................................................. 21
Activity Overview ........................................................................................................................ 21
Artifacts ........................................................................................................................................ 21
Notes on the Artifacts .................................................................................................................. 22
Reports .......................................................................................................................................... 23
Notes on the Reports .................................................................................................................... 23
Additional Review Procedures .................................................................................................... 23
Other Issues ................................................................................................................................. 23
Configuring the Discipline .......................................................................................................... 23

Roles ........................................................................................................................................... 24

Appendix A .................................................................................................................................... 26
Sample Iteration Plans .................................................................................................................. 26
Development Case

Introduction

Purpose
Rational Unified Process (RUP) is a software engineering process (SEP). As there is no universal SEP, each project has to tailor RUP for its specific needs. The document describes the use of RUP and existing method in SID 5.0 project. The description is done in terms of deviations from RUP. Where not otherwise specified, the descriptions in RUP Online (Ref. 1) shall be used. Note that the Development Case is a "living" artifact (document).

For more information about the Development Case, see Rational Unified Process – Artifacts, section, Environment.

Scope
All members in the project must read this document in order to understand how RUP together with other, existing process/methodology parts is to be employed in this particular project.

This document is the definition of which artifacts the project shall produce. With this information, project members use steps and guidelines in RUP etc to find out how to do it. Additionally, RUP together with the plans in the project (Project Charter and Iteration Plans) defines who should do it and when.

Definitions, Acronyms, and Abbreviations
This subsection provides definitions of all terms, acronyms and abbreviations required for properly interpreting this document.

Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>CM</td>
<td>Configuration Management</td>
</tr>
<tr>
<td>N/A</td>
<td>Not applicable. Meaning that the referred item has no meaning in its context</td>
</tr>
<tr>
<td>RUP</td>
<td>Rational Unified Process</td>
</tr>
<tr>
<td>SEP</td>
<td>Software Engineering Process</td>
</tr>
</tbody>
</table>
References

This subsection provides a complete list of all documents and other items referenced elsewhere in this document. The documents are identified by title, number, revision and publishing organization.

Ref. 1 Rational Unified Process 2002.05.00
Ref. 2 Project Charter XXX, No XXX, Rev YYY, VIT

Questions are handled by XXX
Phone: +4631 XXX
Email: YYYY

Overview of the Development Case

To ensure that you understand the material presented here, you should first read

- RUP - Project Management. Use this in parallel with descriptions of the RUP disciplines.
- For the details of a core discipline, refer to it in RUP Online.

Lifecycle Model

The disciplines can be applied to almost any project model. For this to be done, the following plans must be formulated.

- **Project Charter.** The layout of the four phases (Inception, Elaboration, Construction, and Transition), and a specification of what will be achieved at each major milestone and when. This is called a phase plan and is captured in one section of the Project Charter.

- **Iteration Plan.** The organization of each iteration.

Each iteration consists of planning (project management), requirements, analysis & design, implementation, and test, configuration & change management.
Disciplines

This Development Case covers the disciplines:

- Business Modeling
- Requirements
- Analysis & Design
- Implementation
- Test
- Deployment
- Configuration & Change Management
- Project Management
- Environment

Project Management is adopted to support PCM.
Discipline Configuration

Other Issues
Disciplines

**Business Modeling**

The Business Modeling discipline is optional in the project. Most projects at Volvo IT will only do domain modeling.

For details on the activities, see the Rational Unified Process – Business Modeling discipline.

**Discipline**

The only workflow detail used is “Develop a Domain Model”.
## Artifacts

<table>
<thead>
<tr>
<th>Artifacts</th>
<th>How to Use</th>
<th>Review Details</th>
<th>Tools Used</th>
<th>Templates/Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Incep</td>
<td>Elab</td>
<td>Const</td>
<td>Trans</td>
</tr>
</tbody>
</table>

### Notes on the Reports

- Additional Review Procedures
- Other Issues
- Configuring the Discipline
Requirements
This section describes how the Requirements discipline is to be employed in the project.

