Evaluating Software Configuration Management tools for Opticon Sensors Europe B.V.

Fatma Tosun
25 June 2004
Masters Thesis Software Engineering

Supervisor: Dr. M.G.J. van den Brand
Organisation coach: R. de Vries

UvA Universiteit van Amsterdam

Evaluating Software Configuration Management Tools

For Opticon Sensors Europe B.V.

Fatma Tosun
University of Amsterdam, 2004

MASTER OF SCIENCE IN SOFTWARE ENGINEERING

from the

UNIVERSITY OF AMSTERDAM
June 2004

Author: Fatma Tosun

Supervisor: Dr. M.G.J. van den Brand
Organisation coach: R. de Vries
Organisation: Opticon Sensors Europe B.V.
Opaallaan 35
2132 XV Hoofddorp
The Netherlands
www.opticon.nl

Acknowledgement

First of all, I would like to thank Opticon Sensors Europe B.V. for giving me the opportunity to realize my final assignment. I would like to thank Ron de Vries, for coaching and guiding me through this thesis and special thanks to everyone at Opticon Sensors Europe B.V for answering my questions.

I would like to thank Dr. Mark van den Brand for supervising during the final assignment and for giving feedback on my thesis.

A special thanks to my brother Tuncay and his girlfriend Mirjam for careful spell checking and helping with the layout.

As for my family, I would not have finished my thesis without support from my parents.

Abstract

This thesis is a part of the Software Engineering final assignment of the University of Amsterdam, The Netherlands. It presents results of three months research that I performed at Opticon Sensors Europe B.V. The core intention of the project is to find the best Software Configuration System for Opticon Sensors Europe B.V, hereafter abbreviated as OSE. OSE is a manufacturer of automatic identification products such as code readers, magnet card readers, RFID readers (Radio Frequency IDentification), etc.

This thesis establishes a framework for evaluating Software Configuration Management tools for use in software development projects and evaluates many freeware and commercial Software Configuration Management tools.

Furthermore, it describes the benefits of the Software Configuration Management systems, the differences between several freeware and commercial systems and its advantages or disadvantages. Finally there is a requirements analysis of OSE. This is done to find the best suitable Software Configuration Management system for OSE. The results of the evaluation show which tool with some specific features meets the requirements of OSE.

The last part describes why Subversion is the most suitable Software Configuration Management System for OSE.

Table Of Contents

Acknowledgement 4
Abstract 5
Table Of Contents 7
1. The Project Description 8
1.1 The Occurring Problems 8
2. Software Configuration Management 9
2.1 The Definition of a SCM System 9
2.2 The benefits of a Software Configuration Management System 10
3. Defining The Requirements 12
3.1 Preparing for the Requirements development 12
4. Requirements Elicitation 13
4.1 Functionality Requirements 13
4.2 Usability Requirements 14
4.3 Scalability Requirements 14
4.4 The Scenarios 14
4.5 The SCM Systems 15
5. The Most Relevant SCM Tools 18
5.1 GNU Arch 18
5.2 ClearCase 18
5.3 Concurrent Version System(CVS) 19
5.4 Perforce 19
5.5 Serena ChangeMan DS 19
5.6 Subversion 20
6. Conclusion 21
6.1 Subversion vs. CVS 21
6.2 The Solution 21
7. Project Evaluation 22
Literature 23

1. The Project Description

Opticon originated in Japan at OPTO Electronics co. Ltd (OPTO). OSE is manufacturer of automatic identification products such as code readers, magnet card readers, RFID readers (Radio Frequency IDentification), etc. OSE has a lot of barcode reader types, based on CCD (Charge Coupled Device) or laser technology. The used firmware is based on several sources and is not always the same. This results in the occurrence of implementation differences in the various bar code readers, despite of all tests that are carried out. Furthermore currently it is very cumbersome to work on the same software with several software developers, especially if developers are located in Japan.

The project assignment is to do research about the possibilities of a Software Configuration Management system, hereafter abbreviated as SCM system, to permit access to the central software repository for several international software developers simultaneously. Accordingly, the system has to offer one software pool for all the bar code readers. Such system makes it possible to generate the firmware from this central repository. A proposal for a design about how such a system can look like and which SCM system is most suitable for OSE and OPTO. This thesis will serve as a base for future implementations of a Software Configuration Management system that will ensure a uniform firmware.

