HELSINKI UNIVERSITY OF TECHNOLOGY
DEPARTMENT OF ELECTRICAL AND COMMUNICATIONS ENGINEERING

Henri Elemo

DEFINING SOFTWARE CONFIGURATION MANAGEMENT FOR PRODUCT DEVELOPMENT

Master's Thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in Engineering

Espoo, May 30, 2008

Supervisor of the Thesis: Professor Timo O. Korhonen

Instructor of the Thesis: M.Sc. Jarkko Jussila

HELSINKI UNIVERSITY OF TECHNOLOGY **ABSTRACT OF THE MASTER'S THESIS**

Author:	Henri Elemo
Name of the Thesis:	Defining Software Configuration Management for Product Development
Date:	May 30, 2008 Number of pages: 11+79+10
Department:	Department of Electrical and Communications Engineering
Professorship:	Communications Code: S-72
Supervisor:	Timo O. Korhonen, Professor
Instructor:	Jarkko Jussila, M.Sc.

Normally, the usage model for Software Configuration Management (SCM) used in software development projects has been chosen by making a individual feasibility studies for each project. This way of making decisions is accurate, but it requires plenty of time and resources. There is a need to ease the selection of a SCM usage model to increase speed and profitability of decision making process.

The object of this thesis is to discover the factors affecting the software configuration management process in the SW development environment and create question lists that can be exploited when investigating the characteristics of SW development. In addition, we have studied how different version control systems and software configuration management usage models used inside Nokia Siemens Networks handle these different requirements for SCM. When combining these two outcomes we have a good basis for designing new software configuration management processes for new or existing software development projects.

When comparing different SCM processes, we have studied the strengths, disadvantages and similarities of each solution. This is, due to the fact that different software configuration management tools and processes have characteristics that can be applied better to one software development context than in another.

Keywords:	SCM, SCCM, VCS, SCM Usage Model, SW Development

TEKNILLINEN KORKEAKOULU **DIPLOMITYÖN TIIVISTELMÄ**

Tekijä:	Henri Elemo
Työn nimi:	Ohjelmistokonfiguraation hallinnan määritteleminen tuotekehitykselle
Päivämäärä:	30.05.2008 Sivumäärä: 11+79+10
Osasto:	Sähkö- ja tietoliikennetekniikan osasto
Professuuri:	Tietoliikennetekniikka Koodi: S-72
Työn valvoja:	Professori Timo O. Korhonen
Työn ohjaaja:	Diplomi-insinööri Jarkko Jussila

Yleensä ohjelmistokonfiguraation hallinnan käyttömalli ohjelmistokehitysprojektille on valittu tekemällä ainutkertainen soveltuvuustutkimus jokaiselle ohjelmistokehitysprojektille erikseen. Tällainen päätöksentekoprosessi on tarkka, mutta se vaatii paljon aikaa sekä resursseja. Onkin tarve pystyä helpottamaan ohjelmistokonfiguraation hallinnan käyttömallin valintaa, jotta päätöksentekoprosessista saadaan nopeampi ja kannattavampi.

Tämän diplomityön tavoite on löytää tekijät, jotka vaikuttavat ohjelmistokonfiguraation hallinta prosessiin ohjelmistokehitysympäristössä, sekä luoda kysymyslista jota pystytään hyödyntämään ohjelmistokehitysympäristön ominaisuuksien tutkimisessa. Lisäksi tutkitaan miten eri Nokia Siemens Networksillä käytettävät versionhallintajärjestelmät ja ohjelmistokonfiguraation hallinnan käyttömallit tukevat löydettyjä vaatimuksia. Yhdistämällä nämä kaksi tutkimustulosta pystytään saamaan hyvä pohja kun suunnitellaan ohjelmistokonfiguraation hallinnan prosesseja uusille tai jo olemassa oleville ohjelmistokehitysprojekteille.

Tutkittaessa eri ohjelmistokonfiguraation hallinnan prosesseja keskityttiin jokaisen prosessin vahvuuksiin, heikkouksiin, sekä samankaltaisuuksiin. Tämä sen takia, koska ohjelmistokonfiguraation hallinnan työkalut, sekä prosessit sisältävät ominaisuuksia, jotka soveltuvat paremmin eri ohjelmistokehitysympäristöihin kuin toiset.

Avainsanat: SCM, SCCM, Versionhallintajärjestelmä, SCM Käyttömalli, Tuotekehitys

Acknowledgements

This thesis work was carried out in the Global SCM Service department at Nokia Siemens Networks Oy in Finland.

I wish to express my gratitude to the people who have supported me in this work. First, I want to thank Professor Timo O. Korhonen for supervising this thesis and his valuable advice. In addition, I am grateful to Keijo Kähärä and Jarkko Jussila for providing me with the opportunity to write this thesis. Furthermore, thank Jarkko Jussila for all the valuable advice at all stages of the work and being the instructor of this thesis. I would also like to thank Eero Koskimaa, Tomi Kahila, Lauri Kiviniitty for familiarizing me with different version control systems and the rest of the SCM service team for their general feedback. I also wish to thank William Martin for revising the language of this thesis. I want to give special thanks for my colleges Olli Järvinen and Jyrki Karhu for providing the time for me to make this thesis.

Finally, I am grateful to my dear parents Kirsti and Asko, fiancée Kirsi Koskenheimo, and friends Petra Säntti, Minh-Hai Nguyen and Kimmo Yrjölä for their support during my studies.

Helsinki, May 30, 2008

Henri Elemo

Table of Contents

Figures and Tables

Key Concepts

Baseline

The software baseline refers to all source code, run time files, documentation, configuration files, and installation scripts that comprise a software release, both input and output of the build management process. The baseline must include any data or database related scripts or files that were used in the creation of the release. (Afora 2008)

Branch

Branching in version control system enables copying items in another path to maintain independent changes to the configuration items. (Afora 2008)

Change Management:

Change management identifies the procedures used to process changes to known baselines, including activities to control of changes to a configuration item after the formal establishment of its configuration documents. (Kääriäinen 2006)

Merge

Merging is an operation where the content of branch is added to another branch inside the version control system.

Multisite Development

In software development, multisite projects stands for a situation where the development organization is geographically distributed to more than one development site. (Armstrong & Cole, 2002; Herbsleb et al. 2005)

Repository

Repository is a database where software items are stored. Content of repositories are managed with the Version Control System (VCS) in SCM. (Tichy 1982)

Software Configuration Management (SCM):

Software configuration management (SCM) is a service function in software development which focuses on managing the evolution of software systems. SCM practices include version control for software items and the establishment of software baselines. (Conradi & Westfechtel 1998)

Software Project

Software project includes all the parts needed to produce SW project including people, processes, tools and the content of SW product.

SCM Usage Model

Usage models describe the fundamentals SCM process is based on. These models define how different SCM tasks are performed in software project and it describes the main SCM principles. Usage models define the identification schemes and file systems used in SW project.

Version Control System (VCS)

The basic function of VCS is to manage revisions of text-files, for example, software code and documentation. (Tichy 1982)

Research Questions

What does a Software Development Project require from Nokia Siemens Networks SCM Service Department?

The software development project is the main interface for the SCM project. This is because SCM acts as a service function for the software development. This study researches the requirements from this interface.

How Can the SCM Service Department Support the Development of a Software Product?

SCM is an important part of the software development process from the beginning to the end of the whole SW project. This is why the SCM service department must seek ways to support the development of the SW product.

How do Existing SCM Version Control Systems serve Software Development?

The version control system is a crucial tool in the work of the SCM and there is a need to investigate how different version control systems used inside Nokia Siemens Networks support different software development situations.

Abbreviations

ALM:	Application Lifecycle Management
ANSI:	American National Standards Institute
CI:	Continuous Integration
CM:	Configuration Management
DC:	Development Center
GUI:	Graphical User Interface
HTTP(S):	Hypertext Transfer Protocol Secure
HW:	Hardware
I&V:	Integration and Verification
IP:	Internet Protocol
IBM:	International Business Machines
IEEE:	Institute of Electrical and Electronics Engineers
ISO:	International Organization for Standardization
NSN:	Nokia Siemens Networks
R&D:	Research and Development
SCCM:	Software Change and Configuration Management
SCM:	Software Configuration Management
SLOC:	Source Lines of Code
SRC:	Source Code
SSH:	Secure Shell
SVN:	Subversion
SW:	Software
UCM:	Unified Change Management
VCS:	Version Control System
VPN:	Virtual Private Network
VOB:	Version Object base
VTT:	Technical Research Centre of Finland

1. Introduction

The normal Software Configuration Management (SCM) usage model for software (SW) development projects has been typically chosen by making individual feasibility studies for each SW project separately. This way of decision making is accurate, but it requires plenty of time and resources. There is a need however to ease the selection of an SCM usage model to improve the speed and profitability of the decision making process. This thesis finds out the factors affecting SCM solution in SW development environments and creates a question list that can be exploited when investigating the characteristics of SW development. In addition, we have studied how different Version Control Systems (VCS) and SCM usage models used inside Nokia Siemens Networks (NSN) handle these different requirements for SCM. More defined processes for determining SCM usage models will reduce the possibilities of making drawing conclusions when choosing the model for SW development projects. This way offers a better guarantee that the model chosen is the best possible one for the project without spending too much time and resources to do so.

This thesis studies the SCM field from different points of view and presents how new development methods like Agile and multisite development are taken into account in the VCSs and SCM usage models studied. The Agile development model, for example, has become very popular inside NSN and multisite development methods are important when development is distributed between different sites. These new software development methods create different requirements for the SCM process and tools that, for example, the traditional waterfall development model do not require.

SCM is a service function to manage this change and produce controlled software builds with correct content on them. This is important in SW development because software packages have become larger and more complex than ever. There is also great pressure to develop software more quickly for the markets without sacrificing the quality of software. In short, customers and suppliers want software to include more features with better quality standards within shorter releasing times. These requirements have pushed software development to evolve new working processes,

because each of these factors can lead to bad software. (Kokkonen 2003; Koskela 2003) To ensure that the changes do not end up in total chaos, they must be fully controlled. A simplified picture of SCM service is presented in Figure 1. Mainly, SCM is placed between SW developers and Integration & Verification (I&V). This is because SW developers produce software code that is used in SW packages and I&V verifies that the SW operates properly. The SCM promotes SW builds when SW release or correction packages are ready for customer use.

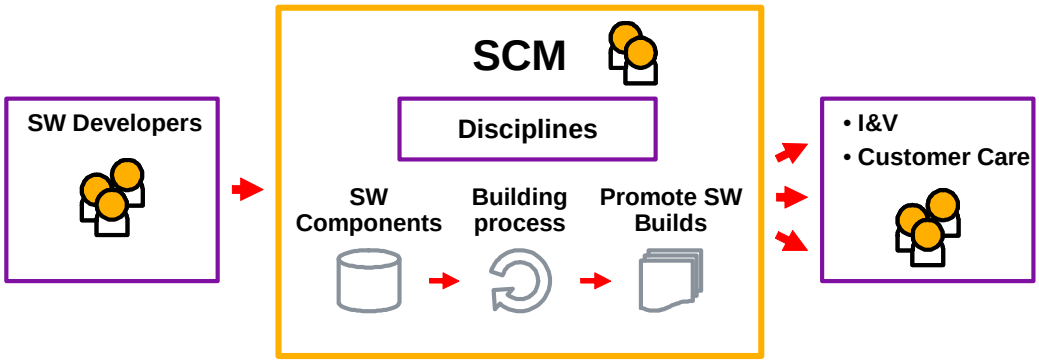

Figure 1: Simple description of SCM

From a tool point of view, the SCM process is usually wrapped around the VCS because it manages all the software components, versions and history information, and in that way, holds all the software evolution. With the help of the VCS, software components for a particular software build can be identified in a controlled way. (H. Chen & C. Sun, 2001) If a company wants to use the same kind of processes and methods in SCM for its SW products, there is a need to use the same VCS in all the SW products. Usage of only one VCS inside a company for all of the development projects can however create problems, because every product has its own special characteristics that are handled better in some tools than in others. In this case, “The Golden Mean” is important when considering the number of applied VCSs. There should be only a few VCSs in use, because information and resources can be shared in a more easy way, when people know the basics of their usage. But using more than one VCS and SCM process may give a competitive advantage because different tools and methods suits better to correct contexts. This is why we have not tried to find only one VCS and one SCM usage model which could be realized in every SW product, but instead studied the

strengths, disadvantages and similarities of each solution. This same principle could be applied to the usage models implemented inside a company.

1.1 Structure of the Research

This thesis is divided into two parts. The first part presents the theoretical background and the second part presents the study and results of the thesis. The second part and especially Chapter Nine, includes the most important details from research point of view. Finally conclusions based upon the results are made in the final chapter.

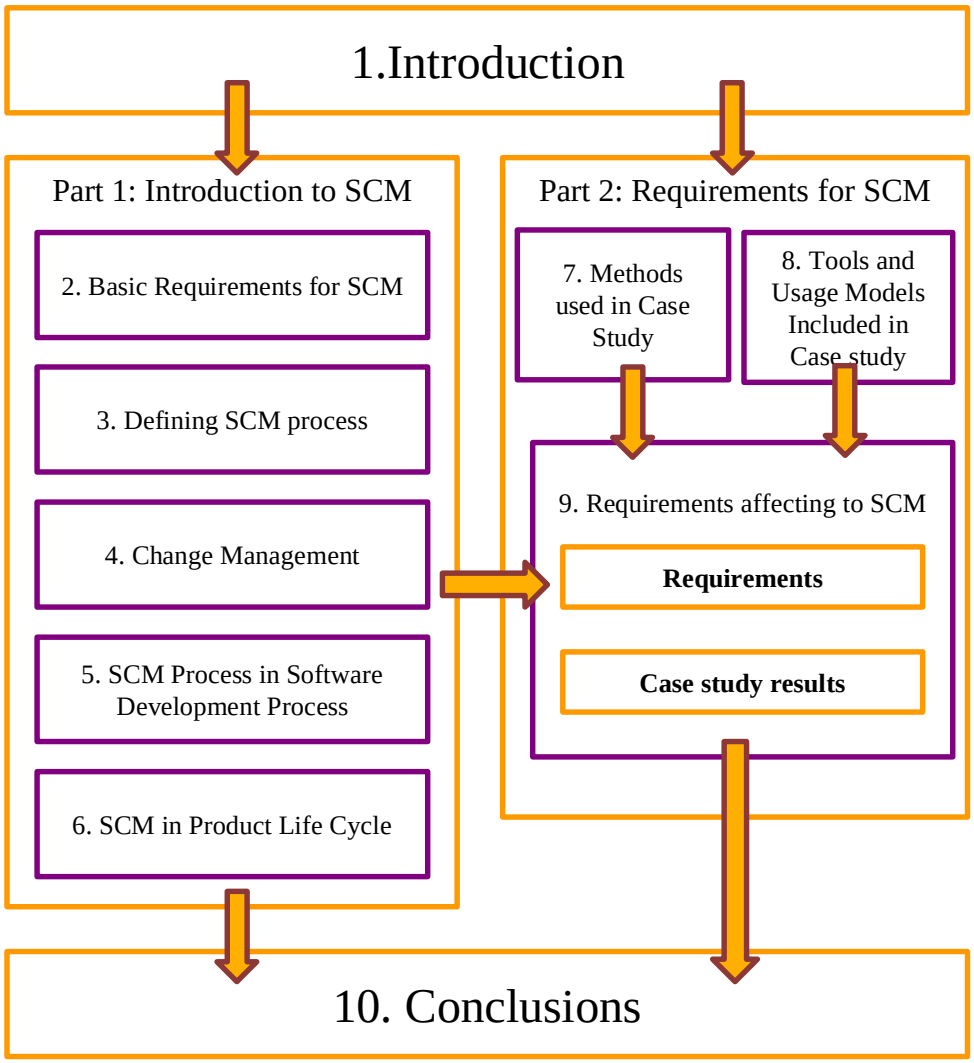

Figure 2: Structure of the Research

The first part begins by defining SCM and discusses how SCM usage models and processes. Once this has been accomplished, the change management process is

then introduced and discussed. There is also a description of how SCM is connected to SW product and development environment that it serves. The second part contains the most significant part of this work from a researcher's perspective. First of all, it presents how the study is realized and introduces the five VCSs and the four SCM usage models studied. Then main contribution of this thesis is presented in Chapter Nine "Requirements affecting to SCM", and a conclusion of the main findings is presented in the final chapter.

PART 1. INTRODUCTION TO SCM

SCM is a service function in software development which focuses on managing the evolution of software systems. It includes the tools and methods that a company uses to manage change in its software during the entire product life cycle. SCM manages the artifacts in the development process, controls the changes in the software and its components, and assists the software development. SCM can also be understood to be a discipline that enables an evolving software product to be under control (Conradi & Westfechtel 1998). SCM's task is also to produce data of the software's state for the rest of the software project. (Appleton 2006)

As defined in the literature (Conradi & Westfechtel 1998; ANSI/IEEE 1042-1987 1987), SCM focuses on managing the evolution of software systems and is a discipline that enables an evolving software product to be under control. Inside Nokia Siemens Networks SCM is defined as being a service function that goes beyond the traditional software configuration process. Here, the SCM service is composed of six different elements that are presented below.

- **SW Building** holds all the tasks that need to be taken into account when constructing SW builds, including tasks from the management of software configuration to the compilation of single SW components, distribution of SW builds for needed parties and content and quality evaluation of the SW build in question.

- **SCM Development** defines tasks to evolve the SCM service. There are many different ways to develop SCM solutions and their execution such as SW compilation times, quality of service and better methods for controlling software configuration.

- **SCM Consulting** means supporting SW developers in their use of the necessary SCM tools as well as participating in the architecture planning of SW product.

- **SCM Infrastructure Administration** holds all the tasks that lead to functional and well controlled SCM environment. Tasks relating to this are maintaining SCM tools and repositories, administering the content of VCS and supporting the usage of SCM tools.

- **SCM Project Management** includes planning and management related tasks that aim to maintain quality of the SCM work and inform other SW development parties of the SW's state. These tasks are, for example, planning of the SCM, SW integration and the SW builds schedule. Information sharing and reporting is also part of it.

- **Support for Collaboration** holds the support and steering of the SCM collaborators and their work as well as the SCM support for collaborator SW development.

In SW development, it is important to understand that even in small software projects it is essential to have real SCM processes to administer changes and software configuration. Software is easy to change and therefore, it can change very quickly. In the Agile development environment there is a need to update the SW build every time changes to the SW emerge, meaning that the update process can be running continuously. Here small software projects may also carry a significant amount of software configuration management load. In telecommunication systems, projects are usually quite large and, therefore, update activities can totally overwhelm manual configuration management procedures. That is why SCM processes needs to be optimized for projects usage and automated processes are required. (Tichy 1988) SCM has a tight bond to the product it serves, with different characteristics of software development like development models and product's life cycle being significant to the whole SCM process.

2. Basic Requirements for SCM

SCM processes generate software packages from developers' code to multiple parties and it is crucial that these processes function correctly. SCM needs to ensure safety, stability, control, auditability, reproducibility, traceability and scalability. When any SCM process is designed, these requirements needs to be noticed in all the tasks that SCM provides for its customer. Table 1 below clarifies what these terms mean from the point of view of the SCM. (IBM 2004)

Table 1: Basic requirements of Software Configuration Management (expanded from IBM 2004)

Safety	SCM must recover from situations that end up in failure and ensure that the system can only be used by the authorized personnel. Recovering from failure is very important in a continuous integration process where the SW build and testing is done in small time cycles. Failures in the SCM environment can delay other development functions receiving the needed feedback of the SW builds quality.
Stability	Stability stands for controlled workspaces. In these workspaces SW developers can make and store changes in the needed SW components and announce the needed changes to the SW build.
Control	The SCM needs to be in control of the software's configuration and have control of the user's workflows. This means that the SCM needs to ensure that the needed SW changes are included in the SW build and configuration of the SW build is as needed.
Auditability	The auditable process characterizes a situation where the SCM has information to monitor SW builds. The content of SW builds is the most important information that must be recorded, because the correct content of an individual SW build can be verified based on this data. It includes all the files, tools and their versions used in the SW building process.
Reproducibility	When considering the purpose of SCM, the most important task is to make software builds. Each build must be reproducible from the first build to the latest build. It does not matter whether the build has a baseline or not. All the builds needs to be reproduced, because there can be a situation where some bug needs to be tested to sort out the branches that includes that bug.
Traceability	The ability to identify and reconstruct particular versions of a software product, which makes it possible to track changes in the software's development history.
Scalability	The SCM needs to have an ability to support SW development projects from small to large as well as in a distributed development. It is important that the SCM process supports different changes in the development environment during the development life cycle.