Activity Overview

Artifacts

<table>
<thead>
<tr>
<th>Artifacts</th>
<th>How to Use</th>
<th>Review Details</th>
<th>Tools Used</th>
<th>Templates/Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirements Management Plan</td>
<td>Could</td>
<td>Informal</td>
<td>Word</td>
<td></td>
</tr>
<tr>
<td>Requirements Attributes</td>
<td>Could</td>
<td>Informal</td>
<td>Requisite Pro + Word</td>
<td></td>
</tr>
<tr>
<td>Glossary</td>
<td>Must Shoul d</td>
<td>Formal-External (non-formal communication between Customer and Project)</td>
<td>Word</td>
<td></td>
</tr>
<tr>
<td>Change Request / BCR (Stakeholder Requests)</td>
<td>Must</td>
<td>Formal-External</td>
<td>Requisite Pro + Word</td>
<td></td>
</tr>
<tr>
<td>Vision</td>
<td>Must</td>
<td>Formal-External</td>
<td>Requisite Pro + Word</td>
<td></td>
</tr>
<tr>
<td>Use-Case Model</td>
<td>Must</td>
<td>Formal-External</td>
<td>Rose</td>
<td></td>
</tr>
<tr>
<td>Actor</td>
<td>Must</td>
<td>Formal-External</td>
<td>Rose</td>
<td></td>
</tr>
<tr>
<td>Use Case</td>
<td>Must</td>
<td>Formal-External</td>
<td>Requisite Pro + Rose</td>
<td></td>
</tr>
<tr>
<td>Use-Case Package</td>
<td>Could Could</td>
<td>Formal-External</td>
<td>Rose</td>
<td></td>
</tr>
<tr>
<td>Supplementary Specification</td>
<td>Must</td>
<td>Formal-External</td>
<td>Requisite Pro</td>
<td></td>
</tr>
<tr>
<td>User-Interface Prototype</td>
<td>Could Could</td>
<td>Informal</td>
<td>Any development tool or MS Powerpoint</td>
<td></td>
</tr>
</tbody>
</table>
Software Architecture Document (Section Use-Case View)

<table>
<thead>
<tr>
<th>Artifacts</th>
<th>How to Use</th>
<th>Reason</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boundary Class</td>
<td>Won’t</td>
<td>(Artifacts)</td>
</tr>
<tr>
<td>Software Requirements</td>
<td>Won’t</td>
<td>Normally replaced by Use-Case Model and Supplementary specification. This is the normal usage of RUP.</td>
</tr>
<tr>
<td>Specification</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Reports

<table>
<thead>
<tr>
<th>Reports</th>
<th>How to Use</th>
<th>Tools Used</th>
<th>Templates/Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actor</td>
<td>Could</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Use-Case</td>
<td>Could</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Use-Case Model Survey</td>
<td>Could</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Notes on the Reports

Additional Review Procedures

Other Issues

Configuring the Discipline
## Analysis & Design

This section describes how the Analysis & Design discipline is to be employed in the project.

### Activity Overview

**Artifacts**

<table>
<thead>
<tr>
<th>Artifacts</th>
<th>How to Use</th>
<th>Tools Used</th>
<th>Templates/Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deployment Model</td>
<td>Could</td>
<td>Informal</td>
<td>Rose</td>
</tr>
<tr>
<td>Data Model</td>
<td>Won’t</td>
<td>Informal</td>
<td>Rose</td>
</tr>
<tr>
<td>Design Class</td>
<td>Won’t</td>
<td>Informal</td>
<td>Rose</td>
</tr>
<tr>
<td>Design Model</td>
<td>Could</td>
<td>Formal</td>
<td>Rose</td>
</tr>
<tr>
<td>Design Package</td>
<td>Could</td>
<td>Informal</td>
<td>Rose</td>
</tr>
<tr>
<td>Design Subsystem</td>
<td>Could</td>
<td>Informal</td>
<td>Rational Rose</td>
</tr>
<tr>
<td>Interface</td>
<td>Won’t</td>
<td>Formal</td>
<td>Rose</td>
</tr>
<tr>
<td>Reference Architecture</td>
<td>Could</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Software Architecture</td>
<td>Shoul</td>
<td>Formal</td>
<td>Rose + SoDA</td>
</tr>
<tr>
<td>Document</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Use-Case Realization</td>
<td>Could</td>
<td>Informal</td>
<td>Rose</td>
</tr>
</tbody>
</table>