1.1 The Occurring Problems

The need for a Software Configuration Management system arose from several problems that originated during software development of barcode readers. The way the history files are organized, makes it difficult to see what has been changed to the code and by whom it has been modified. Sometimes project names are not logical, as a result of which it is not clear what kind of project it is (this occurs mostly with customer-specific versions which have been named differently).

Within the OSE-department, the main intention is to obtain a certain overview concerning the modifications. On distribution area there are different intentions. Currently the communication between software developers of the Netherlands and Japan is not optimal, e.g. changing software between the two companies runs not very smoothly, which means before Japan send a new software release, a newer version is already issued. The most important aspect to change to SCM system is efficiency in software development for both companies.

2. Software Configuration Management

The current definition of Software Configuration Management (SCM) is the control of the evolution of complex systems. More pragmatically, it is the discipline that enables us to keep evolving software product under control, and thus contributes to satisfying quality and delay constraints [1]. A standard definition taken from IEEE standard 729-1983[7] highlights the following operational aspects of SCM:

- **Identification**: an identification scheme reflects the structure of the product, identifies components and their type, making them unique and accessible in some form.
- **Control**: controlling the release of a product and changes to it throughout the lifecycle by having controls in place that ensure consistent software via the creation of a baseline product.
- **Status Accounting**: recording and reporting the status of components and change requests, and gathering vital statistics about components in the product.
- **Audit and review**: validating the completeness of a product and maintaining consistency among the components by ensuring that the product is a well-defined collection of components.

2.1 The Definition of a SCM System

As to what constitutes a SCM system, there is no universally accepted definition. For instance, if a system has version control, is it a SCM system? Ideally speaking, a SCM system is one that provides all functionality based on the definition given above. But practically speaking, any system that provides some form of version control, configuration identification, system structuring, system modelling, and has the intent of providing SCM (to some degree) is considered by the software engineering community to be a SCM system. It should be noted that existing SCM systems provide their own combination of functionality rather than a standard set. This thesis mentions 22 SCM systems, yet there are at least 50 SCM systems that can be acquired for use today, most are commercial products.

Software development efforts are marked by a number of challenges that can easily create chaos, including:

- Large teams
- Time pressure to meet customers requirements
- Changing requirements
- Widely dispersed teams over several locations
- High complexity
- Developers working in parallel
- Multiple versions for different markets or customers

In such a chaotic environment, critical questions such as who made recent changes (and why and with whose approval), when some piece of code broke, what code a specific customer is using, and which components are related, often cannot be answered easily.

SCM attempts to prevent the chaos. SCM is the discipline of controlling changes in a large and complex system throughout its life cycle [6]. SCM can be used to control any type of system development, but this thesis focuses on SCM for software system development. SCM in software development is the disciplined approach of managing the evolution of the software development and maintenance practices. In a formal sense, the objective of SCM is to ensure a systematic and traceable development process, so that at all times a system is in a well-defined state with accurate specifications and verified quality attributes [6]. SCM should validate and maintain the systems' integrity by ensuring that the systems' objects are the appropriate ones.

2.2 The benefits of a Software Configuration Management System

One of the most important reasons to have such a SCM system is the need to have insight in the source code that is being changed in two different countries, namely The Netherlands and Japan. There is a need for such a Software Configuration Management System because of the development of software at distributed places. In such cases there is a lack of insight in the source code that has been changed or managed.

OSE concludes that they require a better SCM solution because of:

- The desire to use a better technology
- Unacceptable pain level with existing solution
- Overwhelming cost of maintaining home-grown system
- A need to meet various internal and external quality standards (such as ISO 9000)
- The consequences of corporate mergers: multiple SCM systems
- The goal to maintain a competitive edge by being more productive and responsive to the marketplace.
- Teamwork optimisation
- Very easy audits
- Insurance against the unknown
- Fewer bugs in released products
- Eliminate avoidable mistakes
- Foundation for quality and process improvement
- Enabling parallel development
- Keeps a specific audit log
- Repeatability of all development steps
- Failure recovery
- All objects are versioned
- Minimal change complexity
- Faster change cycles
- Change propagation

The above listed arguments are some of the advantages of having a SCM system for OSE and OPTO.

