ISSN 0280-5316
ISRN LUTFD2/TFRT--5835--SE

Configuration Management for Industrial Automation

Henrik Israelsson
Leonardo Bello

Department of Automatic Control
Lund University
May 2009

Lund University
Department of Automatic Control
Box 118
SE-221 00 Lund Sweden

Document name
MASTER THESIS

Date of issue
May 2009

Document Number
ISRN LUTFD2/TFRT--5835-SE

Author(s)
Henrik Israelsson and Leonardo Bello

Supervisor
Magnus Wendt at Tetra Pak Processing Systems, Lund
Charlotta Johnsson Automatic Control, Lund
Karl-Erik Årzén Automatic Control, Lund (Examiner)

Sponsoring organization

Title and subtitle
Configuration Management for Industrial Automation (Konfigurationshantering för industriell automation)

Abstract
Core Issue: Developing and maintaining industrial automation systems requires a lot of coordination of programming code. To support this task a configuration management system can be used but many companies of today find it hard to select a system that suites them best.

Purpose: The purpose of this thesis is to examine various configuration management systems and evaluate them according to the needs that Tetra Pak Processing Systems – Business Unit Dairy, Beverage and Prepared Food (BU DBF) has. The Platform & Standards department, where the thesis has been written, has started to work with templates that are shared between different projects. This leads to more versions which make it harder to organize and structure the work.

Methodology: Qualitative information has been gathered through interviews and quantitative information has been gathered through a survey. Using configuration management requires the user to understand the theory behind it and the area has been studied thoroughly. Various tests on different tools were performed according to test specifications.

Conclusions: For BU DBF the largest problem was to find a configuration management tool that supported binary files, which are used for PLC and HMI, from several manufacturers. Both freeware applications and commercial tools were tested. The tool that best fulfilled the test specifications was VersionWorks from GepaSoft/Rockwell Automation. Implementing configuration management within BU DBF is something that definitely should be done and based on the results of this thesis.

Keywords

Classification system and/or index terms (if any)

Supplementary bibliographical information

ISSN and key title
0280-5316

ISBN

Language
English

Number of pages
141

Recipient's notes

Security classification

http://www.control.lth.se/publications/

Acknowledgements

The Master Thesis has been written within the Platforms & Standards department of Tetra Pak Processing Systems. More specific at BU DBF (Business Unit Dairy, Beverage & Prepared food).

Even though the subject at hand was completely unfamiliar to us when starting our work, we believe that the result of our research can be a step in the right direction for BU DBF when approaching the world of configuration management.

The assignment could not have been carried out without the help from a variety of people and we would like thank them for their cooperation.

Magnus Wendt, our supervisor. Our numerous meetings have pointed us in the right direction and given us a deeper insight into the company.

Tommy Carlsson, for providing us with answers to all the technical questions that have emerged and for the help with acquiring some necessary software applications.

Udo Krüger, for helping us keeping contact with GepaSoft and showing us the basic use of Siemens Simatic S7.

Jessica Gräntz, for helping us with administrative questions and for sharing your good mood.

Erika Hodge, Wictoria Johansson and ***Helena Persson*** for many good laughs in the office. We bet your future work colleagues will enjoy your company as much as we did.

Henrik Jönsson, for giving us the opportunity to write our master thesis at Tetra Pak.

All other employees that were kind enough to participate in the survey and interviews.

Charlotta Johnsson, our supervisor at LTH, for making us think outside the box and for giving us good ideas about how to approach and structure our work

Karl-Erik Årzen, our examinator at LTH, for providing us with literature about Configuration Management.

We hope we did not forget anyone and in case we did, we apologize and send our thanks to you too.

Table of contents

1 Introduction

The purpose of the master thesis is to find a suitable configuration management tool for handling the development of industrial automation systems. Tetra Pak Processing Systems, Business Unit Dairy, Beverage and Prepared Food (BU DBF) develops a variety of dairy, beverage and food machines. The machines are equipped with industrial automation systems such as PLCs and HMIs. To be able to organize and structure various PLC and HMI development projects within Tetra Pak Processing Systems BU DBF in a better way, different configuration management tools should be evaluated. The advantages with such a system are numerous. Keeping track of all versions and variants of code can be a hard task if configuration management is not used. By keeping track of all versions in a structured way a company that works with industrial automation can gain many advantages towards its competitors. By using configuration management many common problems encountered when working with software development can be resolved faster and more accurate. This can strengthen the reputation of a company on the market since customer demands can be met in a more efficient way. It will also facilitate the day-to-day work for the developers

1.1 Background

Configuration management (CM) is a very general way of describing a work method. It was originally created by the US military in the 1950s as a technical management discipline [26]. The CM method can be applied to a large variety of applications and it is often used for software development.