### Notes on the Artifacts

The following artifacts are **not** used:

<table>
<thead>
<tr>
<th>Artifacts</th>
<th>How to Use</th>
<th>Reason</th>
</tr>
</thead>
<tbody>
<tr>
<td>Capsule</td>
<td>Won’t</td>
<td>Only for usage in a real-time environment.</td>
</tr>
<tr>
<td>Event</td>
<td>Won’t</td>
<td>Only for usage in a real-time environment.</td>
</tr>
<tr>
<td>Protocol</td>
<td>Won’t</td>
<td>Only for usage in a real-time environment.</td>
</tr>
<tr>
<td>Signal</td>
<td>Won’t</td>
<td>Only for usage in a real-time environment.</td>
</tr>
<tr>
<td>Analysis Class</td>
<td>Won’t</td>
<td></td>
</tr>
<tr>
<td>Analysis Model</td>
<td>Won’t</td>
<td></td>
</tr>
</tbody>
</table>

### Reports

<table>
<thead>
<tr>
<th>Reports</th>
<th>How to Use</th>
<th>Tools Used</th>
<th>Templates/Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class</td>
<td>Could</td>
<td></td>
<td>Rose + SoDA</td>
</tr>
<tr>
<td>Design-Model Survey</td>
<td>Could</td>
<td></td>
<td>Rose + SoDA</td>
</tr>
<tr>
<td>Design Package/Subsystem</td>
<td>Could</td>
<td></td>
<td>Rose + SoDA</td>
</tr>
</tbody>
</table>
Use-Case Realization: Could

Notes on the Reports

Additional Review Procedures

Other Issues

Configuring the Discipline

**Implementation**

A description of how the Implementation discipline is to be employed in the project.

**Activity Overview**

**Artifacts**

<table>
<thead>
<tr>
<th>Artifacts</th>
<th>How to Use</th>
<th>Use</th>
<th>Elab</th>
<th>Const</th>
<th>Trans</th>
<th>Review Details</th>
<th>Tools Used</th>
<th>Templates/Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Build</td>
<td>Must</td>
<td>Must</td>
<td></td>
<td></td>
<td></td>
<td>TBD</td>
<td>Rose + any development tool</td>
<td></td>
</tr>
<tr>
<td>Component</td>
<td>Could</td>
<td>Must</td>
<td></td>
<td></td>
<td></td>
<td>Informal (Source code is formal internal, readme-files are formal external)</td>
<td>Rose + any development tool</td>
<td></td>
</tr>
<tr>
<td>Implementation Model</td>
<td>Could</td>
<td>Must</td>
<td></td>
<td></td>
<td></td>
<td>Informal</td>
<td>Rose</td>
<td></td>
</tr>
<tr>
<td>Implementation Subsystem</td>
<td>Won’t</td>
<td>Could</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Notes on the Artifacts**

The following artifacts are not used:

<table>
<thead>
<tr>
<th>Artifacts</th>
<th>How to Use</th>
<th>Reason</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integration Build Plan</td>
<td>Won’t</td>
<td>The complexity of the products developed at Volvo IT normally does not need a Integration Build Plan.</td>
</tr>
</tbody>
</table>
### Test

This section describes how the Test discipline is to be employed in the project.