A modern SCM solution can provide companies with many benefits such as [2]:

- Productivity enhancement, especially for developers who have their main focus on the creative part of their work (code development) as opposed to chores (setting up their own workspace, tracking down people to find when they should merge code changes, attending meetings, communicating status and filling out forms).
- More opportunity for growth with capabilities not possible before, like the ability to have variant releases for multi-platform products and propagate changes across them and to be able to ship patches to specific customer.
- Visibility into the state of the software development and maintenance activities as well as to all the changes and data items; history of all items such as who made a change, when, how, why an what changed.
- Better forecasting of release dates because the change cycle time becomes measurable and hence, more predictable.
- Change impact analysis to help determine how much effort is involved in a change and hence, how much time it will take.
- Fewer bugs in filed releases; many bugs found in releases (such as having the wrong patch or missing a patch or having not been through testing) could have been avoided if SCM were in place. Some companies assume, that 50% of the bugs found in their filed releases by customers, could have been avoided when proper SCM had been in place.

- More independence for a company. There is less reliance for a company on a single guru, i.e. instead of the SCM knowledge being in one persons' mind, it is disseminated throughout the organisation via the SCM tool; this reduces some risk for the company.
- Enables concurrent engineering, e.g. optimise productivity across a team (developers, testers, build managers, change engineers, release managers). All team members can work simultaneously and do not hold up each others' work. For instance, some developers are fixing bugs while others are creating new code while testers are testing certain bug fixes in test releases while quality assurance staff is doing quality checks on Quality Assurance baselines while build managers are building the latest test baselines.
- Concurrent engineering could significantly reduce cycle time for a change and hence a release.
- Enables distributed development so that teams can work on related changes at different geographical sites.
- Better quality releases: because of the repeatability, there is the guarantee that the necessary steps, such as build and integration testing or Quality Assurance have been carried out on the releases. There are no surprises and less opportunity for the introduction of errors.
- SCM enables a company to leverage its development activities providing strategic corporate benefits. A company can expand its product lines because it now knows how to manage parallel or variant releases of it software and propagate changes properly
- A company with effective SCM can be more responsive to its customers need since change cycle time is more predictable. The company can meet its software release schedules.
- A cheaper way of doing SCM where each step takes less time and man power because of the automation so there is less opportunity to introduce errors. This can provide a project with more slippage time, i.e. time that, in the past, would have been spent on doing horrendous code merges and can now be spend on more productive activities such as adding more enhancement.

3. Defining The Requirements

A set of prioritised requirements is developed based on the feedback from the colleagues, in particular, the eventual users of the tool. The priorities are *must have*, *should have, could have* and *won't have (MoSCoW)*. When developing requirements, it is particularly important to analyse the current way of working to understand what works along with its pains or weaknesses. These pains form the best requirements.

Functionality requirements:
These are the features provided by the SCM tool. Typical functional aspects include: build issues, team engineering, structure and relationship support, versions of all kinds of objects, controlling access, status visibility and reports, audit trails, process mechanisms and workflow management.

When evaluating SCM tools against the requirements, I would not focus just on tool features. An evaluation by creating a table of features and having a column for each SCM tool in the marketplace is not an effective working method to decide which tool is applicable. This is partially useful in culling out the initial set of candidates, but beyond that, it is not an effective way of choosing the final tool. That is because each system uses different terminology, has different architectures and concepts. Much more is needed to properly differentiate.

Usability Requirements:
These concern the SCM tools' easy to use, including: intuitive interface, simple set-up and installation, straightforward migration of legacy code and SCM information, minimal change in usage model, clear understanding of methodologies, minimal learning time and easily customisable.

Scalability Requirements
These describe the ability of the solution to scale to meet the large number of users and large volumes of data, as well as the large number of platforms distributed around the world. It also includes the potential for growth – the SCM solution needs to evolve as the companies' needs evolve. The tools' repository needs to scale up to easily add new users, roles, access rights, and more complex objects.

3.1 Preparing for the Requirements development

At this stage the requirements and wishes of the users, to think of developers, testing and support group, and other interested parties within the organization, will be carefully identified and documented. These requirements and wishes concern both the functionality and quality. With functionality we can think of what the system can offer the future user and with quality we think of aspects like the demanded reliability, speed and accuracy concerning private sensitive matters, etc.

To start, as much information as possible should be gathered in the area explained above. In this case it is the goal to gather as much information as possible about the desired SCM features and concerning the system requirements it selves. In the field of SCM there are many tools available. There are both good Open Source/freeware and commercial SCM tools available. Hereafter the process Requirements Elicitation starts. This implies that you collect information concerning the requirements of two sources: the users and the application field.

After the system requirements have been determined, the requirements of the individual users will be investigated. To manage this, the interview technique will be used. The intention of this technique is to retrieve the wishes of the stakeholders by interviewing them separately.