One definition of CM is: Configuration management is the art of identifying, organizing and controlling modifications to the software being built by a programming team.[1]

Most CM standards and books define CM as follows[2]:

1. Identification
2. Control
3. Status accounting
4. Audit and review

Identification means that you uniquely identify every version of an item that makes up the software.

Control means that you control and track all changes to any versions and configurations throughout their life cycles.

[1] Babich, Wayne A., Software Configuration Management – Coordination For Team Productivity, page 8
[2] Dart S., Configuration Management: The Missing Link in Web Engineering, page 74

Status accounting refers to reporting and recording the status of all objects. This includes files, change requests and configurations.

Audit and review is the process of keeping an audit trail of all actions, events and notifications that happen to all the objects under CM control. It also means that you ensure that a CI, configuration item, is a valid, consistent set of components and that CM activities are being carried out correctly.

Today most industrial manufacturing is automated to be able to produce larger quantities – faster and cheaper – and with higher precision. Developing new and good industrial automation systems is an important task. These projects typically involve many people and a lot of data. It is therefore fundamental to have a support system that can help keeping track of the different versions of the software as the project moves forward.

When developing new industrial automation systems, the data can consist of everything from pure code and graphical programs to documentation. A configuration management system would ease the process of finding the right version of a program when, for example, a bug is discovered or when an upgrade should be performed.

Most software companies use some sort of CM tool and more companies are starting to realize the benefits of a functioning CM system. The savings, both in money and in working hours that a good system can provide is substantial. For a software company to be able to thrive and be prosperous a good CM standard is needed. [1]

BU DBF develops a variety of dairy, beverage and food machines. The machines are automated and controlled by PLCs from Rockwell and Siemens. The machines also have an HMI from Beijer Electronics and if the customer wants to be able to control the whole plant from a single computer, an HMI called Tetra PlantMaster that are developed in Wonderware Intouch can be bought. A machine is built from a Main Platform Requirement Specification, various blueprints and customer demands. This produces a lot of different file types and versions that are constantly being updated and revised. This creates a need for a system that can help organize and structure the developers work.

Every machine consists of both hardware and software and is delivered with quality guarantee. This puts high demands on the product and requires it to work properly. In case it does not, the procedure of finding and correcting the problem needs to be fast and efficient. This is particularly important since the machines produce food and beverage which means that a defect in the manufacturing, hardware or software, can lead to illness or even death.

Earlier BU DBF has used libraries of blocks when programming PLC and HMI. This means that every machine has been unique and all version history and documentation has been tied to a single machine. For the last two years another approach has been used. Instead of libraries, platform templates are being used. This transition has been made mainly to decrease the number of different components used in the manufacturing, to increase quality, decrease maintenance costs and to be able to share code between different branded processing units - machines. For example, an alarm routine or a sensor of some sort can be used on many machines and tied to this component there is a software template. Since a platform template can be shared between different machines it is important to be able to keep track of the machines that share the same code in order to be able to perform future updates and bug-fixes. It is also important to easily be able to find out which software templates are compatible with each other. A CM tool could facilitate all these tasks, BU DBF does not yet have such a system.

This master thesis includes a case study performed at Tetra Pak with the aim of proposing a suitable configuration management system for the department Processing Systems Business Unit Dairy, Beverage and Prepared Food (BU DBF).

The aim of this master thesis is to present CM, investigate the specific needs that BU DBF has on a CM system, compare various CM tools available on the market, and ultimately to suggest the most suitable CM tool for Tetra Pak Processing System BU DBF.

1.2 Demarcation

The depth of the thesis will be limited to what you can perform in 20 weeks.

The first thing you realize when starting to look into the field of Configuration Management is the enormous depth of the area and the huge amount of available CM tools. This may sound contradictive but having many available applications makes the process of choosing which ones to evaluate more difficult.