All of the requirements presented in Table 1 relate to quality of service. In SCM it is important to have control of the whole SCM process to ensure integrity of SW builds and continuum of the SW development process from the preliminary planning and coding to implementing and verifying. Service must be assured when ever there are changes in the development environment.

3. Defining SCM process

The SCM process describes in detail how SCM is realized in SW projects. The SCM usage model is part of the SCM process and it presents a more general picture of the SCM process and describes the principles and tasks performed in SCM in any SW project. These two definitions are distinguished from each other by how precise they define SCM disciplines. A description of how the SCM usage model is included in the SCM process in Figure 3.

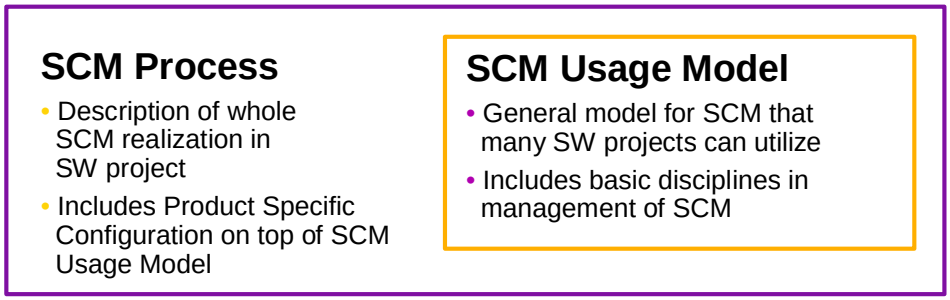

Figure 3: SCM process and SCM usage model

Every software product is a unique entity, each with its own content and configuration. The SCM solution is also unique because SCM is bound to an individual SW product. Even if all the SCM processes have similar same goals (like providing a stable VCS for product environment and making complete SW packages for testing and for customer care), every software product has its own special requirements and every SCM process needs to be modified to support these requirements. Definition of the SCM process takes into account all the characteristics that the SW product in question has and describes how the SCM will work to realize the SW product.

Even if SCM solutions are always unique entities, some of them use similar tools and methods to others for managing the SCM in SW product development. The SCM usage model is a definition that describes and gathers SCM solutions that are developed and realized with similar tools and methods. Even completely dissimilar products can use the same SCM usage model in their SW development. However, products that usually belong to the same product families use the same kind of solutions. The product family can use, for example, a common SW platform, the

same SW structure or even the same development methods. Advantages of these common methods are, for example, cost savings and competence growth because many of the SW projects can implement and exploit solutions that other SW projects have invented or developed. It also eases the ramping up of SCM processes for new SW projects. SCM usage models and SCM processes are defined more precisely in the next chapter.

SCM Usage Model

The SCM usage model is the same for multiple SW projects that use the same methods and tools to manage software configuration and components. It defines the steps and principles by which a specific task or set of tasks are performed. It does not, however, give answers to product questions. Tasks describe how, for example, new SW components are updated to the SW build, SW components are identified or what the structure is in the VCS. Specifically, it describes how an SCM tool is applied to accomplish a set of tasks. The Usage model can also define development lifecycles and how to manage changes to requirements, design models, documentation, components, test cases, and source code.

When reading the outcome of this thesis, it is important to remember that usage models are one way to manage certain software projects and there can be many ways to change and improve processes relating to these usage models. It is possible to change the usage model to manage SCM in a SW product as long as the product specific configuring is not changed. For example, software components can be checked from different VCSs with different version numbers as long as software components are managed in a way that the content and functionality of the SW build will not be affected. Changing usage models is, however, laborious.

SCM Process

As mentioned before, the SCM process is unique for every SW project and it describes the whole SCM solution in one SW project. It includes the SCM usage model and the configuration specific information related to the SW product. The SCM process describes a way to manage SCM in any SW development project and, as a whole, it is the way SCM is performed on a SW development project. It

defines both the methods and tools that are used in all SCM related tasks. The SCM plan is a document that describes all relevant SCM processes for one SW product. (ANSI/IEEE 1042-1987, 1987)

4. Change Management

Change management as a concept covers activities that controls SW changes in the content of the SW build. Modern Software Change and Configuration Management (SCCM) solutions like *UCM (Unified Change Management)* by IBM Rational and *Dimensions* by Serena enables versatile change management functionalities. By utilizing these functionalities the SCM service is able to supply tools and methods to control the life cycle of software change. The change management process identifies the procedures used to process changes to the known baselines, including activities for controlling changes to a configuration item after the formal establishment of its configuration documents. It contains all the elements for evaluation, decision, and implementation, as well as, for change requests. Its target is to ensure that every SW change is made in a controlled way and similar information is collected for each change. Figure 4 presents a high-level description of the change management process. (ANSI/IEEE 1042-1987 1987; Kääriäinen 2006; White 2001)

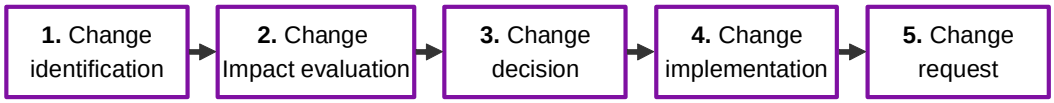

Figure 4: Change management process (modified from Kääriäinen 2006)

First the need for the change is identified in order to understand what is required to change. Second the need for change is evaluated to define how critical it is to make the change in question and how much work it will require. Third, the decision of making the change is made and after that it can be implemented (fourth stage). Finally when the required SW changes are made a request is made to the needed SW releases.

The SCCM tool *UCM* by IBM Rational, for example, provides control for this change management process and different factors to support it. Figure 5 describes the UCM process overview where there are five different actors in the process. The architect defines the implementation process. The configuration manager sets up the SCM environment. The project manager manages the content of the project, specifies the policies and supervises the project. The developer joins at specified

parts of the project that the project manager has created and makes changes to the SW code and delivers those changes. The integrator baselines components, creates SW builds and promotes baselines. Sections highlighted in the picture with orange can be under the responsibility of SCM Service. (White 2001)

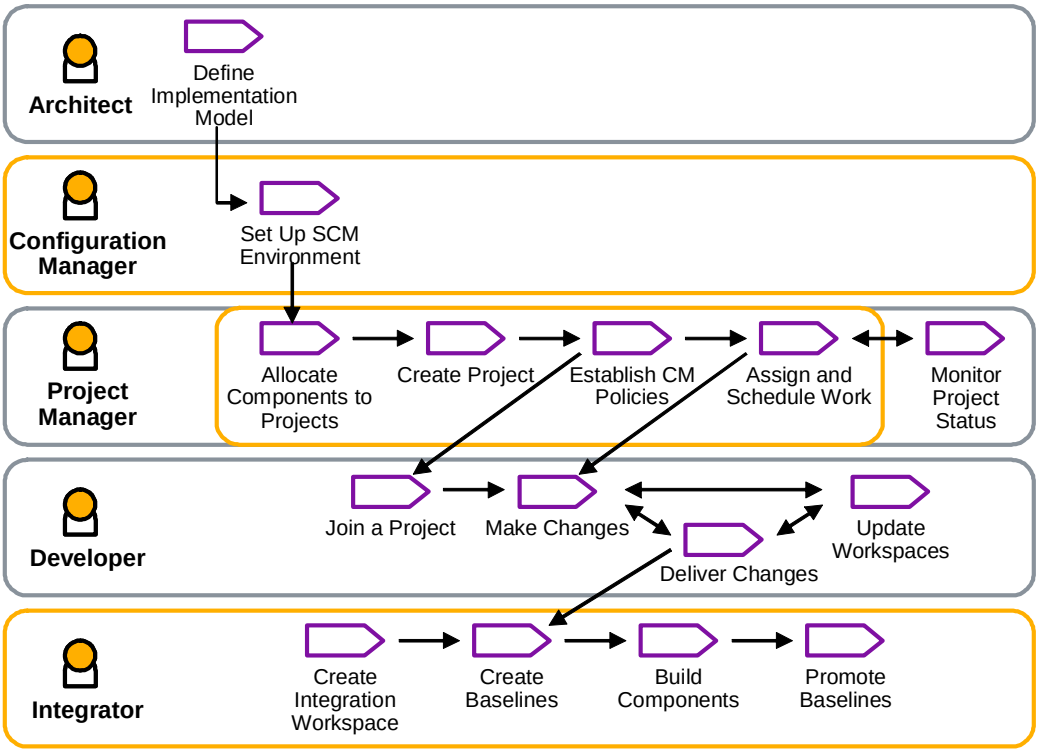

Figure 5: UCM process overview (White 2001)

This change process is unique for UCM, but the same kind of principles can be exploited with the other VCSs also. We must, however, always remember that different VCSs are designed from different points of view and used in different development processes. This means that these models must be always adapted to the specific process in question.

5. SCM Process in Software Development Process

The SCM is an important part of the software development process and it is needed from the beginning to the end of the SW project. The reason for this is that the SW configuration principles must be designed before any SW can be implemented and these principles must be maintained until to the end of the project.

IEEE's standard 1042-1987 "IEEE Guide to Software Configuration Management" (1987) defines SCM to be a support activity that makes technical and managerial activities more effective. Effectiveness of the SCM process increases in proportion to the degree that its disciplines are an explicit part of the normal day-to-day activities of everyone involved in the development and maintenance efforts. SCM provides a common point of integration for all planning, oversight and implementation activities for a project or product line.

SCM is a service function and must be adapted to the product it serves. There are many requirements that must be taken into account when planning a process for the SCM Service. The main things affecting are SW products' management specific models, such as, integrating SCM into the products life cycle and SW development model. These points are introduced in subsequent chapters and the second part of this thesis examines more deeply the different requirements that affect the SCM service in the SW development.

6. SCM in Product Life Cycle

SCM processes are designed to support all the parts of the SW development. That is why SCM is involved from the early start to the end of a product's life cycle. In the start of the product's life cycle, the SCM team plans the processes and the development models. These models define the product's development with the project management and other development functions like implementation, testing and documentation. (ANSI/IEEE 1042-1987 1987) SCM has also an important role when tools are chosen for the development environment. This is because the same tools need to support the whole software configuration functionality and serve all the other functions in the development project. Developers, for example, need to have access to the VCS and be able to include new software components to the software packet in a controlled way. Also, the software packet needs to be available for testing for software I&V. The SCM service can also have direct linkage to other functions like customer care and SW project management. Different linkages give special requirements for SCM and needs to be designed to work with other parts of SCM.

The software's lifecycle can be divided into two parts. The first one is a software development project and the second one is a software maintenance project. The development and maintenance projects can run simultaneously after a software release is in state to be published in the markets. After that, the development project can develop the software product further and the maintenance project implements bug fixes and possible new features on the top of the software product's published releases. (Rajlich & Bennett 2000) The maintenance project can produce the bug fixed or a have miniature waterfall model inside that provides some new features. Figure 6 overleaf describes essentially how this is realized in the waterfall model.

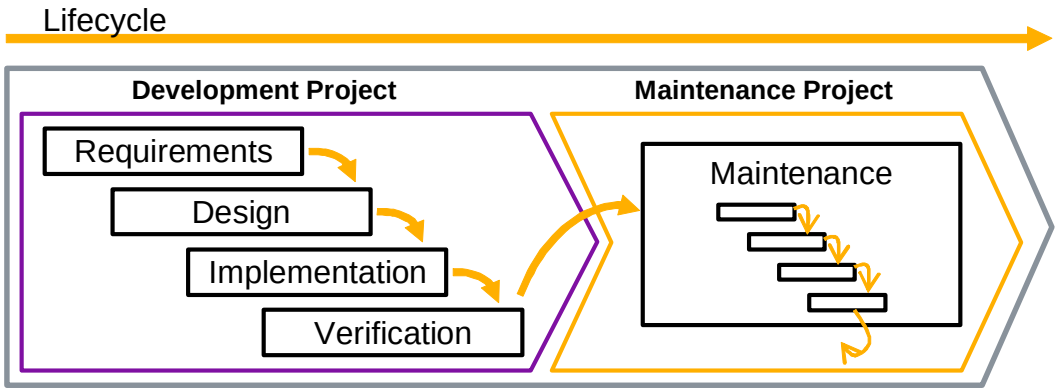

Figure 6: Development and maintenance in SW development life cycle

In the SW development project, the target is to produce a working software product ready for the markets. The goal is to add new features and correct existing faults to SW without adding new faults. The main role for SCM is to provide software builds for the development project. The builds are made in defined time cycles like in the Waterfall development model (Cheatham & Crenshaw 1991) or when the SW changes occur in the software in the Agile development model (SVN-handbook 2004). It is important that a time cycle between builds is short enough because then different changes can be verified separately and a new functionality can be delivered for other developers.

In the software maintenance project, the goal is to offer software corrections that have emerged from software faults from customers that are using that software version where the fault has been detected. There can be also new features added to the release in the maintenance project. The role of SCM is to manage changed software components by making new software versions to produce a change delivery and promote it to the testing and after the testing to the customer care. In cases of critical faults, penetration time from the error detection at the customer site to fixing the problem at the customer site must be as small as possible. Regardless of development and maintenance projects, different approaches to the SW change requirements for SCM are quite common. In both cases there must be SW builds produced and their contents verified.

6.1 SCM in SW Development Process

When reading this study, it is important to understand the role of SCM in the software development process and the interfaces to the other functions in the development of the software. The SCM is directly between the implementation of software code and testing, but it also has linkages to the other parts in the software development process. Figure 7 below describes this role in the software development process according to the waterfall development model introduced in Section 6.2.

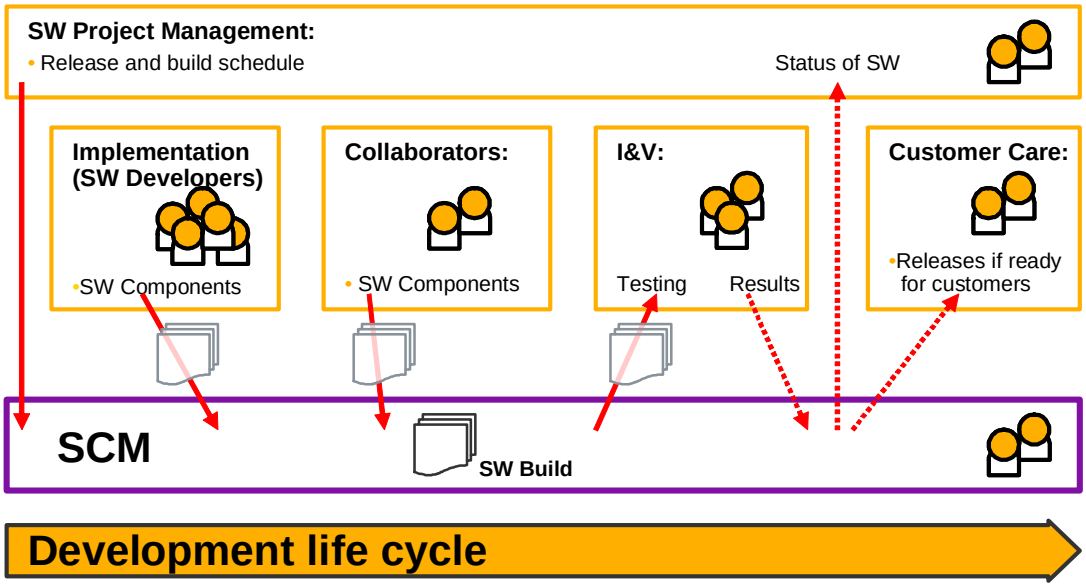

Figure 7: SCM Service in development process

The SW project management determines both release and build schedules for the SW development. The SCM service provides the SW build where the software changes can be made by SW developers and possible collaborators. When the scheduled time for a SW build comes, the SCM service freezes the content of the SW build and assembles it. After the SW build has been created successfully, it is promoted by SCM service to I&V for testing purposes. Additionally, the SCM service can make different kinds of checks for the SW baselines, for example, content checks, code line and complexity calculations. The SCM service informs the status of the software to the SW project management. When the SW release is ready to be published for the customers, the SCM service promotes these SW builds to the customer care.

6.2 SCM in SW Development Models

What is the current way to develop software? In this thesis we concentrate on two different development models. These are the Waterfall model which is straightforward and the Agile model which is an iterative way to develop SW products.

Today's iterative software development using Agile software development models, create more demand for performance, because the SW builds are required to be generated directly after the software changes have occurred. This can mean that the building process is running continuously. There is a need for fully automated SCM process which supports everything from the software implementations SW change to the testing of the software. In the product development phase, many development models bring special requirements for SCM processes. From the figure below, the main differences between the Waterfall and Agile models lifecycles can be seen, from the SCM and SW development points of view. More precise descriptions are presented in the next sub-sections.

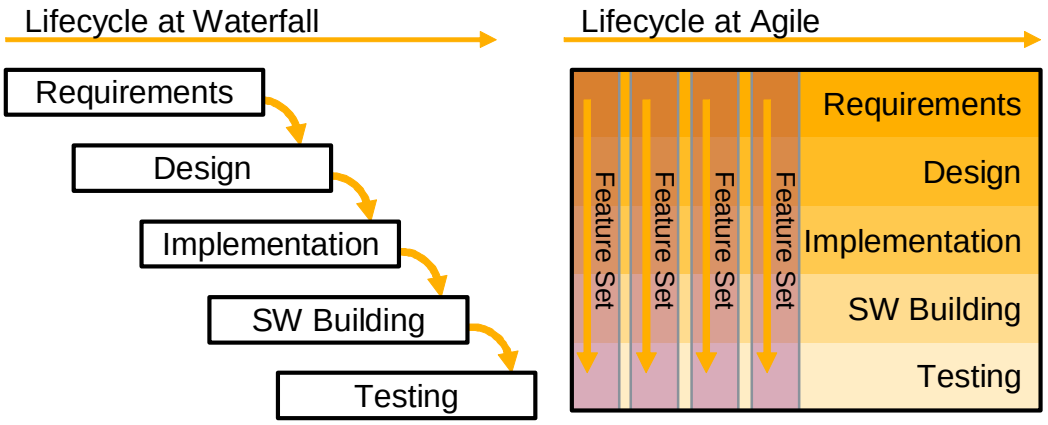

Figure 8: Differences between Waterfall and Agile models (modified from Appleton 2004: TechRepublic 2006)

Waterfall Model

The Waterfall model is a software development model that has been developed in the year 1970. It is maybe the most frequently utilized software development model in the world and it is a traditional model in software development. The Waterfall model includes a phased progression of activities leading up to the release and maintenance of the SW product. It is a sequential software development model in which development is seen as flowing steadily downwards (like a waterfall) through the phases of requirements analysis, design, implementation, verification, testing and maintenance. (TechRepublic 2006) The different phases of the waterfall model are described in Figure 9 below.