After collecting all information concerning the system requirements and the requirements and wishes of the stakeholders the applicable tool has to be stipulated.

4. Requirements Elicitation

The next phase is the requirements elicitation phase where I tried to retrieve all requirements. This took place with the journalist technique.

With interview technique everyone is interviewed separately. This will avoid some bias in the group: a central group idea within the group. People have the inclination to coordinate each other or there are always people who have to say more within the group than others. This is the most efficient manner of elicitation in my situation.

After this phase I made a list of all functional requirements. These I subdivided in the MoSCoW method:

- Must have: MUST have this.
- Should have: SHOULD have this if all possible.
- Could have: COULD have this if it does not affect anything else.
- Won't have: WON'T have this time but would like in the future (it concerns points which are at this moment not relevant, but maybe in the future. Then this should be taken into account)

With such subdivision it is simple to compare the functional requirements with the features of the SCM systems.

4.1 Functionality Requirements

These are the requirements that became obvious after the interviews with the software developers of OSE:

Must have:

- Version control
- Version management of the files and folders.
- Compatible with the following attributes:
  - Client platform: Win98 or higher and WinNT4 and Mac OS10
  - Internet browser: Internet Explorer 5 or higher, Mozilla
  - Number of clients: 35 (OSE: 9 employees and 2 to 3 trainees, OPTO: around 20, maybe more)
- Clear repository tree
- Secure server access, because of the distributed teams
- The system keeps up a change log (who has modified what and when)
- Branching feature, important to have because of the multiple versions of different customers.
- Tagging or labelling of a release
- Track of release date has to be simple
- Possibility of parallel programming

Should have:

- Server platform: Linux Red Hat 9, but this is optional
- Graphical User Interface for the clients (command line is not preferred by the most of the employees)
- Diff-feature which can compare more recent version, with a previous one
- Open Source system,

Could have:

- Web based source browsing.
- File scripting, this makes it possible to retrieve the right release files and build it.

Won't have:

- none

4.2 Usability Requirements

These requirements concern the ease of using the tool. To think of the interface, simple set-up and installation, clear understanding methodology, minimum apprenticeship etc.

4.3 Scalability Requirements

In the future the SCM system has to offer one software pool for all different barcode readers. Finally the firmware can be generated from this central repository. This feature is similar to off-the-shelf software, where different types of barcode readers use one source component.

4.4 The Scenarios

Below are four different scenarios of the current repository of OSE and of the future SCM repository. Because of the familiarity of the CVS terms there will be used some of these in the SCM scenario e.g. commit, branch check-in and check-out etc.

The scenarios of the current software development process
Currently there is a directory on the network of OSE named 'Support' with all the source files. The next scenarios describe the different steps that have to be taken to develop or change software and make a new release of it.
Scenario 1 - The most occurring simple scenario of the current system is as follows:

- ➔ The source code, which must be modified, is retrieved from Support and saved on the local system.
- ➔ The source code is modified on the local system
- ➔ The source code is compiled on the local system
- ➔ The source code is build and tested on the local system
- ➔ The modified source code with the associated change-log file is sent to another colleague for the Quality Check
  - ➔ If approved: these will be forwarded to another colleague to update Support with the new release
  - ➔ If rejected: the developer should offer a fixed version to Quality Assurance again.

Scenario 2 -The second scenario describes a situation that occurred a couple of times and caused some trouble:

- ➔ One developer retrieves source code, which must be modified, from Support and saves it on the local system.
- ➔ An other developer, who is not aware of the work of the other developer, retrieves the same source code, which must be modified, from Support and saves it on the local system.
- ➔ Both developers are modifying the source code on the local system, and increasing the software version number. Because the developers are not aware of each others work, the new software version number is the same.
- ➔ Both developers compile the source code on the local system
- ➔ Both developers are going to test their work on the local system
- ➔ The modified source code with the associated change-log file is sent to another colleague for the Quality Assurance
- ➔ It may take some time before QA can test the software.
- ➔ In the mean time one of the versions is sent to the customer because the customer needs the software quickly
- ➔ In this case this specific software version number system gets corrupted. Because software is send to customer, it should be registered on the server.