The work will be limited to automation within the BU DBF department. The cooperation between the processing side and the packaging side of the company has so far been close to non-existing and suggesting a way to implement a CM model that goes beyond the platforms and standards department would require a lot more than 20 weeks of work.

Finding a tool has been a part of the work and due to the fact that licenses for software that specializes on handling binary files are both rare and extremely expensive, not all the tools on the market that we would have liked to test have been possible to investigate.

It turned out to be really hard to get a hold of good literature about CM. Most facts can be found on the Internet and Lars Bendix at LTH handed out a few extracts from various books. Most CM books available at the library are extremely monotonous and not up to date. This is most likely due to the fast development of the software industry and releasing a book that will be outdated in a few years is not profitable. Internet seems to be the best source of information for this subject and even the authors of the books we have used refer to Internet in their references.

A lot of different wills have made themselves heard during the work. It would have been nice to please everyone but going into too small details in the CM model during the work here has been impossible, mostly due to lack of time. We have tried to focus on the big picture and small technical details regarding what the tested tools can do has had to be overlooked. We have tried to focus on the basic and most used features in the various tools. Overall user friendliness of the applications has also been prioritized.

CM is a wide topic and version control is only one part of it. Since there is a wish for a version control system, focus has been put on that part of the CM model. Another strong wish has been the ability to make graphical compares between different revisions and this particular feature has therefore been investigated thoroughly.

We did not get access to all the software we needed when starting the project and has been forced to chase it on our own. By providing us with the necessary development tools from the beginning, much time could have been saved.

1.3 Abbreviations and Definitions

Branch

Branching allows parallel versions, branched of from the main version to be developed. Useful when it is uncertain how to continue the development. It can later be decided which branch to use. In case many branches contain useful changes they can be merged into the main branch.

Checkout/Pull

Fetching files from the repositories and placing them in folder on the local machine to be able to modify them.

CM

Configuration management.

Commit/Push

Uploading changed files to the repository.

Conflict

Occurs if two users modify the same file at the same time. When one of the files are committed the other user will get a conflict when doing an update. This is resolved by viewing the changes with a diff-tool to see if it is safe to merge the files.

CVS

Concurrent Versions System.

GPL

The GNU General Public License. A widely used free software license.

GUI

Graphical User Interface.

HMI

Human Machine Interface.

IDE

Integrated Development Environment. An IDE normally consists of a source code editor, a compiler and/or interpreter, build automation tools and a debugger.

Merge

When working with branches or when a conflict has occurred, merging can be used to interweave changes between different revisions.

PLC

Programmable Logic Controller.

Repository

Directory containing files and all revisions of them. Can be placed on a centralized server, distributed between users or on the local machine. The repository is often referred to as the repo.

Revision

When modifying the contents of a file and saving it, the new version of the file is often referred to as a new revision.

Subversion

Most used VCS today.

Tag/Label

Used to give a revision a release number or name that does not have to be included in the name of the file.

Update

Fetching the latest version of the project from the repository. Needed to be done before committing files to repository to avoid conflicts and to be able to carry out tests with the latest versions of subroutines.

VCS

Version Control System.

Wizard

Application or script that helps the user through an installation, a setup or configuration of an application.

1.4 Guide to this report

The thesis has been divided into the following chapters:

1. **Introduction –** In this chapter we describe why the master thesis is important and how Tetra Pak Processing Systems, BU DBF can benefit from CM.

2. **Methodology –** This chapter describes which methods we used to solve the assignment and why we chose them.

3. **Theory: Configuration Management** – This chapter will hopefully cover the basic knowledge the reader will need to posses about Configuration Management in order to appreciate the contents of this thesis. The history of Tetra Pak and the structure of the company are also briefly explained.

4. **Tetra Pak –** The history of Tetra Pak and the structure of the company are explained in this chapter. The roles of the employees that have been involved in the Master Thesis are also explained.

5. **Empirical Foundation –** This chapter describes the way in which we collected data to solve the assignment.

6. **Configuration Management- and Software Development Tools –** In this chapter we give an introduction to the various CM tools that have been tested and the software development tools that they should support.

7. **Results and Discussion –** In this chapter we present the results from our case study along with a discussion about the them.

8. **Conclusions –** Based on all the impressions from our work and from the discussion and results in chapter 7, this chapter presents the conclusion we have reached.

9. **References –** This chapter lists the references for this thesis.