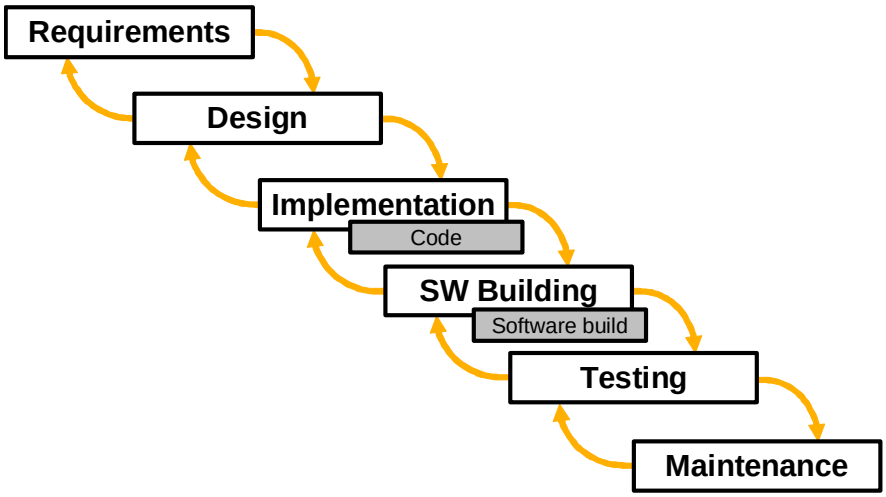

Figure 9: Waterfall model (modified from Royce, 1970; Answers, 2008)

The first and the most important phase is where the requirements for SW products are analyzed. In this phase, understanding the customer business context and constraints, the functions the product must perform, the performance levels it must be capable of, and the external systems it must be compatible with. This first phase is the most important one in Waterfall development, because every mistake made in this phase is hard and expensive to repair in later phases. This is because in order to identify the problem these phases must be followed. The second phase is where the content of SW is designed. In third phase, software code is implemented. In the fourth, phase SCM verifies and builds the software for testing that is done in phase five. After the SW is declared ready for customers, it is published and maintenance of the SW product starts. (TechRepublic 2006)

Agile Software Development

There are new iterative development models which differ substantially from the Waterfall development model described above. This creates a need for a new way of thinking from the SCM point of view. The Agile development model is an iterative development model which has been utilized already inside NSN. These iterative development models try to overcome the single main drawback of the Waterfall model, which is its inflexibility. Most Agile methods attempt to minimize risk by developing software in short time boxes, called iterations, which typically last from one to four weeks. Each iteration is like a miniature software project of its own, and includes all the tasks necessary to release the mini-increment of the new functionality: planning, requirements analysis, design, coding, testing, and documentation. (Koskela 2003; Appleton 2004) This is presented in Figure 10. While iteration may not add enough functionality to warrant releasing the product, an Agile software project aims to be capable of releasing new software at the end of each iteration phase.

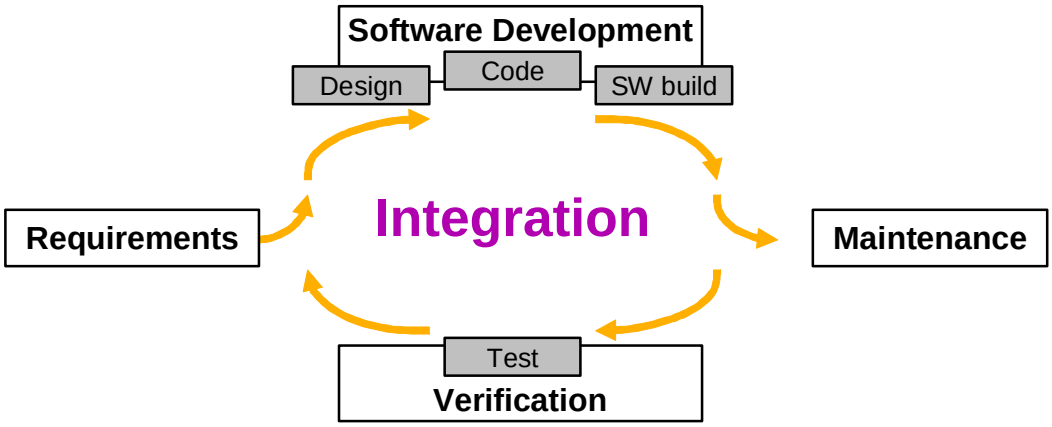

Figure 10: Integration Cycle in Software Development (Unboxed, 2008)

The Agile model makes more flexible development possible. This is because every iteration differentiates changes from each other and every feature can be verified separately. Additionally, it makes possible to change and plan requirements to the software in the middle of the development process.

PART 2. REQUIREMENTS FOR SCM

The second part of this thesis holds the most important content from a strictly research point of view. Firstly the methods used and the research context are presented in Chapters Seven and Eight. Chapter Nine continues by introducing the requirements found, as well as including results of how different VCSs used inside NSN support these requirements. The final chapter of this thesis, “Conclusions” summarizes the outcomes of the work.

We concentrate here on studying the VCSs and review those inside the SCM usage models, because the process used influences how these tools are implemented. The VCS and the SCM usage model are compulsory building blocks in the development environment that have a strong impact on the development disciplines. The VCS impacts directly on how versions are versioned and how new versions can be attached to the SW build. The SCM usage model affects the SW configuration control, for example, on how SW configuration is managed on new releases and what is the level of control when developers make changes to new software components in the SW builds.

7. Methods used in Case Study

The research was started by going over the different requirements for the SCM Service in the SW development. The first step was to study the SCM requirements and usage models presented in the literature and in discussions with SCM professionals inside NSN.

In the literature, Jukka Kääriäinen has studied factors affecting the configuration management solution in this VTT (Technical Research Centre of Finland) publication. The results of this study are included in Appendix 1. Additionally, the standardization organization, IEEE (Institute of Electrical and Electronics Engineers), has introduced the factors concerning SCM from the SCM planning point of view in standard 1042-1987. These different factors of SCM plans are introduced in appendix 2.

After gathering the requirements for SCM, a question list was made based on the requirements found. These different questions examine the characteristics of version control systems and are addressed to SW development organizations. The question lists was talked through with SCM professionals who have worked with the examined tools and methods inside NSN. Before the discussion sessions, the tools were introduced to provide a better understanding of the background behind the answers and to create a two-way dialogue on different topics. The results of the discussions and the question list have been compiled into a table that is included in Appendix 4. The requirements found as well as the results for the list of questions are revealed inside Chapter Nine “Requirements affecting SCM”.

8. Tools and Usage Models Included in Case study

This chapter introduces the different VCSs that are studied in this thesis. The VCSs were selected because these tools are widely used both inside and outside of NSN. Different types of VCSs have been selected for this case study in order to understand the differences between tools that have been developed with different viewpoints in mind. For the purpose of this classification, these differences are commercialism, size as well as multi-functionality of these tools.

VCS

VCS is an important tool in SCM because it serves some of the most important functions in the SCM process. The basic function of VCS is to manage revisions of text, for example, software code and documentation. Revisions are made manually by changing this text, for example, with any text editor. The VCS stores these files in a repository, logs changes, identifies revisions, merges revisions, and controls access to files in VCS. (Tichy 1982).

The VCSs evaluated in this thesis offer different amounts of functionalities that can be utilized in SW development environments. Subversion and CVS offer only VCS functionality, whereas SCCM tools *UCM* by IBM Rational and *Dimensions* by Serena offer a set of different features across different development teams. Base Clearcase offers a great set of different features for development environments, but does not share the same philosophy of “Do it all with one software” which is behind UCM and Dimensions tools.

Usage Models over VCSs

In this thesis, VCSs are evaluated inside the development process. Different usage models described in subsequent chapters reflect the fundamentals that the SCM process is based on. There are four different models under study which are used on the top of different VCSs, some which offer certain advantages and basics that direct the use of certain VCSs. For example, UCM is a commercial usage model that is fully designed to be used with Clearcase. When there is a need to reuse one SW component in other SW projects, it is complicated when there are different VCSs in use. This is, because different VCSs use different identification schemes

and file systems. Reuse of SW is a basic assumption in SCM usage model that uses Subversion as a VCS. The SCM usage models studied in this thesis are three in-house usage models used with different VCSs (*Clearcase*, *Subversion* and *Dimensions*) and one commercial usage model UCM by IBM Rational.

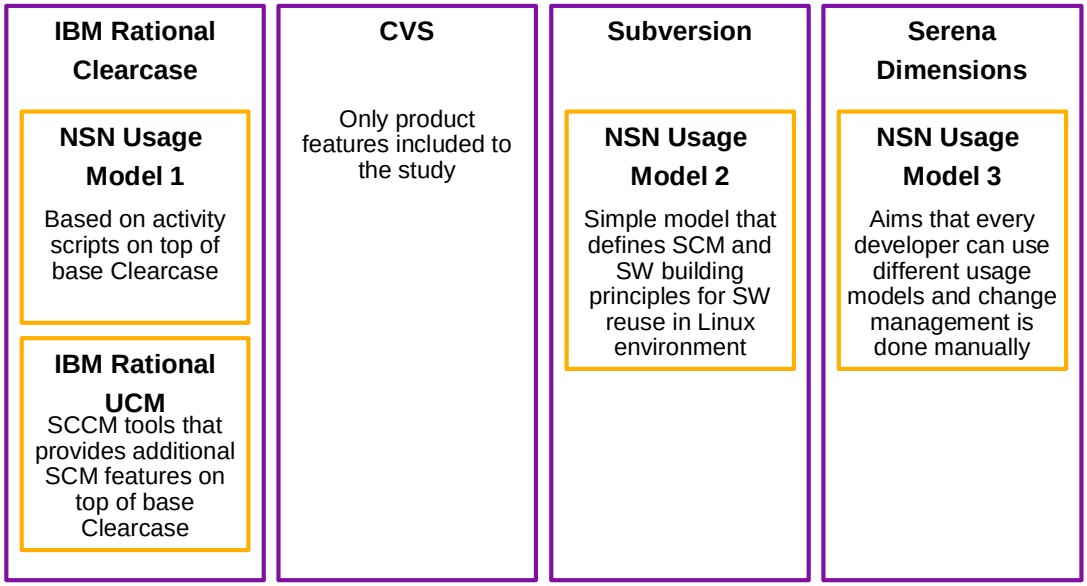

Figure 11: Version Control Systems and SCM Usage Models studied

IBM Rational's *Clearcase* is studied with two different SCM usage models. The first one is an in-house model that is realized with activity scripts on top of base Clearcase and the second one is IBM Rational's *UCM* that includes all needed features under one graphical user interface (GUI). Only product features are studied from the CVS. This was because Subversion is built with same principles and it should include all the features included CVS and more. CVS was selected for this study to compare it to the Subversion and other VCSs. Subversion and Serena Dimensions are studied alongside the in-house SCM usage models. Dimensions has own SCM usage model, but it was only partly exploited in the development environment.

8.1 Base Clearcase with UCM and NSN SCM Usage Model 1

Base Clearcase is a widely used commercial SCM tool. Clearcase provides a complete package for SCM that includes a great amount of features that can be exploited in the SCM process. It provides features like secure version control

system, parallel development support, automated workspace management and activity-based change management. Additionally, it provides individual and team stream models with private workspaces and public integration areas. (IBM 2007)

For base Clearcase, there is an additional package called Rational Clearcase Multisite which provides support for multisite development. Each location has its own replica of VOB (Versioned Object Base). A VOB provides permanent storage for an entire directory tree: directories, files and links. Updating to other replicas can be done manually or it can be automated. This can also be used in single location development environment for backups, for independent groups to work with the same development data and to enable VOB interoperation in mixed environments. (IBM 2003)

SCM Usage Model - UCM

UCM is IBM Rational's approach to managing change in software system development from requirements to release. "Unified change management spans the development lifecycle, defining how to manage change to requirements, design models, documentation, components, test cases, and source code." (Clemm 2000) Clearcase and ClearQuest are the foundation for UCM. (Clemm 2000)

UCM unifies the activities used to plan and track SW development processes. These activities fulfill the change management process presented in Chapter Four. UCM manages all the artifacts produced in a software project including system and project management artifacts. It also manages the project's tasks, defects, and requests for enhancements and provides charting and reporting tools for SW project management. (Clemm 2000) The projects tasks are, for example, development of new feature or making correction of a SW fault found.

SCM Usage Model – NSN SCM Usage Model 1

NSN SCM Usage Model 1 is a solution that utilizes basic Clearcase functionality and additionally increases the amount of features with a set of activity scripts on top of Clearcase. This usage model exploits a large amount of features like UCM, but it is designed and developed inside NSN to meet the criteria of a SW product

that has this model in use. It has also been designed to offer better performance than UCM by customizing it to the development environment of the SW project.

8.2 CVS

CVS is a version control system for SCM use that enables history of source files and documents. Subversion has been developed to replace CVS and correct downsizes of the CVS. (Collabnet, 2007) CVS is included for comparison to see how well it bears out when compared to modern VCSs. However it is not however, discussed very deeply in this thesis, because it is not widely used inside NSN and therefore, not a VCS that is worth supporting in future. This is because, for example, the development of Subversion is based on same principles as the CVS and provides further developed functionalities and better support. (Collabnet 2007)

8.3 Subversion (1.4.x) with NSN SCM Usage Model 2

Subversion is an Open Source VCS that manages files and directories, and the changes which are made to them over time like any other VCS. Subversion has a central repository that holds all the software components. Every designer has a copy of the entire software package on his computer which is based on a certain revision number. Every update to the repository holds only deltas compared to the older version. When a developer checks updated files to the repository, only changes in the files are saved, not the whole file. This makes it possible to recover older versions of data and to examine the history of how the data has been changed. When changes are relatively minor compared to the file's size, saving only deltas reduces the amount of hard disk space needed. (SVN-handbook 2004)

Subversion does not save the individual file version or track the history of individual files, but it does however track changes to the whole directory tree. Commits are truly atomic, because no part of a commit takes effect until the entire commit has succeeded. Revision numbers are per-commit, not per-file. (SVN-handbook 2004)

Branching and tagging are cheap constant time operations in Subversion. That is because when doing a branch or tag, only the revision number is saved not the whole directory tree. (Collabnet 2007)

One big advantage with Subversion is that it can be plugged into the Apache HTTP Server as an extension module. This gives Subversion a big advantage in stability and interoperability, and instant access to existing features provided by that server—authentication, authorization, wire compression, and so on. A more lightweight, standalone Subversion server process is also available. This server speaks a custom protocol which can be easily tunneled over SSH (Secure Shell). (SVN-hadbook 2004)

There is still a major problem with merge, because subversion does not support merge tracking. This generates additional work when merging a fix branch to the main branch if the revision number has not been stored when making the branch. (Collabnet 2007) Subversion fulfills only VCS functionality and because of that it requires additional tools and scripting when implementing it to a midsized or a large project. However solutions utilized in one SW product can be exploited in other SW products.

In Subversion, the local computer holds the whole copy of the software package at a certain revision number and when pulling the revision from the Subversion repository for the first time, a great amount of data can be transferred to the local computer. This can take a long time if the network connection is slow, for example, between continents. (SVN-handbook 2004) To solve potential multisite problems, some extensions have been made on the top of Subversion. Svnsync and Pushmi are introduced below.

Svnsync is the Subversion remote repository mirroring tool. Basically, it allows replaying the revisions from one repository to other repositories. By mirroring with Svnsync, the mirrored components are read-only copies in repositories that have mirrored the master repository. This enables faster checkout times when pulling in all the data from repositories, because every repository has its own local copy of every repository. (Collins-Sussman et al. 2007) Svnsync is presented in Figure 12 overleaf.

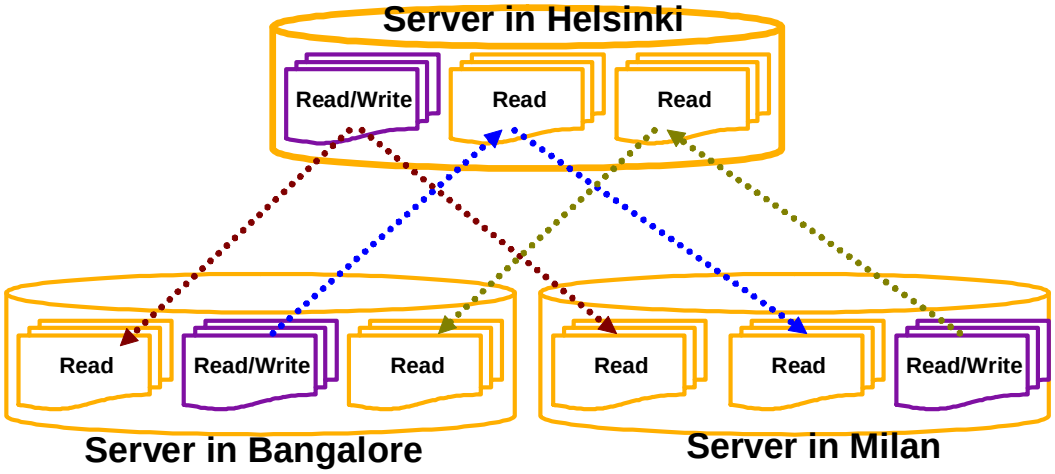

Figure 12: Replication with Svnsync

Pushmi provides a mechanism for bidirectional synchronizing of Subversion repositories. The main difference between Pushmi and other replication tools is that Pushmi makes the "slave" repositories writable by normal Subversion clients. It provides shared responsibility and write-permissions for all sites. The actual write operation does not take place in the slave repository however. The developer performs write operation through the master server, by adding a lock to the server and committing changes to the repository. (Chia-liang 2007) Pushmi is presented in Figure 13.

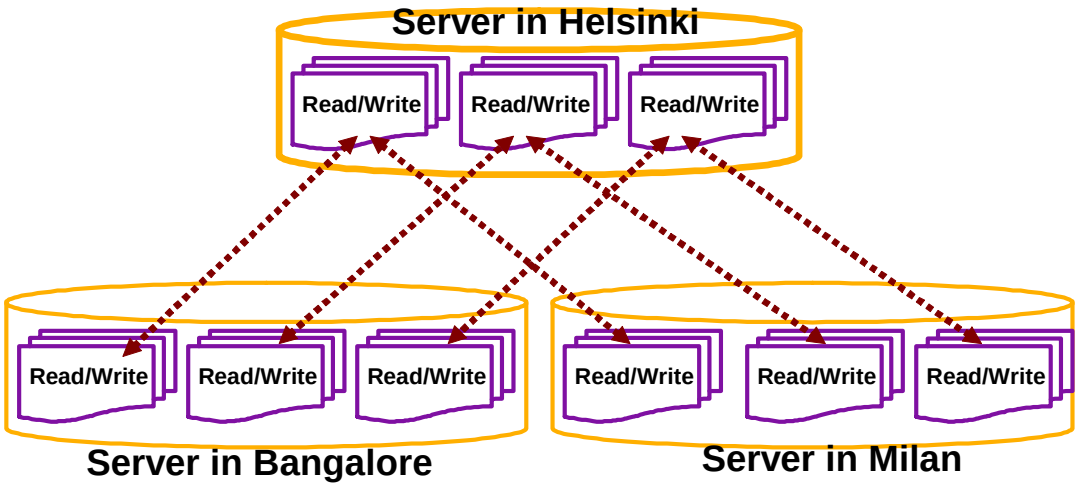

Figure 13: Replication with Pushmi

SCM Usage Model – NSN SCM Usage Model 2

Usage Model 2 is NSN's in-house SCM usage model that defines SCM and SW building principles for SW reuse in a Linux environment using Subversion. Subversion aspires to create common SW environments between different SW projects to have similar processes and tools to generate software code that can be used in several SW projects if needed. Subversion usage in this thesis is examined at NSN in the SCM Usage Model 2 process.

The NSN SCM Usage Model 2 states that for the practical success of the enterprise level SW reuse, both the VCS and SW build system must be the same. Subversion holds all the software components enabling several SW projects to use the same Subversion repository. The main point is that, it requires all SW projects to use the same shared repository and have common principles to enable component reuse. The SW build system also needs also to be similar to limit the number of changes that must be made when reusing SW components elsewhere in the organization.

NSN SCM Usage Model 2 has a set of principles that specify it. This usage model is used for component based SW development where all the SW components are stored in the Subversion repository which has a Subversion specific directory structure. This structure holds four directories: branches, hooks, tags and the trunk. (SVN-handbook, 2004) The model describes the repository structure to store all the SW projects in one repository and it defines different rules how to name and label SW components.