A Typical SCM User Scenario
Before discussing SCM systems, a simple, typical SCM user scenario is described, in order to present a frame of reference. The structure of the SCM repository remains simply the same as the 'Support' structure.
Scenario 3 - The simple SCM scenario of the future scenario will be as follows:

- ➔ The first thing to do is to get the working copy of the source (project) that has to be modified on the local system. This is possible with the check-out command.
- ➔ The source code is modified on the working copy
- ➔ The source code is compiled on the working copy
- ➔ The source code is build and tested on the working copy
- ➔ The modified source code is ready to commit to the repository
- ➔ The 'commit' command sends all the changes to the repository.
  - ➔ If the source files on the repository has not changed by someone else it will commit the changes
  - ➔ Else it will give a conflict message. Accordingly, the working copy has to be merged with the latest version in the repository.
- ➔ Another colleague will do the Quality Check.
  - ➔ If approved: make a release with the 'tag' command
  - ➔ If rejected: these must be adapted by the developer

Scenario 4 - With the second SCM scenario, *scenario 2* will be prevented:

- ➔ One developer retrieves a working copy of the source code that has to be modified. This is possible with the check-out command.
- ➔ An other developer, who is not aware of the work of the other developer, retrieves the same working copy of the source code, which must be modified.
- ➔ Both developers are modifying the same working copy.
- ➔ Both developers compile the source code on the working copy.
- ➔ Both developers are going to test their work on the working copy.
- ➔ Both modified source codes with the associated change-log file are ready to commit to the repository.
- ➔ The 'commit' command of the first developer sends all the changes to the repository
  - ➔ The changes are committed.
- ➔ The 'commit' command of the second developer sends all the changes to the repository.
  - ➔ The SCM system gives a conflict message. The developer could simply check the log message to see who made the previous modifications or merge his version with the previous version.
- ➔ Another colleague will do the Quality Check of the latest released version.
  - ➔ If approved: make a release with the 'tag' command
  - ➔ If rejected: these must be adapted by the developer

4.5 The SCM Systems

After retrieving the necessary requirements the search for the most applicable SCM starts. This is managed by searching for all open source and commercial SCM systems. To look for the most appropriate SCM system, some important features are taken into account:

- Merge function
- Branch function
- Tag or labelling function
- Platform type
- GUI client
- Web based view
- Change log file
- Documentation

A list of all SCM systems are divided over table 1 and table 2. The first table is a list of all open source systems and in the second table the commercial SCM systems are evaluated.

		Platform	GUI client	Web Based	M(erge), B(ranch), T(ag), L(abel)	Change Log	Documentation: E(xellent), G(ood), M(edium), P(oor)	Extra features	Comment	License(for 35 clients)
Open Source	**CVS**	Win/ Unix/ Linux/ Mac OS	Many GUIs	Yes	M/B/T	Yes	E	Many GUIs and make/build tools available.	It has a to complex tag/branch feature. Does not support directory-versioning and file renaming	GNU GPL
	Vesta	Linux/Unix	No	Yes	M/B/	Yes	M	Integrated build, renaming	Does not support Windows platform. Last release 22/7/'03	GNU LGPL
	SubVersion(SVN)	Win/ Unix/ Linux/ Mac OS	Many GUIs	Yes	M/B/T	Yes	E	Directory versioning, file/dir renaming, It checks if the committed file has a conflict.		Apache/BSD
	OpenCM	Unix	No	No	B		G	Renaming files,	Only Linux	GNU GPL
	Aegis	Unix	Yes	Yes	M/B	Yes	M	Renaming files and directories	Only Unix and each project has his own repository	GNU GPL
	GNU Arch	Unix???	Yes	Yes	M/B/T	Yes	M	Renaming files and directories		GNU GPL
	Monotone	Unix	No	No			G	Renaming is possible		GNU GPL
	Svk	Linux/Unix/ Win/ Mac OS	No	Yes	M/B/T	Yes	P	Renaming is possible	It uses SVN file system	Perl license