10. **Appendix 1: Basic usage of the tested CM tools –** Since it might be a bit dull to read a paper about software applications without knowing what they look like, this chapter is constructed to give a view of the basic look and usage of the tested CM tools.

11. **Appendix 2: Survey –** This chapter contains the survey we handed out.

2 Methodology

All CM tools have the same reason for existing, which is facilitating and automate certain aspects of the developers' work. There are also specific functions within the tools, for instance graphical compare and traceability of files. In our search for a suitable application, we needed to know more about what the typical developers' day-to-day work looks like and which particular features they wanted to see in the tool.

2.1 Mode of operation

The main task of the master thesis is to find a suitable CM tool for BU DBF's projects. In order to do so, you need to understand the basic principles about CM. With this knowledge at hand, you can fully appreciate the tool and make the best use of its' advantages. Our first task is therefore to read and gather information about CM and CM tools. The importance of having a good understanding of what CM is cannot be pointed out enough since the implementation of CM in a company will be more likely to end in failure if the people using it do not know how to use it properly and especially if they do not know why they use it [3].

It is also essential to understand the daily work at BU DBF. In order to illustrate how the work is performed, flowcharts can be used. We have therefore constructed two flowcharts. The first one is a model of "who contacts who" when it comes to changes in case of bugs or upgrades. The second flowchart depicts the ownership to different software platforms and templates. The developers at BU DBF were given several opportunities to comment and update these flowcharts in order to assure that they are correct.

By performing interviews and surveys with the developers we got a good understanding of the work procedures and the general opinions on configuration management.

We decided that testing different configuration management tools with the file types used within BU DBF would be the best approach to find a tool. This was done according to test specifications we had set up, based on the theory we had acquired, but also with the wishes of features from developers that participated in the survey and the interviews in mind. The investigation of a suitable CM tool would include testing available programs to the extent that it was possible. When it comes to open source CM tools this would include setting up a server, in our case a virtual machine for convenience, and testing certain specified criteria. The applications that could not be tested for free would hopefully be available in trial versions. In the cases they were not, the company providing the program would be contacted to work something out.

We chose to test the tools ourselves since it would be hard to do field studies at other companies due to the fact that a CM system varies between different companies. Conducting interviews with employees at other companies that use CM would not answer all our questions either since the company would have to use exactly the same file types as BU DBF to be able to give reliable answers. We did however contact several other companies and asked which CM tools they are using and how well they are working. This helped us in the process of deciding which tools to evaluate.

2.2 Gathering data

First of all as we wanted to learn more about what people actually knew about CM and to find out what the developers day-to-day work looks like we decided that handing out a survey was the best approach. Conducting interviews were another option but considering that the survey was handed out to over 40 employees; this would not be possible due to the time limit. The survey was a quantitative study with predefined answering alternatives and since the survey was constructed to give us an overview about the developers' knowledge about CM, we reasoned that a quantitative investigation would be enough. Closer details about the survey are found in *Chapter 5.1 Survey* and the survey itself can be found in *4.2.1 Flowcharts*.

We still felt that interviews could provide us with more qualitative information and decided to sit down in private with eight specially selected employees within the department. Closer details about the interviews are found in *Chapter 5.2 Interviews*.

The last step would be to find a suitable CM tool. This would require a lot of testing with the wishes of features from the survey and the interviews in mind, and our own test specifications. More details about the testing are found in *Chapter 5.3 Finding a CM tool*.

3 Theory: Configuration Management

A common definition of CM consists of four aspects; Identification, Control. Status Accounting and Audit and review (see also *Chapter 1.1 Background*). This is a very general way of describing CM and a description that was created before there were any CM tools facilitating a development process. When it comes to software development there are many more aspects that needs to be considered. Another definition of CM is presented in *Table 1*[1]. This definition consists of eight steps.

Most of the aspects in *Table 1* are more or less built into the different CM tools that exist today. Depending on the areas that the company wants to focus on different strategies can be applied and also different tools can be used.