8.4 Serena Dimensions (10.1.2) with NSN SCM Usage Model 3

Dimensions by Serena Software is a multi-functional SCCM tool that provides a wide range of functionalities for SW development use. It offers an end-to-end cross-platform, highly scalable solution for distributed software development. Dimensions integrates a complete set of SCM capabilities, including version, issue, build, baseline, and release management, all within a process-based change management engine. (Serena, 2008)

Dimensions offers a highly adjustable platform for development view and usage model possibilities. Development view, for example, can be changed to respond to the needs that different developers or development functions have. In addition Dimensions provides functionality that aspires to control a whole product development life cycle. This functionality is called ALM (Application Lifecycle Management).

SCM Usage Model – NSN SCM Usage Model 3

NSN SCM Usage Model 3 is based on Dimensions using the VCS, change management and compilation functionalities. This usage model aspires to offer a common interface to the SCM so that all developers can use their own methods to develop SW and announce SW changes to the baselines. Some developers uses Dimensions' own change management interface that provides the best way to manage changes from the SCM process point of view, because it supports workflow in Dimension. In addition to this, developers use their own activity scripts and also notify changes with certain types of email templates. Because of different announcement models, this part of the usage model is difficult to automate. In this usage model baselines are produced from these different SW announcements and, upon these baselines, all SW builds are generated.

9. Requirements affecting SCM

This chapter holds the main contribution of this thesis. It presents requirements found for SCM service and results of how well these requirements are noticed in examined VCSs. These different requirements are collected in the sub-sections. These include an introduction, the question set and study results relating to the requirements in question. Each question in the question set includes background information that describes why a particular question should be asked. The question set created can be used when studying, the current or future state of any SW project examined, and responses to these questions can be exploited when designing the SCM solution for studied SW project. The sets can be used for both new and existing SW projects by applying these to the research context. The full question sets are presented at the final chapter of this thesis, “Conclutions”.

Results of this survey are discussed in this chapter and included in Appendix 4. Different sections in the question list were rated from 0 to 2 based on how well requirement can be controlled with different VCSs. This numbering is not exact and has not been included to grade different VCSs, but rater to give indicative information of how well different VCSs handle the section in question.

- “2” Section is well controlled by the VCS in question
- “1” Section is not easily supported in the VCS in question
- “0” Section is not feasible to realize with the VCS in question and need to be realized using other VCSs

9.1 Existence of SW Project

Details of SW product existence are the first thing to ask when starting to plan any SCM solution for a SW product. This is because the characteristics of the SW product and development environment must be taken into account when designing a new SCM process or making changes to it. The point in the products life cycle is also important to identify in order to measure which changes are profitable to make. This depends greatly on the future of the indivual SW project.

Is the SW Project New or an Existing one?

When a product is new, the usage model can be planned from scratch. With an existing product however, parts from the existing usage model and processes can be exploited if needed. There is a significant change, for example, when moving from Clearcase to Subversion. It requires new software, new processes, new thinking models and new competences. In some cases, it can be too difficult or too expensive to change something from the existing model. It is important to plan what is both profitable and productive.

At the start of a new SW product, the amount of data can be zero or there can be some code imported, for example, from another SW project. When there is no previous content repository structure and utilization, as well as practices for building software configuration can be designed from the beginning of development. On the other hand, when designing a new SCM solution for a product that is under development, there is a software configuration and data existing in some repository or repositories. In this case the existing solution must be taken into account and considered if some parts of the existing models or tools must be included in the new SCM solution. Depending on these processes and tools in the existing product, new ways of working will be designed and implemented. This is the case also when SW components are reused from other SW products.

Starting a SW Product or Ramping It Down?

It is important to know where the project is going in a product's lifecycle. If the product is at the end of its life cycle it is not worthwhile to change everything. When a project is ramping up, there is much more latitude to change things because everybody is still tying to find the right model and way to work.

Research Results

Both Subversion and CVS are quick and easy to set up, whereas other VCSs studied require more configuring. There are no license fees to consider when starting or ending the SW project, because Subversion and CVS are Open Source software. These VCSs offer only version control functionalities, but there are Open Source tools that offer additional development features. UCM and Dimensions are,

on the other hand, designed to include all needed features making additional tools unnecessary. These two tools offer the best “out of the box” solution for working with a totally new software product. Base Clearcase lies somewhere in the middle, because it includes some basic SCM features including VCS functionality, project baselining as well as development workspace management. It requires scripting or additional tools for automation and to include more features. In addition, Subversion and CVS are very firewall friendly, because these tools do not require more than the HTTP(S) protocol (Hypertext Transfer Protocol Secure) to get access. Base Clearcase, UCM and Dimensions require wider IP-port (Internet Protocol) space in firewall configurations.

9.2 Quantitative Information

SCM Service is necessary regardless of the development project size. Its formality and scale varies depending on the size of the individual project. Even a single person can be responsible of a whole SCM Service in small projects, but depending on different variables, additional resources are needed. Some of these variables are described in the next chapters and in the figure below.

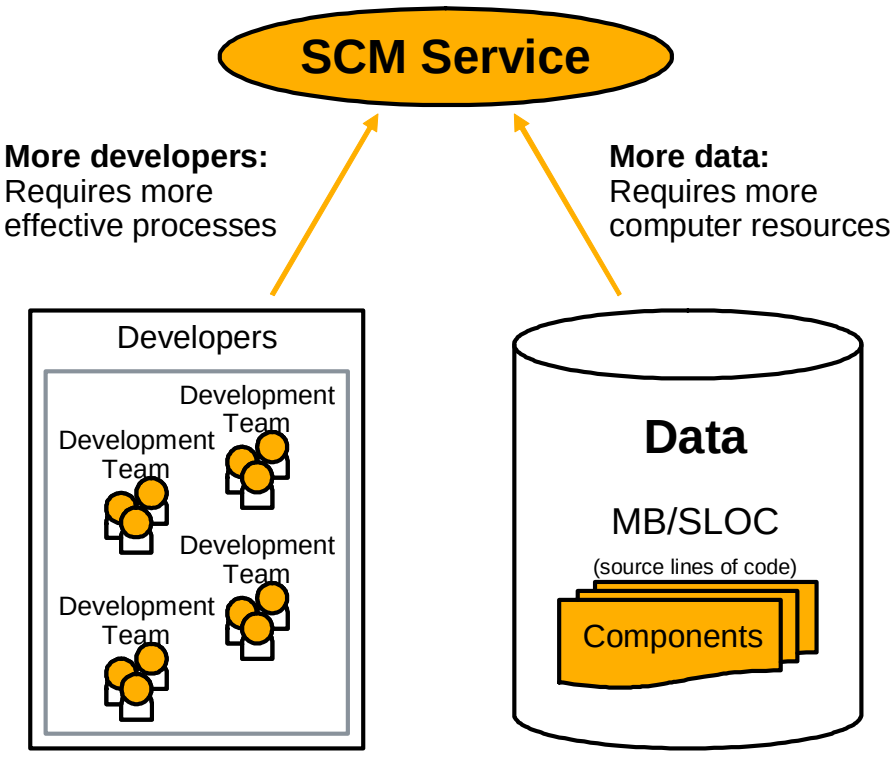

Figure 14: Quantitative Information in a Software Development Project

The amount of different variables directly affects the amount of work and control that needs to be taken into account when designing any SCM solution. However, modern SCM tools can reduce the great amount of work that needs to be done because of automated and far-sighted processes. Tools like Rational UCM and Serena Dimensions offer complete usage models and features for SCM processes and solutions. The Open Source VCS Subversion takes a different approach by simplifying the SCM usage model and offering interfaces that are easy to use and implement in other products that can provide other features like continues integration and software building frameworks. (Kääriäinen 2006)

What is The Number of Developers?

"Small projects need less formal practices and a single person can be responsible for the CM (Configuration Management) coordination." (Kääriäinen, 2006)

The amount of developers in a SW development project gives some hint of the size of the project. The aim is to know how large the project is from the personnel point of view and the estimate resources needed for the SCM. The amount of work depends on the diversity of SW configuration and the responsibilities that are assigned to SCM Service. The benefits of multi-functional tools, like Base Clearcase or UCM, can get under exploited in small projects where product complexity is small.

"Large projects need formal practices and a CM team." (Kääriäinen 2006)

In large and complex SW product development, multi-functional tools that provide "out of the box" solutions for SCM Service will offer major advantages when starting the SW product. Many functions included can bring benefits for the SW project in future. However, complexity and the size of the SW project do not restrict the use of more focused tools like the Open Source Subversion that only provides the base VCS functionality. This is because, for example, Subversion has many tools developed to expand functionalities of Subversion, and these tools do not restrict the use of other tools for other functionalities.

What is the Number of Development Teams?

How is the responsibility shared through the product lifecycle and what is the responsibility of one development team in the cycle? When development is organized between development teams, the requirements are different from situations where responsibilities are divided between individual developers. Responsibilities can be designed in a way that it is the designers' responsibility to know who has rights to make changes to different components. There can also be some group that is responsible for some particular component, but then the information flows need to be arranged so that everybody in that group is aware of all changes in that component. The development team can have shared responsibility for specified SW components or SW features. For example, in Agile development teams share the responsibility of all the software components, but have specified responsibilities for certain SW features or changes made in one development cycle.

What is the Amount of Data in SW Build or in repository?

This question can be divided in to two different viewpoints. The first part is the physical size of the data, which can be measured in megabytes (MB), source lines of code (SLOC) or number and size of SW components. The amount of data in SW builds displays how large the SW is and it gives the magnitude of how much configuration work there is. The second point of view is the difference in the magnitudes of the source code and binary data. VCSs are mainly designed to store text-files (e.g. source code), but the structure of binary-files is not text-based, meaning that the efficiency of storing binary-files is not so good. There are also differences in the management of binary-files in different VCSs. Base Clearcase and UCM, for example, have good support for managing and storing binary-files.

Research Results

The study indicates that there are no strict limitations of different quantities. None of the VCSs limit the amount of developers using the VCS in the development environment. However, there has been a guideline recommended that the maximum number of developers should be under 50 when using UCM. This is because UCM development with UCM is based on making baselines of every

development event and including them in the baseline of one release. With many developers, control of the development becomes more complex and the amount of metadata produced by UCM increases dramatically.

VCSs can be configured to support several development teams. The main point is that permissions can be set so that either only one development team can change certain components from a repository and development can be distributed between different development teams. There are no limitations on the amount of data stored in repositories with any of the studied VCSs. However, if data stored to the repository is binaries, Clearcase is the best VCS to control these items.

9.3 Building Software

The building of software is a phase in software development where baselines are done and baselined software items are compiled with specified compilers into executable code. This phase can be considered to be a part of testing, because one purpose is to ensure that all the software items included in the baseline are built together. SW builds are made by using some build utility like Make or Maven. (White 2001)

What is the Development Model?

Here should be described the current way to develop software in SW projects. SCM is a service function that operates in the context of the development process. (Kääriäinen 2006) Development models can range, for example, from straightforward to iterative. The main focus in this thesis is on two different models. The first, Waterfall, which is a sequential software development model in which development is seen as flowing steadily downwards (like a waterfall) through the phases of requirements analysis, design, implementation, testing (validation), integration, and maintenance. (TechRepublic 2006) The second model uses the Agile method which attempts to minimize risk by developing software in short time boxes, called iterations, which typically last from one to four weeks. Each iteration is like a miniature software project of its own, and includes all of the tasks necessary to release the mini-increment of the new functionality: planning, requirements analysis, design, coding, testing, and documentation. While iteration

may not add enough functionality to warrant releasing the product, an Agile software project is designed to be capable of releasing new software at the end of every iteration. (Koskela 2003; Appleton 2004)

Is There a Need for Multiple Customer Versions?

Is the product the same for every customer? If the same software is delivered to every customer, the product can be open for every customer or limited by licenses (or parameters). In this case, the customer needs to buy licenses to get features activated. Software can also be truly independent for all the customers. (Kääriäinen 2006)

What is the Software Architecture and is there Need to Support Multi-Architecture?

SW architecture is basically a specification that defines the hardware system where the software runs. The SW architecture includes the high-level structure of the SW system's interfaces, components and components' relationships. The platform SW packages operate as an interface that enables an application SW to use hardware (HW) characteristics. The SCM is connected to the SW architecture with SW components and those relationships. The SW objects must be stored in a version controlled way. In addition, the SCM must be able to baseline different SW components to different baselines that are designed to run on specified hardware systems. It is also important that the build system supports all kind of relationships between different SW components. (Westfechtel 2001)

Is the SW Product Component Based?

Component based products consist of several SW components that all are loaded to the target environment. The component structure enables easy use of off-the-shelf and third party SW components in SW builds, significantly reducing cost and time to market in development. System maintainability and flexibility is also improved because new SW components can be replaced with older ones in target systems. (Pour 1998) In non-component based SW the whole SW package is always delivered and installed to the target system. Component based SW development

requires more from the SCM, because an ability to track changes between different baselines and ability to produce upgrade SW packages is needed.

Is the Developed Product a Software Platform or Application Part?

Software can be consisted of two different parts called the platform and the application. The platform in software development realizes a software interface to hardware architecture and this interface is offered for the application SW package to exploit. The application part of the SW realizes the functionality of the SW product. This means that the platform needs to govern the interfaces for the products that are using the platform. There can, for example, be some differences how different applications utilize different computer units or different applications use different configurations of hardware. The development and delivery of a platform SW package has a requirement to generate SW packages of different content for different SW projects using that platform.

In large companies, like in NSN, there are several products that use platform packages as base packages for a product. This is because the existence of a versatile SW platform can reduce a significant amount of development expenses. From SCM point of view, when designing any SCM solution for SW projects, it is important to know what SW platform is in use for the SW product in question. This is because every platform has its own characteristics that are common for all the products that are using it. There can also be ready-made implementations to be exploited in the design.

Is there a Need to Have Development and Runtime in same Environment?

Having both the development and runtime environments in the same system offers some advantages that individual systems do not allow. For example, new SW components can be compiled and tested directly on the target machine. In individual environments, development of SW is done in the development environment where the SW components are compiled and assembled as software packages that can be downloaded to the test environment where testing is carried out. In telecommunications, SW development test machines are usually located in a standalone laboratory network to avoid routing the network traffic of the testing

environment to the rest of the company intranet. This can be problematic when both development and runtime environments are located in a laboratory network, because developers are usually located on the company's intranet and need to have access to the development environment.

What are the Item Types Produced during the Software Development?

During the software development process many types of items are produced ranging from software components to SW specification as well as different tools. SCM has conventionally operated with source code elements, but today all items produced in the software development process must be managed. Items managed and stored are, for example, requirements, management documents, designs, code, compilers, operating systems, binaries, instructions, etc. (Kääriäinen 2006; ANSI/IEEE 1042-1987 1987) How these different items are managed and maintained, depends on the development organization and the SCM service.

Research Results

All the VCSs' support both the Waterfall and Agile development models. All the VCSs can be integrated to CI (Continuous Integration) build servers like CruiseControl or Atlassian Bamboo to enable Agile development. Subversion in particularly integrates well to the CruiseControl and a great amount of the Open Source CI development is done on top of Subversion. Serena Dimensions includes also its own CI build server. Then Subversion needs scripting to handle rules that restrict access in some components while allowing it in other components. UCM has a great amount of features which can handle many operations. The basic Clearcase also needs scripting if the SCM wants to use unusual features. This can be issue when development is using the Agile development model, but it is desired that only some parts of the software are open for changes.

All of the VCSs support branching of software to provide baselines with different content. The SCCM tools UCM and Serena Dimensions include change management tools to control the workflow of different baselines. Merging the different baselines is a problem in Subversion, but there should be an improvement in Subversion release 1.5.

The VCSs studied support almost the same sets of operating systems. This means that if the runtime environment is one of the supported operating systems, the environment for runtime and development can be the same. Differences arise from the connection to the environment if the development and testing organizations are located on different networks. Subversion and CVS require less configuring work to firewalls than other studied VCSs. Clearcase's file system is highly integrated to the Linux kernel and this is problem when SW development follows the latest Linux Kernel version. This is a problem, because Clearcase supports Linux kernels with a delay. With Subversion and CVS this is not a problem, because with Open Source the source code of the tool can be compiled against each environment needed.

Every VCS can manage all the file types and are designed especially for storing text files. Clearcase's Merge Manager can support different file types and help solve conflict problems when the same items in the repository are changing at same time.

9.4 Programming Language

A Programming language is an artificial language that is made for controlling the behavior of a computer. Programming languages are defined with syntactic and semantic rules that define its structure. (Friedman et al. 2001) Some of the programming languages are, for example, Python, Java, C, C++ and Perl.

What are the Programming Languages?

Programming languages have significance when selecting the build system to the SCM environment. This is because every build system does not support every software language. Maven2, for example, is designed only for Java compiling. There can also be some characteristics of different build systems that can influence the selection of VCS. Maven2, for example, functions well with the VCS Subversion. (Maven 2008)

Research Results

The VCSs studied can store any file types and in this way support storage of all the programming languages. In addition, Clearmake in Clearcase and UCM offers support for the compilation of different programming languages. Winkin functionality in Clearcase can enhance the performance of SW building if the SW code is implemented with the right principles. Winkin provides access to already compiled SW components and reduces the building time when using these items. When using Subversion and CVS, the dependency structure for the compilation of different programming languages must be developed separately. There is also the possibility to use separate build systems like Maven2 or Make.

9.5 Multisite Development and Replication

In software development, multisite projects are those where the development organization is geographically distributed to more than one development site. The most significant challenges in multisite development environments are greatly reduced communication, network limitations for data transfer, and cultural as well as national differences. (Armstrong & Cole, 2002; Herbsleb et al. 2005)

The figure below describes a multisite situation where the development between distributed to three different development sites. The central repository is located at the Helsinki site and the repositories in London and Bangalore sites are replicas of the Helsinki repository. Personnel in all of these sites can connect to their own dedicated repositories.

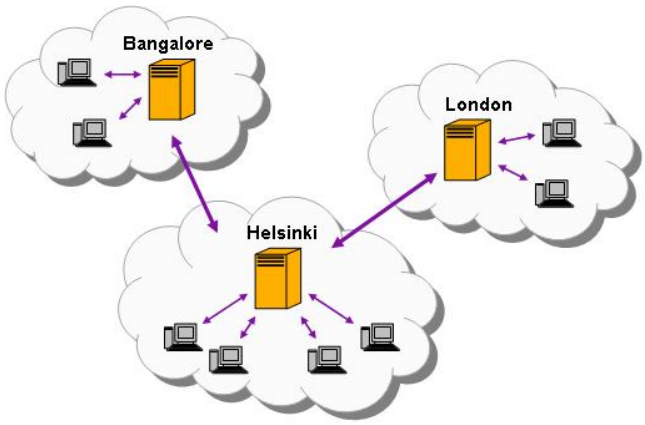

Figure 15: Multisite development

Are There Some Development Teams Located in Different Development Centers?

Multiple development sites generate special requirements for SCM that single site development does not. Repositories need to be located in suitable locations so that all the sites can exploit them as well as possible. Multisite development also demands good information sharing and management of processes. Responsibilities need also to be clear between sites and it is desirable that every site follows the same process. If sites do not follow the same process, then differences need to be clear and processes integrated. (Kääriäinen 2006)

Where is Development Located?

Software development can be clarified as multisite in cases even when the development teams are in the same city but do not have direct face-to-face contact. However, the situation is usually that these teams are located in a different development centers (DCs). There is a significant difference how the development sites are geographically located. Are the DCs located in same country or different countries? Two example scenarios could be:

1. If data needs to be transferred between two different locations inside Finland replication is not needed, because the data connections are generally good. It still, however depends on server placement and network bottlenecks.
2. If data needs to be transferred from Finland to China, replication is probably needed between different servers, because of the long physical distances.

When designing SCM solutions, cultural differences need also to be taken into account. For example, in Finland cultural differences must be taken into account if the solution is used in a different country. Habits and communication practices are different in and between different countries. (Herbsleb & Moitra 2001) Different sites and development projects also have their own unique ways of communicating and doing tasks.

Is There Any Services Located in Different Sites? (Testing, SCM, etc.)