Table 1- Open source SCM systems

		Platform	GUI client	Web Based	M(erge), B(ranch), T(ag), L(abel)	Change Log	Documentation: E(xellent), G(ood), M(edium), P(oor)	Extra features	Comment	License(for 35 clients)
Commercial	**Spectrum SCM**	Linux/Unix/ Win/ Mac OS	Yes	Yes	M/B/T	Yes	G	Issue tracking, reporting, Java based SCM.		$600 * 35 + $1500 = $22.500
	Perforce SCM	Win/ Linux/ MacOS Server:Win/Unix	Yes	Yes	M/B/L	Yes	E	Reporting, defect tracking, labelling(to keep track of old releases), build tool	It has all the necessary features, but is to expensive	20 * $750 + 15 * $700 = $25.500
	Rational ClearCase	Win2000 +/ Linux/Solaris	Yes	Yes	M/B/T	Yes	?	Build management, reporting, stream tree browser,	It does not support Win98!	$1.720 * 35 = $60.200 excl. tax
	Surround SCM	Win/Unix/ Linux/ MacOS	Yes	Yes	M/B	Yes	?	Email notification,		$595 per license
	QVCS Enterprise	Win/Linux	Yes	Yes	B/L	Yes	M	File locking, Reporting, diff tool,	It has no more features than an OS system	$99/ 4 user = $891
	AccuRev	Win/ Linux/ Unix	Yes	No	M/B	Yes	E	Diff tool, stream browser,	To expensive	$1399/user = $48.965
	BitKeeper	Linux/Unix/ Win	Yes	Yes	M/B/T	Yes	E	File renaming, Renaming files and directories	Very expensive	$2800 to $5800/user
	Serena ChangeMan	Win/Unix/Linux	Yes	Yes	M/B/	Yes	?			From $50.000
	Synergy CM	Unix/Win NT server	Yes	Yes	M/B	Yes	?	Conflict detection, build management, renaming	Very expensive	$25.000 for 10 users
	Borland StarTeam	Win NT4 Win2000+							To expensive, does not support Win98	$1299/ user = $45.465
	Razor van visible.com	Win(no win 98), Linux, Unix	Yes	Yes	M	Yes	G	Issue tracking, build management, reporting,	To expensive, does not support Win98	± $26.500
	Sablime van bell-labs.com	Unix/ Linux/ Win	Yes	Yes	M/B/T	Yes	G	Email notification, detects dependencies,	No information about licence price.	
	Bitsafe van bitmanager.de	Unix/Linux/ Win2000+	Yes	?	M/	Yes	?	Diff.	Is to standard for a commercial tool Does not support Win98	€199 * 5 + €149 * 30 = €5465
	Team Coherence	Win/Linux(only command line)	Yes	No	M/B/L	?	G	Build management, Bug tracking		$599 * 5 + $479 * 30 = $17.365

Table 2 - Commercial SCM systems

5. The Most Relevant SCM Tools

In this section an alphabetical enumeration of a number of SCM tools follows, which has been evaluated. I have not described the most SCM systems mentioned in table 1 and table 2, because they do not meet the most requirements of OSE. A selection of six SCM systems are described in this chapter, these systems constitute most of the required features. Concluding, the next chapter describes why Subversion has been chosen as the most appropriate SCM system.

5.1 GNU Arch

Arch is an advanced SCM specification with multiple, independent but compatible implementations. Arch is widely considered as ambitious, mainly on account of its support for fully decentralized (as opposed to merely distributed) repositories. In Arch, any branch or developers' private work area can be treated as a repository of its own, with a global name space for developers, repositories, and branches.

Some advantages of GNU Arch are:

- Works on Whole Trees. Arch keeps track of whole trees – not just individual files. For example, if you change many files in a tree, Arch can record all of those changes as a group rather than file-by-file; if you rename files or reorganize a tree, Arch can record those tree arrangements along with your changes to file contents.
- Change set Oriented. Arch does not simply "snapshot" your project trees. Instead, Arch associates each revision with a particular change set*:* a description of exactly what has changed. Arch provides change set oriented commands to help you review change sets, merge trees by applying change sets, examine the history of a tree by asking what change sets have been applied to it, and so forth.
- Fully Distributed. Arch does not rely on a central repository. For example, there is no need to give write access to a projects' archive to all significant contributors, instead, each contributor can have their own archive for their work. Arch seamlessly operates across archive boundaries.

A very important disadvantage:
A weakness of Arch: it does not work well on Windows-based systems. The smart functions of the system are in the client tools, not in the server. But the GUIs for Arch is still young.

5.2 ClearCase

ClearCase is the market leader and provides change management functionality in addition to the standard version control. ClearCase is part of a suite of products from Rational that implement Rationals' "best practices" software development methodology, Unified Change Management.

Advantages

- Build management and auditing
- Very simple to retrieve old versions
- Labelling of product releases

Disadvantage

- A very important disadvantage of ClearCase is the price, it is too expensive
- It does not support Windows 98; there are still a few computers with Windows 98 platform.