#### Activity Overview

---

### Reports

<table>
<thead>
<tr>
<th>Reports</th>
<th>How to Use</th>
<th>Tools Used</th>
<th>Templates/Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Notes on the Reports

### Additional Review Procedures

### Other Issues

### Configuring the Discipline

### Test

This section describes how the Test discipline is to be employed in the project.

#### Activity Overview

---
Artifacts

For this project, the following artifacts shall be produced for this discipline.

<table>
<thead>
<tr>
<th>Artifacts</th>
<th>How to Use</th>
<th>Review Details</th>
<th>Tools Used</th>
<th>Templates/Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test Case</td>
<td>Could</td>
<td>Formal-Internal</td>
<td>Word</td>
<td></td>
</tr>
<tr>
<td>Test Class</td>
<td>Could</td>
<td>Internal</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Test Components</td>
<td>Could</td>
<td>Internal</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Test Suite</td>
<td>Could</td>
<td>Must</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Test Plan</td>
<td>Should</td>
<td>Formal-External</td>
<td>Word</td>
<td></td>
</tr>
<tr>
<td>Test Results</td>
<td>Won’t</td>
<td>Informal</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Test Script</td>
<td>Could</td>
<td>Must</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Test Evaluation Summary</td>
<td>Should</td>
<td>Formal-Internal</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Test Log</td>
<td>Could</td>
<td>Internal</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Test Ideas List</td>
<td>Could</td>
<td>Internal</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Test Data</td>
<td>Could</td>
<td>Must</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Test Automation Architecture</td>
<td>Could</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Test Interface Specification</td>
<td>Could</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Test Environment Configuration</td>
<td>Could</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Workload Model</td>
<td>Won’t</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Notes on the Artifacts

The following artifacts are not used:

<table>
<thead>
<tr>
<th>Artifacts</th>
<th>How to Use</th>
<th>Reason</th>
</tr>
</thead>
</table>

Comment: Test scripts are the computer readable instructions that automate the execution of a test procedure (or portion of a test procedure). Test scripts may be created (recorded) or automatically generated using test automation tools, programmed using a programming language, or a combination of recording, generating, and programming.
Notes on the Reports

Additional Review Procedures

Other Issues

Configuring the Discipline

Deployment

[See the section titled Discipline Configuration that describes what each of the following sections needs to contain.]

Activity Overview
Artifacts

TBD.

<table>
<thead>
<tr>
<th>Artifacts</th>
<th>How to Use</th>
<th>Review Details</th>
<th>Tools Used</th>
<th>Templates/Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Incep</td>
<td>Const</td>
<td>Trans</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Elab</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Const</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Trans</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Notes on the Artifacts

<table>
<thead>
<tr>
<th>Artifacts</th>
<th>How to Use</th>
<th>Reason</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Reports

<table>
<thead>
<tr>
<th>Reports</th>
<th>How to Use</th>
<th>Tools Used</th>
<th>Templates/Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Notes on the Reports

Additional Review Procedures

Other Issues

Configuring the Discipline

**Configuration & Change Management**

A description of how the Configuration & Change Management discipline is to be employed in the project.

Activity Overview

Artifacts

<table>
<thead>
<tr>
<th>Artifacts</th>
<th>How to Use</th>
<th>Review Details</th>
<th>Tools Used</th>
<th>Templates/Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Change Request (technical, project specific (scope) )</td>
<td>Could</td>
<td>Must</td>
<td>Formal-External</td>
<td>Notes/ClearQuest?</td>
</tr>
<tr>
<td>Configuration Management Plan</td>
<td>Must</td>
<td>Must</td>
<td>Formal-External</td>
<td>Word</td>
</tr>
<tr>
<td>Project Repository</td>
<td>Should</td>
<td>Must</td>
<td>Informal</td>
<td>? (CVS, ...)</td>
</tr>
<tr>
<td>Workspace</td>
<td>Should</td>
<td>Must</td>
<td>Informal</td>
<td>? (CVS, ...)</td>
</tr>
</tbody>
</table>
Notes on the Artifacts

Reports

Notes on the Reports

Additional Review Procedures

Other Issues

Configuring the Discipline

Project Management

A description of how the Project Management discipline is to be employed in the project.