By defining where the different development functions are located, it is possible to determine the requirements for multisite development. These requirements effect different areas of the development environment such as data connections, repository structure, as well as information flow between different development groups or development project managers.

The figure below describes an imaginary multisite situation where the development tasks are divided between three different DCs. The main development is located at Helsinki DC. The development teams in Oulu DC and in Bangalore DC implement some specific SW features, and Bangalore DC has also responsibility for all I&V functions.

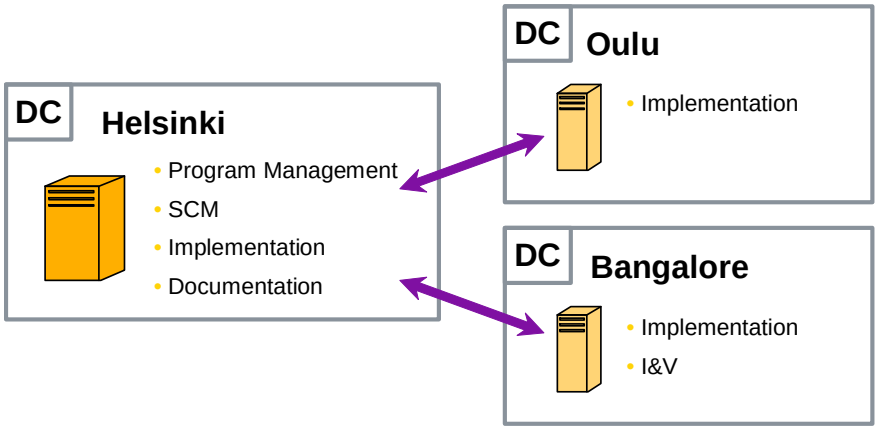

Figure 16: Development Functions Distributed

What are the Methods to Transfer and Share Data between Sites?

Data distribution between different development sites can be realized in numerous ways but it is always important that the data is located in the site or continent where the need for data transfer is greatest. This can be measured with data traffic between different development functions or groups. Without measuring most data traffic may be with SW developers and SCM service because the need for transferring data is frequent and volumes are large. Other functions like I&V and customer care need to transfer data randomly. Clearly different development functions have different requirements. For example, in cases where testing is located at a different site, only the releases that are needed could be replicated or

downloaded to a server where the testing personnel has direct access. There is no need for continuous data synchronization.

When the SCM is located in a different continent than the rest of the development, there could be a need for making the actual software compilation job in a server at the development site, because it can be slow to transfer a great amount of data through the Internet. It is also possible to design a repository network where the repositories are replicated to different sites.

Is Replication needed between Databases?

Database replication stands for the creation and maintenance of multiple copies of the same database. In most implementations of database replication, one database server maintains the master copy of the database and additional database servers maintain slave copies of the database. Database updates are sent to the master database server and after that are replicated to the slave database servers. The load of reading from the database can be distributed between the databases. Database replication has advantages, but also difficulties. The advantages are fast connection to databases and shared load between databases. The difficulties are, for example the fact that all the databases do not have the same content for the whole time. This is because the synchronization of data between databases takes time and is not always performed immediately. Replication also increases the complexity of realization, creates challenges for data transmission between sites and introduces unreliability to data updates. Basically, it is preferred that one database solution should be used whenever possible.

There are two main replication models for database replication: master/slave and multi-master replication. In master/slave replication, the master database logs the updates and then changes are dispatched to slave databases. In multi-master replication, updates can be sent to every database node where changes are distributed to other database nodes. The most common challenge that exists in multi-master replication is transactional conflict prevention or resolution. The network can try to prevent simultaneous updates to the same components or try to automatically resolve this situation.

How Easy Is It to Put Up New Development Groups at new Development Sites in the future?

In large companies, distributed development between different development sites is common and putting up new development sites is possible. This can also be an issue when there are new collaborators included in the development environment. In this situation, there are several issues to consider like firewall settings between the new development site and the existing development environment, new hardware and software is required to development environment, and the degree of training required for incoming personnel.

Research Results

- From every VCS changes in the repository can be tracked and visibility between different sites can be realized. The SCCM tools, UCM and Dimensions and the VCS Base Clearcase include change management functionalities that enable better visibility of development work between different development teams. The VCSs Subversion and CVS can be used with explorer view with, for example, TortoiseSVN and TortoiseCVS. This improves visibility of work between different sites, but does not correspond to the work flow management functionalities included in UCM, Dimensions or base Clearcase.

- With all, the VCS development functions can be distributed between different sites when all development is realized around one development server. All the VCSs, however, enable replication solutions to multi-server development environments.

- Replication is usually problematic; because data between replicas is not real time congruent and replication increases administrational work. Replication with Clearcase is cheap in situations where network bandwidth is small, but it requires many firewall ports open for network traffic and several administrators, because of complexity. However, Clearcase offers more replication possibilities than replication performed in Subversion. Replication for Subversion is easy to set up because of easy replication and

firewall configuration. There are different Open Source replication tools like Svnsync and Pushmi for Subversion that support different replication scenarios. The CVS tool, on the other hand do not support or have Open Source solutions for replication. Replication in the CVS tool can be realized with a commercial replication tool WanDisco. With Dimensions there where no experiences of replication, but replication between different repositories is possible to set up.

- Subversion and CVS differ substantially from Clearcase, UCM and Dimensions when starting new development sites for development. These Open Source VCSs are quick to set up because no license purchase is required. Installation and configuration of these tools is also a quick task. Clearcase, UCM and Dimensions all require licenses and setting up the development environment is more complex because these are versatile tools and have more configuration choices. With Subversion, starting a new development site is remarkable easy. This is because the new development site can put up a new Subversion repository and start to replicate the base repository without changes to the existing development environment and only the HTTP(S) protocol is needed to have access between the different networks.

9.6 Operating System and VCS Workspaces

The choice of operating systems is crucial for the development environments and when designing SCM solutions for SW projects. This is because the VCS and other tools selected must support the operating systems used in the development environment. There can be also a need that the VCS should be able to be installed in a runtime environment on a target machine.

The way VCS workspaces are used can direct processes and general development work done in the development environment. The most significant difference between different workspaces is their connection type to the repository. These workspaces are managed with tool clients. (IBM 2007; SVN-handbook 2004)

What are the Operating Systems in Development Environment?

When selecting a VCS for any SCM solution, it is necessary to identify the operating system used. This will ease the tool selection process, because some products run better on some operating system than others and some tools are not developed to run on some operating systems at all. VCSs like Subversion and Clearcase do not have this problem in most of the development environments because these VCS's offer full cross-platform support for Windows, Linux and UNIX operating systems. (IBM 2008; Collabnet 2008)

The VCSs studied enables the use of multiple operating systems use in the software development environment because the same tool can be needed in different operating systems. This enables, for example, a situation where developers and the SCM can use different operating systems on their computers.

What is the Operating System in Runtime Environment?

Operating system and server utilization can be designed in greater detail when runtime environments operating system is known. This is because if the runtime environment runs on a common platform like Linux, testing of the SW can be partly done on the same operating system where the software has been developed and compiled. Usage of this kind of environment makes both implementation of code and testing possible in offline use. When development tools cannot be installed on the same environment than the SW itself, testing must be done always in real hardware or in an environment that simulates the interfaces of the needed hardware.

Do Tools need Online Connection to VCS?

VCS tools that are evaluated in this thesis have clients that have two different connection types in repositories. Developers can have offline clients when they check out software components needed from the repository and start working on their computer's hard disk. After making changes to the software, the developers check changes in the repository. Online clients mean that the development environment is connected directly to the repository. Developers access the files directly from the repository. Offline clients enable development without Internet

connectivity, but this consumes computer resources. Clearcase supports these two ways of developing software. Online clients are called dynamic views and offline clients are called snapshot views. Subversion supports only working copies on developers' computers.

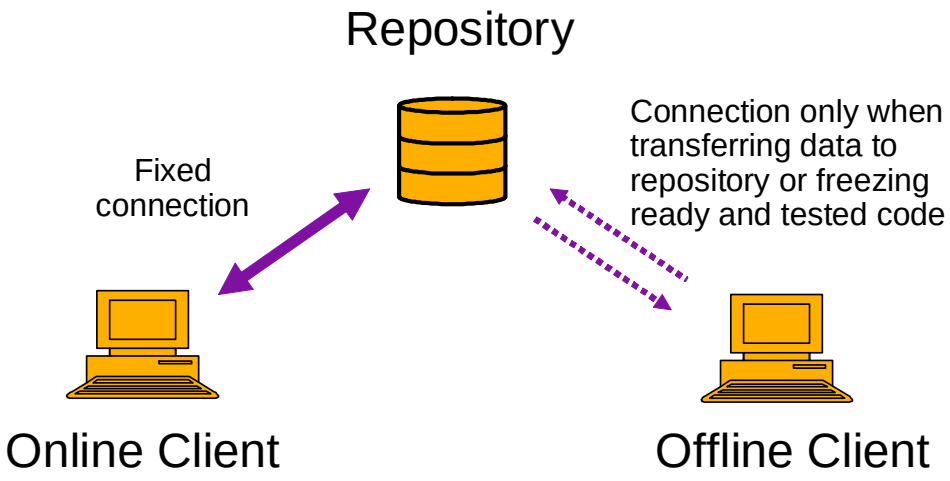

Figure 17: Online and Offline Clients in SW Development Environment

Research Results

- All VCSs studied support the operating systems Linux, UNIX and Windows as distinct from Clearcase which does not support latest Linux kernel version. The VCSs Subversion, CVS, Clearcase and UCM support also Sun Solaris and in addition Subversion can be installed on Mac OS X. Generally speaking, Subversion and CVS are available for all operating systems, because the source code can be compiled for every platform needed as long as compilers are available.

- VCSs can be used in runtime environments as long as the VCS support the operating system. Firewall settings can be an issue for Clearcase and Dimensions because the runtime environments can be located in different networks from the developers. These VCSs require more open firewall settings than Subversion or CVS.

- Development work in both Subversion and CVS is based on offline connectivity where all development is done on the developers' personal computers and after that all the changes are checked in to the repository. Base Clearcase, UCM and Dimensions all support both online and offline development, but the workflow in Clearcase based VCS is designed to function using an online connection called *dynamic view*. Clearcase's dynamic view enables a productive approach to development work that differs from other VCSs. It provides development views directly to the file system where changes to the SW can be made.

9.7 Resources Existing and Needed

Resources existing and needed are important points to consider when making changes or designing a new SCM solution. This is because resources are used effectively when the use of resources is divided between different functions in SW development projects. For example, one SW development project could share the same databases for data storage between different development functions. In large companies, one development function like SCM service could share resources between different development projects. The intellectual property of personnel is also an important asset, because existing skills and experience can make the success of environment changes possible.

What Recourses will be Available or Required for SCM Activities?

In case a SW product has been under development for some time, then there are existing SCM solutions and development environments. By evaluating processes and finding out needs for changes, available resources can be either kept or swapped. This all depends on how well different solutions match the development environment as well as development project's and company's demands. Recourses can contain, for example, people, system, tools and methods. There can also be resources that the SCM can utilize from other development functions or projects like servers and tools. (ANSI/IEEE 1042-1987 1987)

It is also important to understand what the requirements are for SCM service from the development project. It is pointless to design a solution for a SW project

without first understanding what is needed from the SCM and what the limitations for realization are. Changes to the SCM solution can affect the whole development environment by changing tools and ways of work.

Previous SCM Tools

It is important to know either an employee has previous experience of working with SCM tools. If SCM tools are familiar, less training is needed to be provided to developers. Additionally, starting with a new usage model can be controlled better than by using totally new tools and methods. On the other hand, if the project team has bad experiences with some SCM tool, it is better able to consider something new.

How much resources are there available for SCM?

In every SW development project, resources are limited and processes as well as tools used must be designed to fit within these boundaries. There are different situations that have different limitations for available resources. For example, new innovations may require small investments to the development environment because it is not certain that the innovation in question will be popular in the markets. In projects where available resources are small Open Source tools can bring great cost savings because Open Source tools are free to use and exploit by anyone or organization. However more versatile VCSs can also be profitable because those offer ready-made development processes and do not require as much implementation work. In large scale SW development projects, budgets can be more flexible and more expensive tools can be used. In these cases the selection of tool depends more on need than on cost. In large projects, commercial SCCM tools can bring great advantages in supporting different characteristics and requirements of SW development because of their versatile nature. (Clemm 2000; Duggan & Stang 2008) Open source VCSs are usually more focused than commercial VCSs and because of this do not offer an equally wide variety of features. This, however, can be an important asset because using these tools can simplify processes compared to many other commercial tools. For example, Open Source VCS Subversion with Open Source build server *CruiseControl* or commercial *Maven2* offers effective, easy to use and quick to set up CI solutions. In addition Open

Source tools do not restrict the use of extra tools to enhance the features needed. (SVN-handbook 2004)

Are there Automated Testing Tools Available?

Automated testing tools can affect the SCM process because these can be under the same CI environment. Automated testing tools are included in this thesis because with automated tools better usability and efficiency in development solutions can be achieved. This is also cost effective because modern SCM tools and methods offer solutions to exploit the advantages of automation. This aspect is important because today's SCM solution needs to support other development functions like testing and implementation as well as their tools and methods.

Are there Available Tools Nearby?

If a product is an existing one, there is an existing usage model as well as available tools that can be exploited. When the tools used have become familiar, the decision of maintaining these tools can be made. For example, in a situation where a new VCS is planned to be taken into use, different tools that are used with an existing VCS should be reviewed. This is to find how compatible these tools are likely to be with the new VCS.

Research Results

Subversion and CVS are quick to acquire, set up and learn to use. These VCSs can be implemented quickly to any development environment even if these VCSs are not familiar nor used previously in the development environment. On the negative side, Subversion and CVS do not include versatile features that the project environment could exploit. These tools only offer features for revision control and simple database management. Other needed features must be added with additional tools. On the other hand, Clearcase and especially UCM and Dimensions offer a multi-functional set of features that development environments can exploit. Due their versatility, however, these tools require plenty of training and set up.

The development organizations know-how of the VCSs Clearcase, UCM or Dimensions is an important matter to consider when making decisions about the

usage of different VCSs. This is because the development organizations competence of multi-functional VCSs makes quick adaptation of usage models and set up possible. All the studied VCSs do not restrict the use of different tools which makes the use of automated testing tools possible.

9.8 Collaborators

In this thesis, collaboration is defined to be cooperation between companies. This holds cases from where individuals in one company work with individuals in another company to many complete companies working together in harmony. The advantages of collaboration are extended know-how and more flexible resource control when situations change. The main disadvantage is the limitations caused by company boundaries. Working with en other company is never simple, and communication is more complex between two companies than inside only one. Additionally, differences in companies' operational models can lead to a state of inertia. (Huxman 1996)

Are there Collaborators Involved in the Development Organization? (Coding, Testing, Building, etc)

Collaborator companies' involvement in product development brings the same kind of challenges to the development environment as with multisite development. There are, at least, two different development organizations needing to find a way to have common development rules, share information effectively with each other, and have access to software items needed in development environment. Collaborators are usually two different companies that do not share the same intranet and resources. This requires more from the processes that bond these two development organizations together.

Are Collaborators Allowed to See each Others' Code?

When there are several collaborators involved, communication and information flows needs to be well designed, especially if there are some limitations to sharing the information between these companies. If, for example, the collaborating companies are not allowed to see the code of others, control of SW content and rules for creating code must be carefully designed and implemented. This situation requires, for example, repository configuration that enables full read access to SCM

service that compiles the SW packages as well as limited read and editing rights to the development parts of the repository for the partner companies.

Can Collaborator Access via Internet or only through VPN?

Depending on the individual collaborator, access to a company's network can be arranged in many different ways depending on the networks and tools what are used. If the collaborators can access via VPN (Virtual Private Network), they can use the same tools and servers as the collaborator partner uses. It requires firewall and network configuration to allow a VPN connection from the company's network to the other company's network.

Research Results

Subversion and CVS are easy to purchase, set up and the VCS connection between companies' intranets can be implemented easily. This is because firewalls must allow only the HTTP(S) protocol between the companies' intranets. Clearcase, UCM and Dimensions require license purchases for all the parties that are using the software in question. The set up of these VCSs can require more configuring, but this depends on the usage model realized. The single most time consuming disadvantage with these three VCSs, can be connecting them to the companies' intranets. This is because these tools require more open firewall settings than Subversion or CVS. These firewall settings can be bypassed if the VPN connection arranges it between the development environments. With Subversion new repositories replicated from the master can be added without making any configuration changes to the master.

When collaborators are added to the development environment, permission to access the VCS can be configured in order to restrict that collaborators are able to change each others code. Permissions can be configured one by one for all the defined groups or individuals. However, visibility of the content of repositories is a difficult task to realize. This requires that restrictions are made in the repository level, in such a way that different collaborators have access to different repositories.

9.9 Competences

Competence is a requirement for the individual to be able to perform a specified task. To have competence in performing a defined task, an individual must have knowledge and skills to do that task. Individuals having core competence holds wider knowledge and skills for the specified task. Core competence creates benefits for any company as well as for customers it serves. It is also difficult to be imitated by competitors.

What are the Competences of Project Personnel?

By defining the competencies that project personnel have, it can be discovered whether if some tools or methods are familiar to those personnel. Additionally, well-known processes can be understood in greater detail. This can give some guideline for selecting tools and methods, and it also helps to know what kind of training might be needed. For example, if the project team has a good grasp of using Clearcase products, they can select base Clearcase or UCM to be basis of a new SCM solution.

Do Project Personnel have Important Core Competencies?

Core competencies are essential resources and know-how that a company must have. Product features and services for customers are the first things to occur when speaking about core competencies. However a company must also possess core competences that are invisible for customers, like the ability to exploit tools in processes, and the capability to adapt processes and development models to add effectiveness in software development. These core competences could be, for example, expertise of some particular tool or development process. By studying the background of the project personnel, tools and methods familiar to them, different competences can be revealed. From an SCM point of view expertise in using some VCS, like Subversion or Clearcase, can promote usage of that particular tool.

What kind of Training has there been for Project Personnel during the Project's Lifecycle as well as before it?

By asking this question, competence development needs can be identified for software projects in development environments. After finding out what kind of

training a particular project's personnel has taken, the existing skills can be understood. Additionally, skills needed for the development environment can be determined.

In a situation when the SCM usage model or tools are changed, it can be better understood how different solutions fit to the personnel's know-how. For example, if the personnel do not have any experience of Clearcase, it can be easier to start using UCM or Subversion, because those are easier to learn. UCM has a very good and intuitive user interface and Subversion is build to be as simple as possible to use as well as easy to learn. Competencies are needed in both tools and related processes.

Usability of SCM Tools

Usability is an important issue when selecting new tools or methods for a SW project's use. Every hour used in education and practical training means additional expenses. Tools and methods that have good usability characteristics give benefits in future because the learning curve is shorter for new users and general usage is quicker. There is a great usability difference in SCM tools and methods. For example, UCM and Subversion are two different SCM tools which are developed from two totally different bases. UCM combines under a well-executed graphical user interface (GUI) a great amount of features that offer everything that might desired in development environment. Subversion, on the other hand, offers a tool that is as simple as possible and this can also simplify development methods.

Willingness to Change Current Process

This can be thought of in multiple ways. Changing the current process can be seen a good or a bad thing and usually all the individuals in the development environment do not think same way as their colleagues. If most of the personnel are satisfied with the present process, they can be unwilling to make changes to the process and in this situation reasons must be good to make major changes to the process.

If the project has invested heavily in some development tools or methods, there needs to be very significant reasons to change the existing ones. Difficulties can

also arise from people, their ways of working, their competences and unwillingness to change their normal practices can all give impact to change. It is important to understand how the changes will affect productivity and employee satisfaction at work. It is easy to understand how much a project changes when the development model is changed from a basic straightforward Waterfall model to an iterative Agile development model. If the development model is changed to an iterative model, the development tools need to be changed and developers need to interact more with others and to learn the new working model, for example.