5.3 Concurrent Version System(CVS)

CVS, a free, open source tool, is perhaps the most popular Open Source SCM tool. Even though it has some serious shortcomings, CVS is used for dozens of open source projects, including Apache WWW server, FreeBSD, NetBSD, OpenBSD, GNOME, and Xemacs. CVS is one of the first Open Source version control projects, which is used by many people, but it has its shortcomings.

advantages of CVS

- Legacy systems may use CVS
- Many clients have built-in-support for CVS

disadvantages of CVS

- It is not possible to rename or move a file or directory in the repository.
- Moving files without losing data cannot be done without the sysadmin
- File metadata is not versioned
- Need for tags, etc. because of cost of branching large trees
- Diffs only transferred from the server to the client
- Limited support for binary files
- pserver sends clear text passwords

5.4 Perforce

Perforce is a popular tool in the academic community, perhaps because the company provides the tool for free to open source efforts Perforce is known for its simple architecture and unique branching model that promotes outwards instead of inwards towards the trees' trunk. Perforce is a SCM system with emphasis on high performance, using RCS files plus a database. Supports versioning of most objects, change control, shared access, atomic commits, branching/merging, and auditing for software production teams.

Advantage

- Labelling feature
- Built-in defect tracking
- Reporting system that can be accessed by popular reporting tools

Disadvantage

- Very expensive tool
- No directory versioning

5.5 Serena ChangeMan DS

Serena ChangeMan DS is a solution for managing distributed software application from development to development. It supports the most common operating systems using a client-server and agent based architecture. Serena ChangeMan DS is a networked SCM integrated with build control, release management, a programmers' editor, software-distribution tools, and process control.

Advantage

- Sophisticated version control with cross-platform build and release management
- Processes and approvals configurable through GUI
- Direct, seamless, feature-rich IDE integration
- Advanced architecture for true cross-platform support and central point-of-control
- Improved developer productivity

- More collaborative work environment
- Lower development and administrative costs
- Improved application uptime and reliability
- Auditable and traceable activity
- Consistent, enforcement of business processes across environments

Disadvantage

- One of the most expensive systems

5.6 Subversion

Subversion is a freeware/Open Source SCM system. Subversion supports atomic commit, that means that if an operation on the repository is interrupted in the middle, the repository will not be left in an inconsistent state, it only takes effect after the entire commit has succeeded. The revision numbers are per-commit, not per-file like in CVS. Another very important feature is the possibility to move or rename a file or directory without losing the history log of that file or directory. In comparing to CVS, Subversion has a very cheap branch/ tag option: it copies files or directories to a different location at the repository level(with retaining the history). These copies are actually no duplicates but a new directory entry that point to an existing tree, so there is no need to worry about the growing repository data. With Subversion it is possible to define permissions to access to different parts of a remote repository. The WebDAV-based service supports defining HTTP permissions for various directories of the repository.

advantages of subversion

- Files and directories can be renamed or moved easily
- Copies and branches are a constant time operation
- Only diffs in both directions over network
- Revision numbers apply to the entire repository
- Revision numbers apply to the entire repository
- Integration into Apache allows the creation of complex access control models
- Commits are repository-atomic.
- Growing client and application plugin support

disadvantages of subversion

- Old projects and existing build systems need conversion
- Not as wide-spread as CVS yet

6. Conclusion

In this section only Subversion and CVS will be discussed. The open source system was the preference of OSE and it satisfies the requested requirements. That is why there is no need to discuss the commercial products in the following part of the thesis. Nonetheless a comparison of open source and commercial products was necessary to stipulate what appreciation the commercial product would have.

6.1 Subversion vs. CVS

"The goal of the Subversion project is to build a version control system that is a compelling replacement for CVS in the open source community." This is the well-known motto of Subversion. But is it already a replacement for CVS?

CVS has evolved into the standard SCM system of the Open Source community. But CVS is aged, and it has many deficiencies. For example, it is not possible to move or rename files, it tracks only file contents, not tree structure and the branch feature is very complicated. Furthermore, CVS has a poor network support: it has annoying waiting times while committing, because file differences are sent in only one direction, from server to client. To run it in a safe way, you must use it in conjunction with ssh, and this means that you must give all participants a system account. The commit operation of CVS is not atomic: if the network connection breaks in the middle of a commit operation, this may damage your data. Another important reason for not choosing CVS, is the lack of directory versioning. In this case it is not possible to compare different versions of folders with each other. The current repository of OSE has been subdivided in release folders, thus directory versioning is one of the most necessary requirements within the SCM system.