It is important to know that a CM strategy can vary a lot between different companies. The tools used will of course be different depending on the files they need to support and what kind of products the company produces. However, the most important thing when approaching CM in any company is to have a well defined definition of CM and a well thought through strategy. This is essential to avoid confusion about what CM will accomplish and to set expectations at the right level. If this is not done right the situation will be that different people and departments will have different definitions of what CM implies and thus also how to use it. The worst case scenario is that someone does not follow the CM strategy that is set up, thinking that CM does not concern them since their definition of CM differs, and this will inevitability effect the rest of the co-workers. [1]

Functional Area	Operational Aspects
Version and configuration control	• Version identification • Configuration items • Baselines, snapshot, releases and variants • Types of components • History records
Configuration item structuring	• System architecture • Relationship and traceability • Version selection • Consistency management • Project context • Repository management (deltas, administration, metadata)
Construction of configurations	• Build management • Build optimization (partitioning, interfaces) • Repeatability/regenerations of baselines

[1] Dart S., Configuration Management: The Missing Link in Web Engineering, page 95

Change management	• Change impact analysis • Change requests and classification of change • Change tracking and escalation • Change sets • Change propagation • Release planning
Teamwork support	• Workspace management • Parallel development (merging, conflict resolution) • Variant management • Distributed and remote development
Process management	• Notifications • Event triggers • Workflow and release management support • Role support and access control • Task and project management
Auditing	• Logging • Validation of integrity of CM activities
Status reporting	• Queries on status of CM activities and artefacts • Report generation • Metric gathering

Table 1: Functional areas and operational aspects of CM.

3.1 Why use CM?

The fundamental question that a CM strategy answers is the question of “what program/code is this?” Having a well-planned strategy diminishes the developers’ time in figuring out the answer to that question.

Any project should have a person responsible for making sure that the CM strategy is maintained. The strategy should of course cover the purposes of CM that were mentioned in the previous chapter.

When working with software development you come across different problems. Three of the most common ones are the double maintenance, the shared data and the simultaneous update issues. A good CM strategy and a good CM tool will help counteract these problems.

Double maintenance refers to the issue of having more than one copy of any given software. The problem with having numerous copies is that eventually they will differ. If for example, two copies of a code exist and a bug is found, the bug fix must be performed on both copies and more important, the fix must be identical for both. This however seldom happens, especially not when working with big projects. Many times code is shared between developing teams in different projects. When a bug is detected, it is not always communicated to the other projects or perhaps the bug has not manifested itself yet for the other development teams sharing the same source code. Different solutions to the same bug in different projects will lead to numerous set of codes that over time, as more bugs are fixed or features are added, will behave in utterly different ways even though they essentially perform the same task. Consequently, the developers will have to maintain and update two different set of codes.

The problem concerning *shared data* is when several developers access the same program and work with it at the same time. When for instance two developers work within the same code in different subroutines, the actions and code altered by one developer can affect the code being overviewed by the second developer. This will happen even if the developers are aware that they are working with the same code. No matter how well informed they are it is never a good idea to work with code at the same time. The worst-case scenario is if a change in one subroutine causes the entire program to crash due to changes in the other subroutines. Nevertheless, even if it just alters the data output it could have serious affects on the second developers' code.

The last problem is more difficult to solve since it involves both of the previous mentioned. The *simultaneous update* problem reflects on the combination of the dilemmas of solving the shared data and the double maintenance problems. To solve the shared data problem the developers could, for instance, each have a copy of the code and that they work with. This might sound like a good solution but it will eventually lead to the double maintenance problem. If they instead work with only one copy, they are back to the shared data problem.

These three problems are common when working within a software development team and they are all solved by having a good CM strategy or better yet, a CM tool that implements this strategy. [2]

3.2 Basic concepts

To understand what is needed in a CM strategy, we first need to understand exactly what a development team does. Despite what you might first think, it is not often that development teams work with just one file i.e. one main program with all the code in it. This is actually rarely done and in some companies it is never done. Instead code is divided into different sections or subroutines, each with the aim of solving a specific task. These tasks may differ slightly depending on different factors such as hardware, design adaptation and so on. Developers then "pick and choose" from subroutines to build up a main program.

Configuration management must be able to manage all different types of versions that exist so that work can be done as effectively as possible but also to avoid any kind of extra work in maintaining the different versions. The versions can be divided into revisions and variations and the difference between them will be explained in chapters *3.2.2 Revisions* and *3.2.3 Variants*. [2]

3.2.1 Storage

Storing can be done in a couple of different ways. One way is that each main program is stored separately; this however means that subroutines are saved in multiple places. Two different main programs may consist of several shared subroutines; this will lead to the double maintenance problem as the same code exists in different locations. A better solution is to keep the subroutines in a repository from which developer then collect the needed subroutines for their programs, thus avoiding the double maintenance issue. Having code stored in one place will reduce the problem and answer the question of "what program/code is this?"