Research Results

Base Clearcase is more complicated to learn than Subversion because it includes more and more demanding command line commands, and it has also more features than Subversion. In UCM, there is a GUI that helps the understanding of the usage models, UCM characteristics and hides the command line function. Subversion can also be integrated into Windows Explorer with TortoiseSVN, so the logic behind Subversion can be quickly understood.

Subversion can be most desirable of the VCSs studied because it is the easiest to get into and experiences inside NSN have been very good. Subversion does not require any license fee, because it is Open Source software. CVS is also easy to obtain and learn but Subversion offers the same functionality and more with the same principles as CVS. Base Clearcase, UCM and Dimensions offer a great amount of different features for any development organization, but a stiff license fee and great software complexity that hinders the desirability of implementing these tools. Exploitation of these tools requires a large amount of competence as well as financial outlay.

9.10 Software Lifecycle

The life cycle of any software release holds two subprojects which are the development and maintenance projects. In a development project, a product’s features are created and the product’s release is published for customers at the end of the project. After the development project, the maintenance project starts and its

purpose is to produce fault corrections for the release in question. (Rajlich & Bennett 2000)

What Changes are estimated in Projects Life Cycle?

There can always be some changes in the product life cycle which will alter the development of product dramatically. These changes can, for example, be changing the development network, transferring development to another country or to a subcontractor.

If there is a possibility that a product will be transferred to a subcontractor or that some part of this product is developed at the subcontractor, it is necessary to take this into account. This situation can require that the SCM solution will support special requirements like, for example, connection types that will not require modification to firewall settings between intranets.

What is the Life Cycle for Product or for Release?

Answers to this question cover two main things. What is the estimated time from the start to the end of the product development period? This defines the whole length of a product's life cycle. Which affects the effort that is put into designing and implementing the development environment? For example, when a product life cycle is estimated to be long, the development environment must be designed well so that it will require as little as possible maintenance and re-planning. In cases where there is a short product life cycle, the effort spent on designing and implementing the development environment must be optimized to meet the criteria for the development project. The short life cycle can mean several things. There can, for example, be plans to combine this product to be a part of another product in future or this product is created simple to test market interest.

Secondly, the percentage of the development and maintenance periods in the release development should be defined. If the development period is notably longer than the maintenance period, the cost structure must be designed to pay attention so that most of the life cycle is a non-profitable time. Figure 18 overleaf presents differences between lengths of release life cycles.

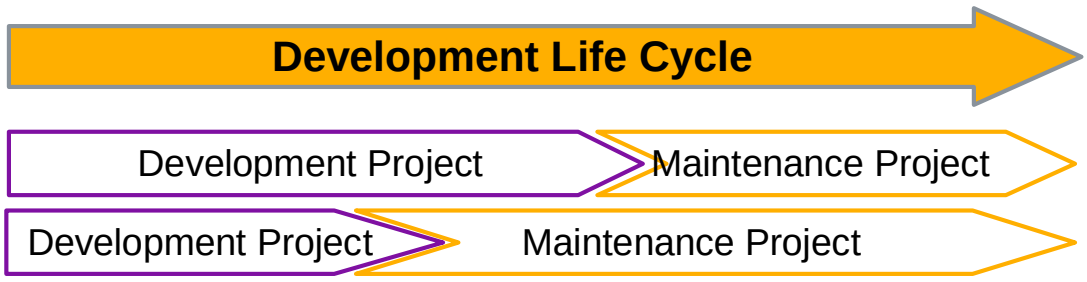

Figure 18: Differences between lengths of Release Life Cycle

What is the Maintenance Responsibility when Ramping Development Down over a 2-15 Year Period?

A long product maintenance period can be a legal issue or depends on contracts with customers. The duration of the maintenance period influences the selections of tools and methods if maintenance period is remarkable long. This is, because if the development of a product is ended unexpectedly, the cost of the maintenance period can become considerable large. In cases of long maintenance periods there can be the need to change tools to more cost-effective ones and rethink the processes to lower costs. When the period in question is short, it can be more profitable to keep the existing solution in use.

Is there a Need for Multiple Releases under Development at the Same Time?

A new development project for next release should start before the foregoing release is at the end of development, offering a new release branch to develop new features. In this way integrity of the releases released can be ensured. There can be a main release, a development release for some feature and maintenance releases under development at the same time. Running concurrently there needs to be some king of bug fixing development which creates need for branching. Multiple development branches create the need for more control because every repair made to a previous branch must be merged to the new development branch. (Rajlich & Bennett 2000) Easy merging and content control between different releases is essential when there are several releases under development and maintenance at same time. This situation is presented in Figure 19 overleaf.

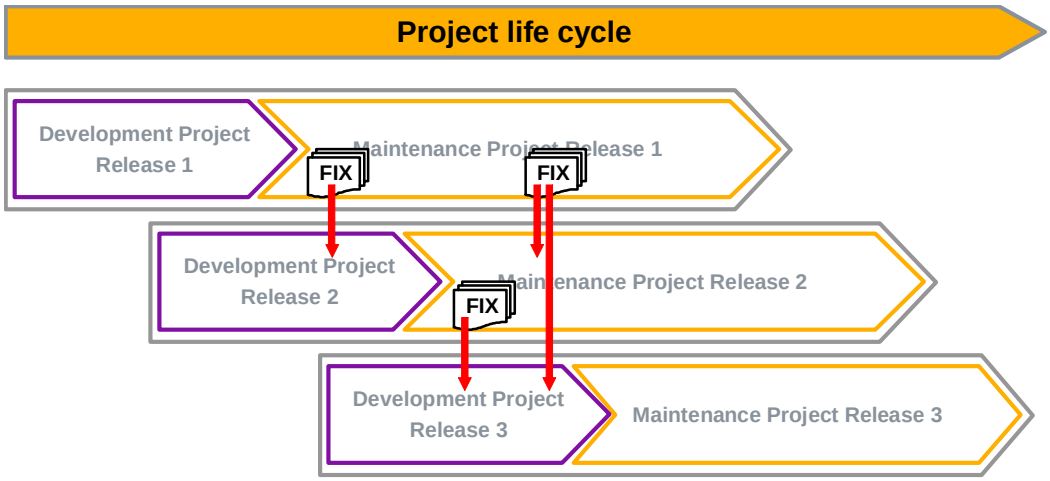

Figure 19: Development and Maintenance Projects in SW Product's Life Cycle

Research Results

UCM and Dimensions offer a great set of features to support different requirements along the development life cycle. Subversion, CVS and Clearcase require additional tools or scripts to provide added functionalities for the development environment. Subversion and CVS are Open Source tools that enable in-house development for these tools, but extensive development of features can be time-consuming as well as expensive. There are also a great many of different Open Source tools developed to run on top of Subversion and CVS.

All the VCS tools tested support CI development and enable fast response times in software building. This is because CI development processes are simple and demand little from the VCS. The Main CI development work in the SCM field is done on top of Subversion. Subversion can be used with, for example, the Open Source continuous build process framework *CruiseControl* or commercial *Atlassian Bamboo*. Dimensions includes its own continuous build process framework among its features. The CI development model can be exploited in the other VCSs studied with either *CruiseControl* or *Bamboo*. This, however, requires creation of interface that supports interactions between these tools.

9.11 Authority Responsibility and Permissions

Authority relates directly to the management of development projects. Individuals responsible for development project are, for example, project managers as well as

other managers like development-, verification-, product-, quality- and SCM managers. These individuals are responsible of the development methods and disciplines which regulate all aspects of the development work. It is their responsibility to state how different tasks and function must work and what the requirements for development are. (ANSI/IEEE 1042-1987 1987)

Planning responsibilities and setting permissions are crucial tasks for any SW project because these rules define a major part of the workflow at the personnel level. Development model specifies how different tasks are carried out in software development and defined responsibilities specify who is responsible for these different tasks. Responsibility spans the whole project organization from the SW product's management to permissions to make changes to software components. Permissions realize these responsibilities in tools that are used in the development environment.

What is the level of Authority in the Project Organization within which the Software is being Developed or Maintained?

The project management style reflects directly on the control of the SCM, because the project's management states how controlled the development must be. Things that project management wants to requlate are, for example, authorities that are responsible for files related to the development project like source code and configuration files, documents like product documentation, source code history information, development plans as well as baseline content. The project management has information about all the parts of the development organization. The project organization can be distributed across multiple development sites and its different parts like I&V, SCM or some part of the coding can be outsourced to a Collaborator Company. The level of authority may vary depending on the individual development phase in question. In a development phase where features are made, authority can be lower and just before releasing the product for customers, authority can be much stricter and every change is strictly validated and tested. (ANSI/IEEE 1042-1987 1987)

Things that interest SCM service are those that influence the configuration of SCM solutions. These kinds of subjects are, for instance, responsibility for source code

and history management, as well as partner access to the development environment because these things are configured in systems that are under the SCM's responsibility.

Who Grants Permissions and in which Principles?

SCM service can control permissions for development projects in several issues. Different subjects that are controlled with permissions can be, for example, software components, documents, servers, tools and computers. The SCM should be an admin to the VCSs that holds all the software components and documents. This is because the SCM provides access and control to the VCS in the development project. The SCM can have also administrative role in the servers that are exploited in the development organization, like file servers that include software packages and other code compilation results. The development environment can be under the control of the SCM if the whole development uses the same tools and servers. This can be a case when SW development has tools like Dimensions or UCM in use because these tools provide functionalities for all areas of development. The same development environment can include tools from text editors to SCM compilation and testing systems.

Is Development Responsibility Shared through Development Sites or is Development Divided between Development Sites?

In software development, responsibility can be divided between more than one site and in one site responsibility can be divided between multiple teams or persons. There can be different kinds of needs to give commission rights for developers to the repository. In development projects, all developers can have permission to commit software changes to all software components. There is also a need to restrict these permissions by defining component responsibilities. In other words, the developer or development team can have permission to change only some defined software components and have read-access to other ones.

Different parts of development like implementation, SCM or testing can also be located in multiple development sites. This creates new management requirements

compared to basic one site development. For example, development administration must have processes and tools to control multisite situations.

How the Permissions to the Repository are configured?

Repository access in the SW development environment reflects development responsibilities in the SW development project because access and change rights to the repository can be granted depending on developers' responsibilities. It is important that permissions for repositories can be defined flexible, because there can be a need to have different kinds of permission sets. For example, in Agile development, the development model defines that every developer can make changes to every SW component in the repository. In some cases, responsibilities are defined for different development groups and in this case, permissions should be granted at group-level. If there are individual responsibilities, it is important that permissions can be granted at the developer-level.

Are There Separate Development Projects for Each Subsystem?

In large and complex development projects, there is a need to have several development projects to simplify the development. These projects can be separated between different features of developed SW or development teams in different development sites. In distributed projects, SCM procedures must redefined for each part of the project. (Kääriäinen 2006) Good information sharing procedures need also to be planned when there are many separate development projects. There can, for example, be two in-house development projects in different sites and a third development project is done by the subcontractor. Information procedures between two in house development projects can be easier to manage than information procedures between different companies.

Research Results

Each one of the studied VCSs can be configured from strict to open type permissions. All the developers in the SW development can have permissions to all the SW components in each of the studied VCSs. This is the basic requirement of CI development. In *Dimensions*, it is possible to give a different status for each of the items in the repository to manage the update process. There can, for example,

be inspection steps before an items acceptance. Administrators of the VCS and repositories can be defined in each of the studied VCSs.

With Subversion permissions can be well-defined, but in a multisite environment this depends on the multisite solution used. *Pushmi* is based on shared responsibility and it is desirable that the network connections are good and stable. Updates to databases should be fast to avoid conflicts because changes can be made from each multisite location to the same component. *Pushmi* is the best solution for shared responsibility in a multisite situation because there are no limitations for mastership. *Svnsync* only supports divided responsibilities between replicated databases. Clearcase and UCM can be configured to support different kinds of responsibility solutions. Dimension supports both kinds of responsibility solutions. Permission to access the repository can be configured well in each VCS tool and permissions to different items in the repository can be determined separately for individuals or groups.

9.12 Third Party Software

Third party software is developed by a Third Party Company. This kind of company provides an auxiliary product not supplied by the primary manufacturer to the end user (Computer Desktop Encyclopedia 2008). By using third party software, innovative solutions can be exploited by different companies. In this way, the company is not tied to one supplier’s products and services.

Dependency on Third Party Software Code and Tools

SCM projects use different kinds of third party tools like compilers, tools and operating systems. The same kind of behavior can be found, for example, in software project using third party software components. It is important to have principles for controlling third party software by defining what will be included, how it will be delivered, accepted and implemented to the tools and methods of the SW project. Third party software code is much harder to control. Additionally, if there is a need to change the code, it is more difficult to get it implemented than when working with in-house software code. There can also be some limitations for

seeing the content of third party SW code. Their release cycles are also probably different from the SW developers release cycles. (Kääriäinen 2006)

Research Results

UCM and Dimensions do not require additional tools because of a versatile features set, but these tools do not restrict the use of different tools for some tasks. However, the use of third party tools with these SCCM tools can reduce efficiency because features included in these tools are designed to function together and additional tools can break the compatibility of different integrated features. Subversion, CVS and Clearcase need additional tools to offer other features than the basic VCS functionality.

Third party software can be used with the VCSs studied with certain limitations. Compiled third party software is not always fully compatible with used VCSs' build systems. This is when design and realization of this software do not respond to the file system used. For example, feature *winkin* in Clearcase does not work properly if the file system of the derived objects does not match the model of the VCS. *Winkin* provides access to one or more already compiled components from a dynamic view when recompiling SW items.

9.13 Open Source

The Open Source software "movement" has received enormous attention in resent years. It is stated to be a new way of developing software. By definition, Open Source licenses must meet 10 different conditions (Perens 1998). These conditions state, for example, that Open Source software code is freely distributed and everybody has rights to use, change, develop and redistribute it. Any user or commercial party cannot be excluded.

Is Open Source Involved in Development Environment?

When using Open Source software as a part of the development environment, its nature is very important to understand because there is a significant difference when compared to using of commercial products. With commercial products there is an interface to a tool provider that can develop a new feature if needed. With an

Open Source product this kind of interface does not exist to be easily exploited, but all the software's source codes and rights to develop and change it are, however, available. In this way, there can be better control of the product's features. On the other hand, there will be more feature and maintenance responsibility issues. Commercial products can have restrictions or poor interfaces to other tools that might be needed to use as a part of the development environment. Open Source products provide the possibility to integrate tools into the same development environment. For example, VCS Subversion has multiple Open Source tools available to enhance Subversion's features such as TortoiseSVN, Svnsync, Pushmi and CruiseControl. TortoiseSVN is a GUI which integrates Subversion into Windows Explorer browser. The other tools Svnsync and Pushmi provide replication solutions between Subversion repositories. CruiseControl is a continuous build process framework.

Research Results

Subversion and CVS are Open Source VCSs that exploit other Open Source software to enhance the feature set of these tools. The features of Subversion are continuously improving whereas development of CVS is at more of a standstill. This is because most of the development work done in the Open Source community is more focused on the newer Subversion tool. Clearcase, UCM and Dimensions are all commercial tools that require the purchase of a software license before using it.

9.14 Configuration Identification Scheme

Configuration identification is one of the most important tasks for SCM Service because the identification scheme is reflected directly in the structure of the final product. It is critical because management control is based on the structure of the product and the identification scheme carries through the life time of the product. (ANSI/IEEE 1042-1987 1987) Software configuration identification consists of all the things that can be distinguished to be an entity, such as SW items, different baselines or other documents. The naming and versioning model must be defined when creating the identification scheme for different items. (ANSI/IEEE 1042-

1987 1987) This policy defines how different items are recognized and the version model defines how evolution of these items can be tracked.

The configuration identification describes activities including determination of the product structure, selection of the configuration items, documentation of the configuration item's physical and functional characteristics including interfaces and subsequent changes. Finally the allocation of the identification characters or numbers of the configuration items and their documents are designed. (Kääriäinen 2006; ANSI/IEEE 1042-1987 1987)

What Project Baselines are identified?

The baseline model is defined in the identification scheme. That specifies how these baselines are labeled and in what principles different entities like SW items, builds and releases obtain baselines. In SW projects, baselines allow many people to work together in a synchronized fashion by setting a set of rules how the life cycle of a SW product is managed and identified. (ANSI/IEEE 1042-1987 1987)

What is the Identification Scheme for Items in SW Project?

SCM has traditionally held responsibility for managing source code related elements, like SW components, binaries and compilers. However all the items handled during the development process should also be managed by using defined identification schemes and placing those under some kind of VCS. These different items can be divided into three different categories:

1. Development environment (HW and operating system)
2. Tools
3. Software code (SRC)

All items that should be also identified and versioned are, for instance, management and customer documents, designs, manuals and instructions. These items have different physical characteristics and it affects their identification and control. Different items need different naming principles, and documents can hold pictures and tables that are not supported in text based VCSs, reducing the

efficiency of these systems. There can be a need for separate VCSs for different types for items. (Kääriäinen 2006)

What is the Versioning Model for Releases?

The versioning model for releases is most visible for customers because different releases are separated from each other with this identification. When changing the VCS in a SW product development or making changes to the VCS, a continuum of release versioning should be ensured to avoid confusion at the customer side.

Research Results

Identification is not an issue in any of the studied VCSs. This is, because the development and build processes can adapt to changes in any items identification scheme. Identification of baselines and releases can be defined freely in each of the VCSs.

9.15 Performance

Every function in software development must provide good response ability to enable a well functioning development environment and organization. Every development function consists of many factors that affect performance. SCM service, for example, can make its services more effective by choosing effective tools, improving processes by exploiting advantages of these tools and simplifying the workflow for performing different tasks, as well as automating tasks.

How quickly do Fault Corrections need to be in the Software Package?

The definition of what is “critical” in software development is different when the product is on the markets or when it is only intended for research purposes. This is because every fault discovered in the product by end customers must be corrected as soon as possible. In the telecommunications industry, faults in software can paralyze large parts of networks and millions of mobile users lose their connectivity to the network. This all means significant losses to operators and a bad reputation for network providers.

The main differences in performance come from the methods used and level of automation, than from the tools employed in the development environment. This is why performance requirements relate more to the SCM usage model and automation than to the particular VCS in use. There are however, differences in speed, reliability and in the process models of different VCSs. Reliability of repository operations depends also on other areas like servers and networks connections. It is crucial to take into account reliability of operations and methods, because if some tool crashes or some component corrupts during a file transfer, a software package cannot be produced and delivered on time. This means that delivery time for the build will be delayed.

Research Results

Today Subversion and CVS offer the best solution for Agile SW development with continuous build process frameworks like *CruiseControl* or *Bamboo*. Dimensions also offers a build process framework in its latest release 10. Agile development is not that easy with Clearcase-based models, because there are no realized models. Implementers must make the effort to create the configuration of the available toolset to meet the Agile development's criteria. Lee (2006) lists seven criteria in IBM's publication that must be taken into account when using Clearcase in Agile development environment: implement a simplified branching strategy; use snapshot-view developer workspaces; automate the build process; stage and re-use pre-built binaries; release as a composite application; limit the use of multisite technology and; avoid over-customization of ClearQuest change control. From our point of view, this methodology undermines some of Clearcase's strengths like branching and dynamic view.

Clearcase's winkin can significantly improve performance of the compilation time of the SW build. This is, because winkin provides access to pre-compiled SW components. For example, in the SW build's update process, only new components are compiled and previously compiled components are used from storage. This same feature can also be exploited in Subversion with additional software. The tool *ccache* speeds up C/C++ compilations with caching compilations and tool *distcc* distributes C/C++ compilations across several machines on a network. (Tridgell 2008; Pool 2006)

9.16 Reliability

Reliability is a very important thing to consider in all SW development. What are the main environmental risks of this product? Can the product paralyze a large part of the Internet or company's intranet by its faulty operations? A more important question is whether somebody could get killed because of a software bug? For example, healthcare systems in hospitals need to be fully-functional every fraction of every second 24 hours a day and seven days a week.

How Reliable does the Product Need to be Regard to Safety Issues?