Activity Overview

Artifacts

TBD. None of the artifacts listed here are decided upon yet. They will however probably used (most of them).

<table>
<thead>
<tr>
<th>Artifacts</th>
<th>How to Use</th>
<th>Review Details</th>
<th>Templates/Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iteration Assessment</td>
<td>Must</td>
<td>Informal</td>
<td>Word</td>
</tr>
<tr>
<td>Iteration Plan</td>
<td>Must</td>
<td>Formal-Internal</td>
<td>Word</td>
</tr>
<tr>
<td>Review Record</td>
<td>Must</td>
<td>Formal-Internal</td>
<td>Word</td>
</tr>
<tr>
<td>Risk List</td>
<td>Must</td>
<td>Formal-Internal</td>
<td>Requisite Pro</td>
</tr>
<tr>
<td>Risk Management Plan</td>
<td>Could</td>
<td>Formal-Internal</td>
<td>Word</td>
</tr>
<tr>
<td>Project Charter (PCM)</td>
<td>Must</td>
<td>Formal-External</td>
<td>Word</td>
</tr>
</tbody>
</table>

Notes on the Artifacts

The following artifacts are **not** used:

<table>
<thead>
<tr>
<th>Artifacts</th>
<th>How to Use</th>
<th>Reason</th>
</tr>
</thead>
<tbody>
<tr>
<td>Measurement Plan</td>
<td>Won’t</td>
<td>Ideally there should be a measurement plan on organization level to ensure that all projects are</td>
</tr>
</tbody>
</table>
collecting the same metrics. This does not exist and therefore we don’t expect the projects to produce a measurement plan either but we want them to do measurements.

Problem Resolution Plan
Won’t

Work Order
Won’t

Software Development Plan
Won’t

This should be covered in the project charter.

This level of ceremony is not normally used at Volvo IT.

Replaced by PCM Project Charter

Reports

Notes on the Reports

Additional Review Procedures

Other Issues

Configuring the Discipline

Environment

A description of how the Environment discipline is to be employed in the project.

Activity Overview

Artifacts

TBD. None of the artifacts listed here are decided upon yet. They will however probably used (most of them).

<table>
<thead>
<tr>
<th>Artifacts</th>
<th>How to Use</th>
<th>Incep</th>
<th>Elab</th>
<th>Const</th>
<th>Trans</th>
<th>Review Details</th>
<th>Tools Used</th>
<th>Templates/ Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Design Guidelines</td>
<td>Could</td>
<td>Should</td>
<td>Should</td>
<td>Should</td>
<td>Informal</td>
<td>Informal</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Development Case (This Document)</td>
<td>Must</td>
<td>Must</td>
<td>Must</td>
<td>Must</td>
<td>Formal-Internal</td>
<td>Word</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Development Infrastructure</td>
<td>Sholud</td>
<td>Must</td>
<td>Must</td>
<td>Must</td>
<td>Informal</td>
<td>Informal</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Development-Organization Assessment</td>
<td>Could</td>
<td>Could</td>
<td>Could</td>
<td>Could</td>
<td>Informal</td>
<td>Informal</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Notes on the Artifacts

The following artifacts are **not** used:

<table>
<thead>
<tr>
<th>Artifacts</th>
<th>How to Use</th>
<th>Reason</th>
</tr>
</thead>
<tbody>
<tr>
<td>Business Modeling Guidelines</td>
<td>Won’t</td>
<td>Business Modeling is done using CAMP.</td>
</tr>
<tr>
<td>Manual Style guide</td>
<td>Won’t</td>
<td>Ideally there should be a Manual Style guide on organization level to ensure that all projects writing manuals the same way. This does not exist and therefore we don’t expect the projects to produce a Manual Style guide either.</td>
</tr>
<tr>
<td>Project Specific Templates</td>
<td>Won’t</td>
<td>It is most often enough with the templates distributed in the RUP Project Kit (RPK).</td>
</tr>
<tr>
<td>Test Guidelines</td>
<td>Won’t</td>
<td>Ideally there should be a Test Guidelines on organization level to ensure that all projects tests the same way. This does not exist and therefore we don’t expect the projects to produce a Test Guidelines either.</td>
</tr>
<tr>
<td>Tool Guidelines</td>
<td>Won’t</td>
<td>There are tool mentors in RUP for usage of a lot of tools. We do not expect the projects to write any tool guidelines.</td>
</tr>
<tr>
<td>Use-Case Modeling Guidelines</td>
<td>Won’t</td>
<td>Use-Case Modeling Guidelines are documented at organization level. These guidelines should be followed and only in exceptional cases adopted.</td>
</tr>
<tr>
<td>User Interface Guidelines</td>
<td>Won’t</td>
<td>Ideally there should be a User Interface Guidelines on organization level to ensure that all projects using design user interfaces the same way. This does not exist and therefore we don’t expect the projects to produce a User Interface Guidelines either.</td>
</tr>
</tbody>
</table>
Reports

Notes on the Reports

Additional Review Procedures

Other Issues

Configuring the Discipline
Roles

RUP defines 31 roles. To avoid having to assign staff to all these roles this development case defines a higher level of roles that should be used for staffing.

<table>
<thead>
<tr>
<th>Role</th>
<th>RUP's Roles</th>
</tr>
</thead>
<tbody>
<tr>
<td>Analyst</td>
<td>System Analyst</td>
</tr>
<tr>
<td></td>
<td>Business Designer</td>
</tr>
<tr>
<td></td>
<td>Requirements Specifier</td>
</tr>
<tr>
<td></td>
<td>Use-Interface Designer</td>
</tr>
<tr>
<td>Developer</td>
<td>Designer</td>
</tr>
<tr>
<td></td>
<td>Integrator</td>
</tr>
<tr>
<td></td>
<td>Implementer</td>
</tr>
<tr>
<td></td>
<td>Database Designer</td>
</tr>
<tr>
<td></td>
<td>Code Reviewer</td>
</tr>
<tr>
<td></td>
<td>System Administrator</td>
</tr>
<tr>
<td></td>
<td>Tool Specialist</td>
</tr>
<tr>
<td>Architect</td>
<td>Software Architect</td>
</tr>
<tr>
<td>Tester</td>
<td>Test Designer</td>
</tr>
<tr>
<td>CM</td>
<td>Configuration Manager</td>
</tr>
<tr>
<td>Project Manager</td>
<td>Project Manager</td>
</tr>
<tr>
<td></td>
<td>Change Control Manager</td>
</tr>
<tr>
<td></td>
<td>Deployment Manager</td>
</tr>
<tr>
<td>Process Engineer</td>
<td>Process Engineer</td>
</tr>
</tbody>
</table>

The following reviewers are needed:

- Requirements Reviewer
- Design Reviewer
- Architecture Reviewer
- Project Reviewer

The following roles are **not** covered:

<table>
<thead>
<tr>
<th>Artifacts</th>
<th>Reason</th>
</tr>
</thead>
<tbody>
<tr>
<td>Business Process Analysts</td>
<td>Business Modeling is done using CAMP.</td>
</tr>
<tr>
<td>Role</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Business Model Reviewer</td>
<td>Business Modeling is done using CAMP.</td>
</tr>
<tr>
<td>Capsule Designer</td>
<td>Only for real-time systems.</td>
</tr>
<tr>
<td>Any Role</td>
<td></td>
</tr>
<tr>
<td>Course Developer</td>
<td>Use if applicable.</td>
</tr>
<tr>
<td>Graphic Artist</td>
<td>Use if applicable.</td>
</tr>
<tr>
<td>Stakeholder</td>
<td>Of course there are stakeholders. They are described in the Vision document.</td>
</tr>
<tr>
<td>Technical Writer</td>
<td>Use if applicable.</td>
</tr>
</tbody>
</table>
Appendix A