Subversion seems to implement all features that are missing in CVS. The version control model is more complete, as you can check in directories. The network support is much better, as it uses the WebDAV protocol. One can tunnel it easily through SSL, and the only requirement is to have a web server user account to get write access. The binary diff-ing algorithm is used to store and transmit files in both directions. The tag and branch commands in Subversion are simple copies of a directory. But there is no need to worry about the growing repository data. These copies are actually no duplicates but a new directory entry that point to an existing tree.

6.2 The Solution

It is obvious that I would advise to choose for the Subversion system. A serious shortcoming of CVS is the impossibility of versioning directories, renaming and moving of files and directories. To complete the system I recommend the following tools:

- With using TortoiseSVN[t4] for Windows you will have a client for Subversion that integrates into Explorer. Your files become colour coded to indicate their status, and most Subversion operations can be invoked using a simple right-click. TortoiseSVN offers a graphical diff and merge tool to help you track changes.
- RapidSVN [t5] is a front-end GUI for Subversion. It runs on any platform on which Subversion can run. It is a tool that provides an easy to use interface for Subversion features.
- ViewCVS [t1] allows you to view a CVS repository in a web browser, but it also supports Subversion repositories. If you are using Apache with Subversion, you can browse your repository straight away using a web browser.

7. Project Evaluation

The core intention of this thesis is to present a proposal for a design about how such SCM system can look like and which SCM system is most suitable for OSE and OPTO.

I am very glad the recommended solution is a open source software. It was important for OSE to find a low budged solution. As you can see in table 2, the most commercial products are very expensive. Fortunately, the open source system Subversion satisfies the OSE requirements.

If I look back to my training period, I would approach some procedures differently, i.e.:

- I would not waste too much time with searching for commercial products, but go more into detail of the possibilities of open source systems.
- I started late with my thesis, consequently I ran out of time. A better planning had prevented this problem.

The **self assessment of this project** is (judging on a scale of 1 to 10):
Quality of the research result: 8
Quality of the thesis: 8
Difficulty of the final assignment: 7,5
The relevance of the Master Software Engineering courses: 7

Literature

[1] Jacky Estublier, *Software Configuration Management: A Roadmap.* Dassault Systemes LSR, Grenoble University, France.
[2] Susan Dart, *Best practice for a Configuration Management system*, System Configuration Management, ICSE'96 SCM-6 Workshop Berlin, Germany.
[3] Ben Collins-Sussman, Brian W. Fitzpatrick, C. Michael Pilato. *Version Control with Subversion*.
[4] Susan Dart, *Concepts in Configuration Management Systems*, Software Engineering Institute
[5] Better SCM Initiative Comparison: http://better-scm.berlios.de/comparison/comparison.html
[6] Jacky Estublier and others, *Impact of the Research Community on the Field of Software Configuration Management: Summary of an Impact Project Report*, ACM SIGSOFT Software Engineering Notes, v. 27, issue 5, p.31-39, September 2002.
[7] *IEEE Guide to Software Configuration Management.*1987. IEEE/ANSI Standard 1042-1987.
[8] C. Walrad, D. Strom. *The Importance of Branching Models in SCM.* Computing Practicec, p31-38, September 2002

Web links:

[s1] Arch – gnuarch.org
[s2] Surround SCM – www.seapine.com/surroundscm.html
[s3] GNU Revision Control System(GNU RCS)
[s4] Subversion – subversion.tigris.org
[s5] CVS – www.cvshome.org
[s6] OpenCm – www.opencm.org/
[s7]Vesta – www.verstasys.org
[s8] Aegis – aegis.sourceforge.net
[s9] Perforce – www.perforce.com/perforce/technical.html
[s10] Sablime – www.lucent.com/livelink/0900940380006366_Brochure_datasheet.pdf, www.bell-labs.com/project/sablime/
[s11] Serena ChangeMan DS - www.serena.com/product/cm_ds_ov.html
[s12] QCVS - www.qumasoft.com/
[s13] Team Coherence – www.qsc.co.uk/doc_tc.htm

SCM tools :

[t1] ViewCVS – viewcvs.sourceforge.net/
[t2] CVSGraph – www.akhphd.au.dk/~bertho/cvsgraph/
[t3] CVSGrab – cvsgrab.sourceforge.net/
[t4] TortoiseSVN – tortoisesvn.tigris.org
[t5] RapidSVN – rapidsvn.tigris.org