Different kinds of reliability requirements push SCM service and require that its reliability and responses are well designed and executed. It is very important to choose tools that are reliable and stable to use and exploit these tools with well-designed methods which take into account characteristics of a specific product. One viewpoint is also how different models behave when network connections are unavailable.

Is Product Only for Testing Purposes or is it a Commercial Product?

If a product is developed only for internal testing purposes, it can have lover requirements for stability and reliability than a product that is destined for use on customer premises. In demo environments, cost efficiency needs to be considered and it is not worthwhile to realize all product requirements. The reason for this is that every new product does not receive market approval and can be ended. Nokia Research Center could be an example where products are realized for testing to determine if these products or ideas have any business potential or not. That is why this kind of testing must be profitable.

Research Results

With all the studied VCSs, different functions can be configured to forbid undesired user actions like deletion of frozen baselines, directing developers to use the desired usage model. With all the studied VCSs, it is important to make configuration carefully to be sure that the tool works the same way the development process has defined it to. Items added to the Subversion repository

cannot be removed afterwards even if needed. In Clearcase based solutions this is, however, possible to do.

The VCS tools, *Subversion*, *CVS* and *Dimensions* use offline clients that enable offline development when network connections are unavailable. There is also an offline development possibility with Clearcase, but the online model is generally used, because the development with Clearcase is based on online connectivity to the repository.

9.17 Audits and Information Sharing for other Development Functions

SCM service has strong linkages to different R&D functions due to its service nature. These interfaces need to be well-designed and executed to get the best from SCM service inside SW development processes. SCM's different interfaces to the R&D project can consist of all the functions that R&D has, such as, for example, SW architects, developers, SW management, I&V as well as customer care. (ANSI/IEEE 1042-1987 1987)

Other SCM teams are also important high-level interfaces that can bring added value. Collaboration experiences and knowledge can be shared across the company's development projects. (ANSI/IEEE 1042-1987 1987) Naturally, it can not be forgotten that outside company borders, there are new innovations and methods that can bring competitive advantage to a company's business. Other interfaces that are useful to follow are tool developers like IBM, Collabnet and the standardization organizations like IEEE and ISO (Institute of Electrical and Electronics Engineers).

What Kind of Reports Needs to be generated and for whom?

SCM has connections to different development functions. In SW development, content of the SW product will evolve and the SCM needs to keep other development functions in touch with the products state. Different functions that the SCM needs to take into account are, for example, project managers, developers, SW architects, I&V personnel as well as customer care. Different subjects that should be audited are, for example, available software packages, announced software components for build and release, content of builds and releases and SW

metrics for differences in source code like codeline and complexity calculations. Here, for instance, project management may be interested in the changed components in different releases and how development has stayed on schedule. Developers may be interested in component changes and the latest releases that are available for component testing.

What is the Level of Audits?

A software product is a complex entity that consists of multiple components. All of these components and configuration files are attached to the SW and changes are made by someone. To ensure that the product has the right content, all the changes to the SW must be somehow tracked. (ANSI/IEEE 1042-1987 1987) There, for example, can be a system that connects different changes to features implemented in the SW. UCM is based on this kind of change tracking, because all the changes are made in pre-specified workspaces called *activities*. All the changes are made and stored in these activities and activities are included in specific software revisions. (White 2001)

How History Information will be exploited?

In SCM processes, all kind of history information is generated. These are, for example, which modules have been in different releases, which developer has made the changes to different components, what changed are between different versions of a software component, when different features will be realized in the software, etc. History information is important because it enables tracking of the SW's development.

How Third Party Software Code should be Reviewed and Audited?

When using third party software code as part of in-house code, the content and compatibility must be validated, to ensure the software's integrity. Methods to control third party code differ depending on the situation because different situations have different types of requirements for auditing. For example, audits can be similarly used for in-house software code in situations where third party software code is developed under the same development environment and processes. In situation where in-house SW development cannot fully contribute to

the content of third party software, the development organization must ensure that in-house code will be compatible with the code available. In this situation, the content and interfaces of both codes must be reviewed to ensure the software's integrity.

Research Results

This requirement is more of a process requirement, but different reports that can be made with VCSs assist when gathering information. Subversion and CVS include only basic reporting types whereas UCM and Dimensions have a lot of metadata where different kinds of reports can be generated. Clearcase can also generate different kinds of reports, but does not enable as extensive a set of different reports as UCM and Dimensions can provide.

History information can be exploited in every VCS studied. With base Clearcase, UCM and Dimensions, chances can be followed by activities. Activities are made for all phases of the development process. They can represent, for example, feature development and bug fixing. For each activity, can be seen which components have been changed when changes are made and what this activity represents.

When a third party uses the same development environment that the in-house development has, the content of the SW can be audited with the same processes that in-house code is audited with. When the development environments are different however, it is more important to ensure the third party software's content at a process level. The reason for this is, that visibility to the content of the SW can be lost and reports from third party vendors are the only sources to check the compatibility of the SW.

9.18 Other Matters to Consider

Features that are included in this chapter do not affect directly on the SCM work and are not basic tasks of the SCM. However these features are important parts of the development environment and should be taken into account by SCM service.

How Backups will be done?

Making backups is a repository-centered issue and this way does not relate to the VCS tools. However, this issue is taken into account because it is one of the most important things to be utilized in a software product. This is because the SW code and components stored in repositories need to be secured for permanent and unchanged data. (ANSI/IEEE 1042-1987 1987)

How easily is VCS Maintenance?

Maintenance of VCS is an important task that can provide a reliable system, because the VCS need to be functional all the time. This means that updating the VCS should not interfere with executed operations in repositories and the faults of the VCS should be solved quickly as well as be unrecognizable.

Research Results

Each of the VCSs studied has the possibility to make backups. There can be specified functionality like in Subversion (SVN-handbook 2004), scripts or using dedicated backup tools. There is also the possibility to make a backup of the database's content where the repository is placed. Making backup copies can become more challenging in multisite repository situation because data is not always fully synchronized between the different repositories. However, the multisite situation in some cases can behave like backup copying. In Subversion's Svnsync each repository holds a copy of the central repository and in this way is a good candidate to replace the main repository. (SVN-handbook 2004)

VCS maintenance has a low effect on research and development (R&D) when using Subversion, because of a dedicated backup functionality. There is not a need to close the repositories before making any maintenance work to the repositories. With Clearcase based tools, the repositories must, however, be closed during maintenance. There was no experience of making backups using the VCS tools *CVS* and *Dimensions*, because responsibility of secure backups was with a separate database administrator. In addition to backups, Clearcase and Dimensions require a great amount of administration work, because of their complexity. Subversion, on the other hand, is easy to maintain because of its simple construction.

10. Conclusions

This chapter combines the results found in this thesis and introduces two question lists that can be used when designing any SCM solution for any SW product. There are two different lists because evaluation can be made for a new SW product or for one that has already been under development. The basic questions presented in Chapter Nine have evolved into compact questionnaires for SCM to exploit. These questionnaires are included in this chapter. It is also possible to use all the questions presented in Chapter Nine to collect a larger amount of data from the SW development environment. Furthermore, this chapter collects together the main points of how the different VCSs under study functioned with the requirements for SCM in order to the answer the research questions.

10.1 Questionnaires Created Based on Study

The outcome of this thesis is two questionnaires that support the selection process for any SCM solution. The first questionnaire is used when the SCM solution is designed for a new SW product and the second one is used when a SW product has already been under development and uses some SCM solution. The questionnaires were created based on the requirements of SW development for SCM presented in Chapter Nine and evaluated with the SCM personnel inside NSN. These questionnaires are designed for the SW project's management as well as specialists. The best result is gained when the results of these questionnaires are compared to the evaluation of the VCSs. Questionnaires with question titles and questions are presented in Table 2 and Table 3.

Table 2: Questionnaire when designing SW solution for new SW product

Key term		Question for project organization
1	*Starting base*	Are there plans to import existing code from some program?
2	*Software basics*	Technical data of the product: What kind of product it is? (Application/platform, component based, SCM architecture hierarchy, multiple customer versions,..)
3	*Quantitative information*	How large the software project is? (How many and how big software components if any components exists yet, how many developers inside and outside NSN)?
4	*Programming language*	What will be the programming languages in development project?
5	*Multisite and Replication*	Is there need for multisite development and how it should be arranged?
6	*Operating system*	On what operation system do development computers, SCM tools and the target system run on?
7	*Version control tool*	Is there any existing SCM infrastructure yet?
8	*Tools existing and tools needed*	What is the development projects state now from the tool perspective? What are the tools that will be used in SCM and implementation?
9	*Development process*	What will be the development process?
10	*Partners*	Is the development project done in-house or with partners? Who are the partners?
11	*SCM in project organization*	Will SCM belong to its own project or will it be a subproject for someone else?
12	*Competences*	What kind of experience does the development projects organization have? Are some version control system or development methods familiar already?
13	*Product lifecycle*	What is the lifecycle for the product?
14	*Development lifecycle*	What are the development and maintenance policies for the product? Is there need for any merge operation?
15	*Change management*	What will be the Change and Fault management policy and process?
16	*Permissions*	Should every implementer have permissions to every software component? Who should grant permissions for software components?
17	*Third-party software*	Will there be any third-party software in the development project?
18	*Open Source*	Will Open Source be involved?
19	*Identification scheme*	Is there an existing versioning or naming model for the software components and for releases?
20	*Audits*	How does the content of the software build needs to be supervised and what kind of reports are needed for the development projects management and for the rest of the development organization.
21	*Response times*	How quickly does SCM need to react to requests from the development organization and how quickly deas the development organization need to react to customer request?
22	*Authority/ Responsibility*	What is the information that is needed when implementation checks change in the software? Is s change note needed?
23	*Managing high-level interfaces*	How SCM does need to communicate with rest of the development project organization?

Table 3: Questionnaire when designing SW solution for existing SW product

Key term		Question for project organization
1	*Products existence*	At which milestone are you now?
2	*Starting base*	Are there plans to import existing code from some program?
3	*Software basics*	Technical data of the product: What kind of product it is? (Application/platform, component based, SCM architecture hierarchy, multiple customer versions...)
4	*Quantitative information*	How large the software project is? (How many and how big software components (if any exists yet), how many developers inside and outside NSN)?
5	*Programming language*	What will be the programming languages in development project?
6	*Multisite and Replication*	Is there need for multisite development and how it should be arranged?
7	*Operating system*	On what operation system do development computers, SCM tools and the target system run on?
8	*Version control tool*	Is there any existing SCM infrastructure yet?
9	*Tools existing and tools needed*	What is the development projects state now from a tool perspective? What are the tools that will be used in SCM and implementation?
10	*Development process*	What will be the development process?
11	*Partners*	Is the development project done in-house or with partners? Who are the partners?
12	*SCM in project organization*	Will SCM belong to its own project or will it be a subproject for someone else?
13	*Competences*	What kind of experience does the development projects organization have? Are some version control system or development methods familiar already?
14	*Product lifecycle*	What is the lifecycle for the product?
15	*Development lifecycle*	What are the development and maintenance policies for the product? Is there a need for any merge operation?
16	*Change management*	What will be the Change and Fault management policy and process?
17	*Permissions*	Should every implementer has permissions to every software component? Who should grant permissions for software components?
18	*Third-party software*	Will there be any third-party software in the development project?
19	*Open source*	Will Open Source be involved?
20	*Identification scheme*	Is there existing versioning or naming model for the software components and for releases?
21	*Audits*	How does the content of software build need to be supervised and what kind of reports are needed for the development projects management and for the rest of the development organization?
22	*Response times*	How quickly does SCM need to react to requests from thedevelopment organization and how quickly does the development organization need to react to customer requests?
23	*Authority/ Responsibility*	What is the information that is needed when implementation checks change in the software? Is a change note needed?
24	*Managing high-level interfaces*	How SCM does need to communicate with rest of thedevelopment project organization?

10.2 Answers to the Research Questions

All answers to the research questions asked are compiled here in this chapter. The answers are combined from both parts of this thesis.

What does a Software Development Project require from Nokia Siemens Networks SCM Service Department?

The basics for SCM services are stated in Chapter One. SCM service must provide processes for six different aspects.

- Software builds are generated in SW Building
- SCM development improves SCM processes and tools
- SCM administration maintains SCM processes and tools
- Support for collaboration directs the work of collaborators and connects them to the SW development environment
- SCM Project Management processes plan SCM work
- SCM Consulting aspires to help different SW development functions in software configuration related questions

How can the SCM Service Department Support the Development of a Software Product?

Basically, SCM service need to offer a stable SCM environment that will serve the requirements from the SW development project. In Chapter Two, there are seven different view points that provide for a well-functioning SCM process: safety, stability, control, auditability, reproducibility, traceability and scalability.

How do existing SCM Version Control Systems serve Software Development?

As a conclusion, CVS is not as good a VCS as others in this evaluation as was expected at the start of this study. This is mainly because Subversion has replaced CVS and main tool development in the Open Source community is centered around Subversion. Subversion is a standalone VCS, and with additional tools like Svnsync, Pushmi and TortoiseSVN, it proved to be able to handle SCM responsibilities very well. When there is a well designed usage model, Subversion is very easy and cost-efficient to implement in SW projects which makes it very desirable to use. In addition Subversion was proven to be the best VCS in CI development.

Clearcase proven to be a good VCS, and with UCM functionalities SCM responsibilities are handled more broadly and easily with the help of a well-designed GUI. At Nokia Siemens Networks, SCM usage model 1 activity scripts handle the SCM responsibilities in the SW project in question well, but maintenance of these activity scripts requires large amount of competence.

Dimensions were proven to be quite similar to UCM, but some of the features could not be evaluated because many features where not utilized and exploited in the SCM usage model. Both of these VCSs' possess a great amount of features, but UCM is more intuitive to use. There where comments that "Dimensions have too many features inside the same QUI". Different features for different development functions like SCM, implementation and project management are placed under same QUI, which makes understanding of the process very complex.

As a conclusion Subversion, Clearcase and Dimensions all have the potential to manage SCM processes well. Inside NSN, Subversion and Clearcase have the greatest potential to be implemented in different SCM processes because of their sophisticated SCM usage models. In addition, in the CI development environment Subversion is the one to beat, for four main reasons. Firstly there are readily available CI development solutions on top of Subversion; it operates efficiently; it is easy to maintain and; finally it is free to use and modify.

10.3 Future Research Possibilities

This study created the basis for usage model selection process in SCM that can be developed further. The next step could be a system that proposes the best possible models when giving the characteristics of SW development projects as an input. This will require more accurate data for each VCS and SCM usage model studied and a scale that evaluates different models against each other. This way there could be the possibility to make, for example, a webpage that asks questions from SW development projects and as an output gives the best solutions for version control systems and SCM usage models. Making this kind of system is, however, very challenging, because SW development projects include large amounts of variables that all can influence the SCM work in different ways.

References

Afora International, 2008. *SCM Glossary, web reference, <http://www.afora.nl/index.shtml>.*

ANSI/IEEE 1042-1987, 1987. *IEEE Guide to Software Configuration Management, IEEE, 1987.*

Answers, 2008. *web reference, <http://www.answers.com/topic/waterfall-model?cat=technology>, May 15, 2008.*

Armstrong David J. & Cole Paul, 2002. *Distributed Work, Managing Distances and Differences in Geographically Distributed Work Groups, MIT Press, 2008, <http://books.google.com/books?hl=en&lr=&id=3aAwmlgC7YMC&oi=fnd&pg=PA167&dq=geographically+distributed+development&ots=3O89Q4gkeC&sig=9fGJVVUWlV7OAm8lFbGjXpUq-Kc>*

Appleton Brad, 2004. *CM//Crossroads, The Agile Difference for SCM, web reference: <http://www.cmcrossroads.com/content/view/6724/202/>, May 15, 2008*

Appleton Brad, 2006. *Dimensions and Views of SCM Architecture, web reference, <http://planetscm.org/user/4/tag/scm-principles/>, May 15, 2008*

Chia-liang Kao, 2007. *Pushmi - Subversion replication system, web reference, <http://tokyo2007.yapcasia.org/sessions/2007/02/pushmi_subversion_replication.html>, video, YACP Asia, May 15, 2008*

Clemm Georffley M., 2000. *Software Configuration Management Strategies and Rational ClearCase®, Addison Wesley*

Collabnet, 2007. *Subversion internet homepage, web reference, <http://subversion.tigris.org/>, Collabnet, May 15.2008*

Collabnet, 2008. *Subversion Packages, web reference, <http://subversion.tigris.org/project_packages.html>, Collabnet, May 15, 2008*

Collins-Sussman Ben, Fitzpatrick Brian W., Pilato C. Michael, 2007. *Version Control with Subversion versio 1.4, Chapter 9. Svnsync, web reference, <http://svnbook.red-bean.com/en/1.4/svn.ref.svnsync.html>, May 15, 2008*

Conradi Reidar, Westfechtel Bernhard, 1998. *Version models for software configuration management, ACM, <http://portal.acm.org/citation.cfm?doid=280277.280280>*

Computer Desktop Encyclopedia, 2008. *Library, web reference, <http://www.answers.com/topic/third-party?cat=biz-fin>, May 15, 2008*

Duggan Jim, Stang Daniel B., 2008. *Magic Quadrant for Software Change and Configuration Management for distributed systems, Garther*

Ferguson John, 2005. *An introduction to Maven 2 – How applied best practices can optimize the Java build process, JavaWorld.com, <http://www.javaworld.com/javaworld/jw-12-2005/jw-1205-maven.html>, May 15, 2008*

Friedman, Daniel P. Wand Mitchell, Haynes Christopher T., 2001. *Essentials of programming languages, second edition, MIT Press*

Haifeng Shen and Chengzheng Sun 2001. *Operation-based revision control systems, School of Computing and Information Technology Griffith University*

Herbsleb James D., Paulish Daniel J., Bass Matthew, 2005. *Global Software Development at Siemens: Experience from Nine Projects, ACM, <http://portal.acm.org/citation.cfm?id=1062550>*

Huxman Chris, 1996. *Creating Collaborative Advantage, SAGE, 200 p, <http://books.google.com/books?hl=en&lr=&id=KrkolOWfMZgC&oi=fnd&pg=PR7&dq=collaborator+development&ots=sulhyI8kiU&sig=NGWOYX7d5mMpPR6m68OvFpvfOvQ#PPP1,M1>*

IBM, 2003. *Rational Clearcase Multisite – Administrator's Guide, Version 2003.06.00, IBM Corporation, 333 p*

IBM, 2004. *SCM 270 Essentials of Rational Clearcase for Windows v2003.06.00, Student Guide, IBM Corporation, 260 p*

IBM, 2007. *IBM Rational Clearcase, IBM, <ftp://ftp.software.ibm.com/software/rational/web/datasheets/clearcase.pdf>*

IBM, 2008. *System Requirements for ClearCase 7.0.x, IBM <http://www-1.ibm.com/support/docview.wss?rs=984&uid=swg21239315&S_TACT=105AGX15&S_CMP=LP>*

Herbsleb James D. and Moitra Deependra, 2001. *Global software development, Lucent Technologies India, <http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=914732>*

Kokkonen Antti, 2003. *Evaluating the Software Configuration Management Process, University of Tampere, Tampere, 52 p*

Koskela Juha, 2003. *Software configuration management in agile methods, Otamedia Oy, VTT Espoo, 48 p*

Kääriäinen Jukka, 2006. *Practical adaptation of configuration management, Otamedia Oy, VTT Espoo, 69 p*

Lee Kevin A., 2006. *Agile SCM and the IBM Rational toolset, IBM, webpage: <http://www.ibm.com/developerworks/rational/library/jun06/lee/>*

Maven, 2008. *Maven's internet homepage, Apache Maven Project, <http://maven.apache.org/>*

Perens Bruce, 1998. *The Open Source Definition, <http://ldp.dvo.ru/LDP/LGNET/issue26/perens.html>*

Pool Martin, 2008. *distcc: a fast, free distributed C/C++ compiler, web reference, <http://distcc.samba.org/>, May 05,2008*

Pour Gilda, 1998. *Technology of Object-Oriented Languages, San Jose State University*