Sample Iteration Plans

Because of the different focus of the different phases of RUP there are one sample iteration plan per phase described below. In the diagrams below we show only internal milestones of the iteration and the major artifacts to be “ready” at the respective milestones. The diagrams are taken from a Microsoft Project plan. We recommend using Project for viewing all the details of the iteration plans. It includes purpose of milestones, workflow details to be performed between the milestones etc.

Inception Phase

The inception phase will only have one iteration. This iteration is a preliminary iteration i.e. it does not result in an executable release. The focus during this iteration is on project management and requirements. Below the discipline for that iteration is described.

Sample Iteration Plan

This illustration shows how a project begins, and how the various workflows relate. It is constructed from the Workflow Details as they would appear at the time of the first iteration of the project.
Figure 1: sample_iteration_inception v1.2

Elaboration Phase

The elaboration phase may consist of multiple iterations. These iterations are real iterations according to the definition in RUP, i.e. they produce an executable release.

The focus during the elaboration phase is on analyzing the domain and reducing risks by producing a stable executable architecture documented in the Software Architecture Document.

Below the discipline is described. The placeholders “Run Configuration and Change Control” and “Implement and Test” are described as separate diagrams later in this section.

Sample Iteration Plan

This illustration shows the relationship of the workflows in an early elaboration iteration. It is constructed from the Workflow Details as they would appear at that time.
**Construction Phase**

The construction phase consists of multiple iterations all resulting in an executable release. The focus during the construction phase is on completing the system by adding functionality. In the discipline is described below the placeholders “Run Configuration and Change Control” and “Implement and Test” are used to represent a chunk of activities. These placeholders are described in the “Elaboration Iteration”-section.

**Sample Iteration Plan**

This illustration shows the relationship of the workflows in an early construction iteration. It is constructed from the Workflow Details as they would appear at that time.
<table>
<thead>
<tr>
<th>ID</th>
<th>Task Name</th>
<th>Mon 2</th>
<th>Mon 3</th>
<th>Mon 4</th>
<th>Mon 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Iteration Milestones</td>
<td></td>
<td>01-08</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Iteration Start</td>
<td>01-08</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Iteration Plan</td>
<td>01-08</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Risk List</td>
<td>01-08</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Design Start</td>
<td></td>
<td>01-15</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Analysis Model</td>
<td></td>
<td>01-15</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Test Start</td>
<td></td>
<td></td>
<td></td>
<td>01-31</td>
</tr>
<tr>
<td>8</td>
<td>Test Model</td>
<td></td>
<td></td>
<td></td>
<td>01-31</td>
</tr>
<tr>
<td>9</td>
<td>Implementation Model</td>
<td></td>
<td></td>
<td></td>
<td>01-31</td>
</tr>
<tr>
<td>10</td>
<td>Use-Case Model</td>
<td></td>
<td></td>
<td></td>
<td>01-31</td>
</tr>
<tr>
<td>11</td>
<td>Test Results</td>
<td></td>
<td></td>
<td></td>
<td>01-31</td>
</tr>
<tr>
<td>12</td>
<td>Review Records</td>
<td></td>
<td></td>
<td></td>
<td>01-31</td>
</tr>
<tr>
<td>13</td>
<td>Iteration Assessment</td>
<td></td>
<td></td>
<td></td>
<td>01-31</td>
</tr>
</tbody>
</table>

Figure 3: sample_iteration_construction v1.0