Rajlich Václav T and Bennett Keith H 2000. *A staged model for the software life cycle, Wayne State Univ., Detroit, <http://ieeexplore.ieee.org/iel5/2/18837/00869374.pdf?tp=&isnumber=&arnumber=869374>*

Royce Winston W., 1970. *Managing the development of large software systems, IEEE WESCON, 9 p*

Serena Software, 2007. *Serena Dimensions 10, Serena Software, web reference, <http://www.serena.com/docs/repository/products/dimensions/control-application-.pdf>*

TechRepublic, 2006. *Understanding the pros and cons of the Waterfall Model of software development, TechRepublic, web article, <http://articles.techrepublic.com.com/5100-10878_11-6118423.html?part=rss&tag=feed&subj=tr>*

Tichy Walter F., 1982. *Design, implementation, and evaluation of a Revision Control System, IEEE Computer Society Press, 10 p*

Tichy Walter F., 1988. *Software Configuration Management Overview, 26 p <http://grosskurth.ca/bib/1988/tichy-tools.pdf>, May 15, 2008*

Tridgell Andrew, 2008. *Ccache, web reference, <http://ccache.samba.org>/, May 15, 2008*

Westfechtel Bernhard, Conradi Reidar, 2001. *Software Architecture and Software Configuration Management, IEEE Trans. Software Engineering, 16p*

White Brian A., 2001. *Software Configuration Management Strategies and Rational ClearCase: A Practical Introduction, Addison Wesley, 286p*

Appendices

Appendix 1: Factors Affecting the Configuration Management Solution

This appendix is based on reference (Kääriäinen 2006). It presents factors affecting the configuration management solution that Kääriäinen has listed at his study. Factors listed are:

1. Size of the Project
2. Product Type
3. Project Hierarchy (Distribution)
4. Multisite Development
5. Development Disciplines
6. Development Model
7. Dependence on Third Party Software
8. Maintenance and Multivariants
9. Item types
10. Management Constrains on the CM Plan

Appendix 2: Content of Software Configuration Plans

This appendix is based on reference (ANSI/IEEE 1042-1987 1987). It presents what should be the content of SCM plan based on standard ANSI/IEEE Std 1042-1987 by IEEE. Different sections of SCM plan listed in standard are:

11. Introduction
    - 11.1. Purpose
    - 11.2. Scope
    - 11.3. Definitions
    - 11.4. References
12. Management
    - 12.1. Organization
    - 12.2. SCM Responsibilities
    - 12.3. Interface Control
    - 12.4. SCM Plan Implementation
    - 12.5. Applicable Policies, Directives, and Procedures
13. SCM Activities
    - 13.1. Configuration Identification
        - 13.1.1. Identify Project Baselines
        - 13.1.2. Delineate Project Titling, Labeling, Numbering
    - 13.2. Configuration Control
        - 13.2.1. Levels of Authority
        - 13.2.2. Processing Changes
        - 13.2.3. The Configuration Control Board
        - 13.2.4. Interface With Other CCB
        - 13.2.5. Support Software
    - 13.3. Configuration Status Accounting
    - 13.4. Audits and Reviews
    - 13.5. Release Process
        - 13.5.1. Version Description Document
14. Tools, Techniques and Methodologies
15. Supplier Control
    - 15.1. Subcontractor Software
    - 15.2. Vendor Software
16. Records Collection and Retention

Appendix 3: Founded Requirements for SCM

1	**Projects existence**	Subversion (SVN)	CVS	Clearcase (CC)	UCM	Serena Dimensions
1.1	New or existing?	2 Because it's easy and very quick to take into use and don't limit use of additional tools. Needs additional change management tool.	1 It's easy to take into use and don't limit use of additonal tools. Needs additional change management tool. Do not have as good set of additional tools than SVN.	1 Not as quick to set up than SVN and CVS and needs scripting when adapted to SW projects. Needs additional change management tool.	1 Not quick to set up, but includes many features to support development	1 Not quick to set up, but includes many features to support development
1.2	Starting development?	2 Easy and quick to set up.	2 Easy and quick to set up.	1 Not quick to set up	2 Not quick to set up, but includes many features to support many development situations	2 Not quick to set up, but includes many features to support many development situations
1.3	Running development down?	2 No licence fee to consider when ramping down.	2 No licence fee to consider when ramping down.	1 Licence period can be an issue when ramping down.	1 Licence period can be an issue when ramping down.	1 Not easy and quick to set up, because Dimension can be configured many ways. Licence can limit ramping up.
2	**Quantitative information**	Subversion (SVN)	CVS	Clearcase (CC)	UCM	Serena Dimensions
2.1	Number of developers	2 No limitations	2 No limitations	2 (More developers means more metadata)	2 No limitations	2 No limitations
2.2	Number of development teams	2 No limitations	1 because many development teams can be parallel to multisite development	2 No limitations	2 No limitations	2 No limitations
2.3	Amount of data (MB/SLOC (source lines of code), src .vs. BIN)	2 No limitations	2 No limitations	2 No limitations	1 No limitations, but UCM generates large amount of metadata. This can decrease UCM's efficiency when size of product gets large.	2 No limitations
2.4	Number & size of components	2 No limitations	2 No limitations	2 No limitations	2 No limitations	2 No limitations

3	**Software building**	Subversion (SVN)	CVS	Clearcase (CC)	UCM	Serena Dimensions
3.1	Development process (Agile/Waterfall)	2 It's possible to be utilized in both models	2 It's possible to be utilized in both models	2 It's possible to be utilized in both models	2 It's possible to be utilized in both models	2 It's possible to be utilized in both models
3.2	Need for multiple SW versions for customer versions (content of SW changes)	1 Merging between different SW versions is difficult, but SVN supports branches	1 There can be, but it is better handled in Clearcase, UCM and Dimensions	2 Multible SW versions are supported	2 Multible SW versions are supported	2 Multible SW versions are supported
3.3	Need for multiple releases under development at the same time	1 Merge is the problem	1 There can be, but it is better handled in Clearcase, UCM and Dimensions	2 There can be multible releases under development	2 There can be multible releases under development	2 There can be multible releases under development
3.4	Need for multi-architecture	1 Can support multible SW configuration. Merge is hard to do.	1 Can support multible SW configuration.	2 Can support multible SW configurations well.	2 Can support multible SW configurations well.	2 Can support multible SW configurations well.
3.5	Platform product	1 SVN do not have real merge operation to enable better control between different release versions	1 Needs additional tools or scripting to support multi-customer development environment	1 It's supported, but it requires great amount of scripting or additional tools	2 Supported	2 Supported
3.6	Using VCS in runtime environment	2 Access to runtime environment can be done with HTTP(S). SVN support most common operating systems.	2 Access to runtime environment can be done with HTTP(S). CVS support most common operating systems	1 Because it doesn't always support the newest versions of Linux Kernell and access to runtime environment can be problem because of firewall settings. However it support most common operating systems.	1 Because it doesn't always support the newest versions of Linux Kernell and access to runtime environment can be problem because of firewall settings. However CC support most common operating systems.	1 Because access to runtime environment can be problem because of firewall settings. Dimensions supports most common operating systems.
3.7	Component based product?	2 Supported	1 Not basic in basic functionality, but can be handled with scripting	2 Supported	2 Supported	2 Supported
3.8	Item types	1 Every item type can be stored, studied VCS are designed for text-files.				
4	**Development language**	Subversion (SVN)	CVS	Clearcase (CC)	UCM	Serena Dimensions
4.1	Are different development languages supported well?	Not an issue at any of the VCSs', because compilers and build environment are dependent on programming languages in this issue.				

5	**Multisite**	Subversion (SVN)	CVS	Clearcase (CC)	UCM	Serena Dimensions
5.1	Some development teams located in different sites? (How well developers can know what others are doing from a tool?)	1 Subversion integrates to ViewVC when it's easier to view changes in repository	1 CVS integrates to ViewVC when it's easier to view changes in repository	1 Multisite is an issue for CC. With CC Multisite or Remote Client multisite development is handled better.	2Development is done in work spaces that include changes made.	2 Modificated items can be locked from server and development is done in work spaces that include changes made.
5.2	Any service located in different sites? (testing, SCM, etc.)	2 With replication solutions, data can be transferred to use of different development functions.	0 No replication by default. With Wandisco possible to do.	2 Development can be distributed around different sites	2 Development can be distributed around different sites	2 Development can be configured around one server or replicated servers
5.3	How replication is supported?	2 Different blug-in tools like Pushmi and Svnsync support different types of replication solutions	0 No replication by default. With Wandisco possible to do.	2 CC Multisite supports replication between different sites	2 Posibility to have one or multiple VOBs	2 Dimensions supports replication between different sites
5.4	Starting up new sites?	2 This is excellent because SVN do not require much computer resources and do no require license	1 This is excellent because SVN do not require much computer resources and do no require license, however SVN offers same and more	1 Every new site need it's own CC licence and configuring needs work	1 Needs great amount of work and requires new licence	2 Needs work, but it don't take long to get it live
6	**Operating system**	Subversion (SVN)	CVS	Clearcase (CC)	UCM	Serena Dimensions
6.1	Can clients be used online and offline?	1 Operation is based on offline development	1 Operation is based on offline development	2 Online clients are not supported on all CC platforms	2 Supports both models	2 Supports both models
6.2	What operating systems are supported in development computers	2 Most important operating systems are supported (Windows, Unix, Linux, Mac)	2 Most important operating systems are supported (Windows, Unix, Linux, Mac)	2 Most important operating systems are supported (Windows, Unix, Linux)	2 Most important operating systems are supported (Windows, Unix, Linux)	2 Most important operating systems are supported (Windows, Unix, Linux)
6.3	Any restrictions of runtime environment?	2 No restrictions as long as environment is supported	2 No restrictions as long as environment is supported	1 Linux Kernell version can be an issue when developing in Linux	2 Firewall setting can be issue because runtime environments can be locaded in different networks that developers	2 Firewall setting can be issue because runtime environments can be locaded in different networks that developers

7	**Resources existing and needed**	Subversion (SVN)	CVS	Clearcase (CC)	UCM	Serena Dimensions
7.1	Recources available for SCM activities	1 Easy to set up and acquire of not available when needed. Do not hold features for different development needs. Learning to use is quick.	1 Easy to set up and acquire of not available when needed. Do not hold features for different development needs. Learning to use is quick.	1 CC environment includes resources to support CC or UCM based development. Big changes are needed if there are need to change VCS. Needs training when using od VCS is starting.	2 UCM environment includes resources to support CC or UCM based development. Big changes are needed if there are need to change VCS. Includes great amount of features to support future develoment. Needs training when using od VCS is starting.	2 ??? Big changes are needed if there are need to change VCS. Includes great amount of features to support future develoment. Needs training when using od VCS is starting.
7.2	Recources required for SCM activities	2 Always available for installing to development environment	2 Always available for installing to development environment	1 Requires acquiring of licence and needs special HW for servers (ie. VOB, Dynamic view server.)	1 Requires acquiring of licence and needs special HW for servers (ie. VOB, Dynamic view server.)	1 Requires acquiring of licence.
7.2	Is there any automized testing tools available or in use?	None of the VCS do not prevent use of other software				
7.3	Available tools nearby (e.g. already clearcase available), (experiences of those tools)	2 Do not prevent use of any tools	2 Do not prevent use of any tools	2 Do not prevent use of any tools, but includes some of the features already	1 Do not prevent use of different tools, but use of other tools is needless because UCM include most of the needed tools integrated to usage model	1 Do not prevent use of different tools, but use of other tools is needless because Dimensions include most of the needed tools integrated to usage model

8	**Collaborators**	Subversion (SVN)	CVS	Clearcase (CC)	UCM	Serena Dimensions
8.1	Functions in collaborator companies (Coding, testing, SCM,..)	2 Easy set up, purchase and connection	2 Easy set up, purchase and connection	1 Needs CC licence. Possible purchase and set up of specific hardware depending location and size of vendor. Needs significant firewall configuration.	1 Needs UCM licence. Possible purchase and set up of specific hardware. Needs significant firewall configuration.	1 Needs licence. Possible purchase and set up of specific hardware. Needs significant firewall configuration.
8.2	Collaboration's permission to see other partners code	2 User access to repositories or single files can be configured	2 User access to repositories or single files can be configured	2 User access to repositories or single files can be configured	2 User access to repositories or single files can be configured	2 User access to repositories or single files can be configured
8.3	Access for collaborators via internet or with VPN (rampup)	2 Access can be realized with HTTPS. Then VPN connection is not needed.	2 Access can be realized with HTTPS. Then VPN connection is not needed.	1 Needs VPN connection between companies development environments or significant firewall configuring.	1 Needs VPN connection between companies development environments or significant firewall configuring.	1 Needs VPN connection between companies development environments or significant firewall configuring.
8.4	Dependency on third party core code.(Big data amount at start)	1 Can be controlled	0 Hard to control	2 Well controlled	2 Well controlled	2 Well controlled
8.5	Dependency on third party core code.(Access to VCS)	2 Most of the needed tools can be founded in open source	0/1 Some needed tools can be founded in open source. No extencive development on top of CVS.	1 Needs CC licence	2 Offers great amount of features for SCM to use	Offers great amount of features for whole development project to use.
9	**Competences (Human perspecive)**	Subversion (SVN)	CVS	Clearcase (CC)	UCM	Serena Dimensions
9.1	Do projects personel need some special competences for using this VCS? (need for training, usability aspect)	2 SVN is simple VCS to learn to use	2 SVN is simple VCS to learn to use	0 It depends of implementation. Base CC is hard to learn, with scripting it can be get easier	1 Although UCM has many features and complex structure, well made GUI makes UCM easier to learn and use than CC	1 It's easy to get in to with short training, but it has so many possibilities to exploit it that it's quite hard to get familiar it by yourself
9.2	Willingness to change current process	2 SVN is easy to use and reliable VCS	0 because it's old version of SVN and do not support development methods as well as other VCSs	1 It's good VCS, but it usually needs lot of learning and it has a licence fee	1 It's good VCS, but it needs lot of learning when utilizing most of its features. Iit has a significant licence fee.	1 It's good VCS, but it needs lot of learning and it has a licence fee

10	**Lifecycle**	Subversion (SVN)	CVS	Clearcase (CC)	UCM	Serena Dimensions
10.1	Flexibility through products life cycle (Transformability)	2 With SVN other tools can be exploited for release handling	1 Because it's quite old it's not so supported as well as SVN	2, but there is some problems like CC don't support nevest Linux Kernel versions	2 Though names are inchangable (Basic principle is that everything putted on VCS is there forever.)	2 Dimenssion is flexible because has many features and many options for its usage
10.2	Maintenance responsibility if shut down (2 vs. 15 years?)	2 No licence period to consider and free to use	2 No licence period to consider and free to use	1 Needs maintenance when OS or tool changes. Has licence fee.	1 Systems needs updating when new operating systems are published. Has licence fee.	1 Systems needs updating when new operating systems are published. Has licence fee.
10.3	Lifecycle for a release (daily building)	2 SVN can utilize different tools for CI development handling	1 Because it's quite old it's not so supported as well as SVN	2 Can be made, but difficult	2 It can be configured to support CI development. Good QUI eases and speeds up usage of VCS	2 It can be configured to support flexible development
10.4	Multiple releases under development at the same time.	2 With SVN other tools can be exploited for release handling	1 Because it's quite old it's not so supported as well as SVN	2 Multible SW releases can be under development at same time	2 Multible SW releases can be under development at same time	2 Multible SW releases can be under development at same time
11	**Authority, Responsibility and Permissions**	Subversion (SVN)	CVS	Clearcase (CC)	UCM	Serena Dimensions
11.1	Ability adapt to different types of authority levels	2 Every VCS can be configured from strict to open type of permissions.	2 Every VCS can be configured from strict to open type of permissions.	2 Every VCS can be configured from strict to open type of permissions.	2 Every VCS can be configured from strict to open type of permissions.	2 Every VCS can be configured from strict to open type of permissions. Update of items in repository can be tracked very carefully.
11.2	How well administrators of SW project can be defined at VCS?	2 Administrators can be well configured in each VCS				
11.3	Can SW component responsibility be shared?	2 With pushmi when network connections are good and stable, 0 with svnsync because it no not support it	2 when network connections are good	2 Basic CC is based on shared responsibility	2 Basic UCM is based on shared responsibility	2 Component responsibility can be well configured
11.4	How well permission to the repository can be configured?	2 Permissions can be well configured in each VCS				
11.5	Can SW component responsibility be divided to different sites or groups?	2 Component responsibility can be well configured.	2 Component responsibility can be well configured	2 Component responsibility can be well configured	2 Component responsibility can be well configured	2 Component responsibility can be well configured
11.6	Can there be separate development projects for each subsystem?	1 It needs to be scripted on top of basic SVN	1 ?	1 ?	2 Can be configured	2 Can be configured

12	**Third party software**	Subversion (SVN)	CVS	Clearcase (CC)	UCM	Serena Dimensions
12.1	Dependency on third party software (tools)	1 Needs additional tools like ViewVC, Pushmi and some kind of change management tool.	0 It don't have as much new tool development that SVN has.	1 Needs additional tools for different features like in change management.	2 No dependency of third party software	2 No dependency of third party software
13	**Open Source**	Subversion (SVN)	CVS	Clearcase (CC)	UCM	Serena Dimensions
13.1	OpenSource	2 Fully open source. Development stategy is strong.	2 Fully open source. No notable development going on.	1 Fully commercial product	2 Fully commercial product, offers great amount of SCM features	2 Fully commercial product, offers great amount of SW development features
14	**Identification scheme**	Subversion (SVN)	CVS	Clearcase (CC)	UCM	Serena Dimensions
14.1	Definitited identification scheme			Not an issue at any of the tools		
14.2	Baselines and duration between baselines			Not an issue at any of the tools		
14.3	Versioning model components			Not an issue at any of the tools		
14.4	Versioning model for releases			Not an issue at any of the tools		
15	**Performance**	Subversion (SVN)	CVS	Clearcase (CC)	UCM	Serena Dimensions
15.1	How Quickly Fault Corrections Needs to be in Software Package?	2 Best CI solution	1 good CI solution	1, CC can be vulnerable for network problems	1, CC can be vulnerable for network problems and need stable network	1 Good response times can be delivered but familiarized implementation did have very basic configuration and had slow response times.
16	**Reability**	Subversion (SVN)	CVS	Clearcase (CC)	UCM	Serena Dimensions
16.1	How easily developer or SCM personel can delete something finally by accident?	2 Selve configured hooks controls rules well, with basic setup everything is removable	2 Selve configured hooks controls rules well, with basic setup everything is removable	2 CC need some configuration from basic configuration. By default baselines can be deleted.	2 UCM is well configured to control user events.	2 It depend on configuration but by default everything can be removed. Only baselines can't be removed by developer. Dimensions new features some times unusable.

17	**Audits and Information Sharing for other Development Functions**	Subversion (SVN)	CVS	Clearcase (CC)	UCM	Serena Dimensions
17.1	Information sharing between different high-level interfaces like R&D management, developers, testing,..	1 Basic reports of repository changes	1 Basic reports of repository changes	2 Reports are good	2 Reports are excellent	1 There are reports that can be utilized
17.2	Reports needed for SCM's use	1 Basic reports of repository changes	1 Basic reports of repository changes	2 Reports are good	2 Reports are excellent	1 There are reports that can be utilized
17.3	Exploitation of history information	1 Basic SVN, 1,5 with ViewVC on top of SVN	1	2	2	It wasn't on real use, but history information is possible to exploit
17.4	Auditing third party software	If third party software is in same VCS that in-house software is, it can be audited with same processes.				
18	**Other features**	Subversion (SVN)	CVS	Clearcase (CC)	UCM	Serena Dimensions
18.1	Backups	Not an issue at any of the tools				
18.2	VCS Maintenance	2 Low effect to R&D	No Experience	2 VOB is locked when in maintenance. It's not a real issue usually, because maintenance is done in quiet time.	2 VOB is locked when in maintenance. It's not a real issue usually, because maintenance is done in quiet time.	No experience