Code can, as we mentioned, exist in different versions and all different versions should be stored. [2]

3.2.2 Revisions

A new revision is created when a program advances to a new stage. This means that it has evolved and has become better than its predecessor in some way. This can for instance be achieved by having less errors, new features or better performance than an earlier revision. [2]

"The intention of the creator is that the newest revision will make the older ones obsolete."[2]

3.2.3 Variants

Variants differ from each other in some way; this can for example be due to adaptations to different conditions, customisations, other I/O parameters or support of different hardware. Variants do not supersede each other, instead they are alternatives and substitutes. They exist as equals, with no one being better than the other. [2]

[2] Babich, Wayne A., Software Configuration Management – Coordination For Team Productivity, page 23

3.2.4 History log

The history logs serves as an audit trail for developers to follow the work that previously has been performed on a version. A good history log can in many cases reduce the time spent on debugging. For example if a development team returns from a break and find that the code suddenly does not work, they could with the help of a good log simply check what the latest actions performed on the code were. This means that they do not have to spend time looking at the entire code to find the error. More importantly if you are looking at old code while debugging, the log can give an explanation to why certain modifications were done. Perhaps many, or some, of the modifications need revising due to new hardware, new I/O-signals and so forth.
A good log consists of the following[2]:

1. Name of the development tool – compilers and version.
2. Identification of inputs or source code used – type of code the revision builds on.
3. Statement of options and arguments used by the tool.
4. Reason for choosing the options and arguments.
5. Person in charge for modifying the data.
6. Date and time.

It is very important to change the name of a version whenever a change has been made. If it is not done the whole concept of storing versions of files and keeping a log will be lost. It is also common sense to never re-use names. A good CM tool automatically gives a new revision a new revision number and spare the programmer the work of manually renaming it.

The history log also plays an important role when recreating set-ups in case that end-users report complaints and bugs. The end-user merely needs to give the version ID to the developer and with the help of the log the information of that particular version can be accessed. [3]

3.2.5 Reproducibility

Sometimes the history log is not enough when debugging and lot of time has to be spent on analyzing the code with conventional methods. This is not an easy task to perform manually and often debuggers are used for this. To be able to use a debugger a copy of the entire code that is being debugged must exist. This means that storage space becomes a big issue. In the case when there are not many versions a company could keep a copy of each version but in the case that there exist hundreds of different versions it is not often done. The history log now works as a blue-print for the particular code version in question. The log holds information about the configuration of tools, input code and options that were used. With the help of this, the original version can be re-created.

[2] Babich, Wayne A., Software Configuration Management – Coordination For Team Productivity, page 36

Being able to re-create programs is very useful in the case when many variations of code exist and this is often the situation when code is customised after customer specifications. If a costumer reports a bug, it is crucial to be able to reproduce the specific conditions for that customer and thereby replicate the error so that it can be fixed.

To be able to perform a good reproduction from a history log it is often required that tools and inputs listed in the log are frozen which means that they cannot be deleted. Configuration management often requires that anything stated in the history log is frozen. This will of course require a large amount of storage space but considering that developers that do not work with CM often store data in accordance to their own desires and without any structure, they will often not use the storage space in the most efficient way. [3]

3.2.6 Global Repository

The database of a shared project is often called a global repository or just repository but it can have many different names. The repository can contain many different components, for example a project can include source code, object code or different subroutines. The repository should, in most cases, have a restricted access. This is to ensure that it is well protected and that changes cannot be performed directly on it. The repository will contain the definitive copy of a subroutine i.e. the latest approved code. Developers should never be allowed to work with the global repository directly. Instead they should work with a local repository. [2]

3.2.7 Local repository

The local repository is the developers' private workspace. This is where the developers can alter code without having to worry about the consequences and without affecting someone else's work. The local repository can be a copy of an entire other repository or just parts of it. It is completely up to the owner of the repository to decide what he needs in it to perform the work. [2]

3.2.8 Locking

Many times while working with code it is required that only one person has write permission. This is particularly true when working with shared repositories to avoid the shared data problem and consequently the simultaneous update.

The developer who locks a file is given the possibility to copy, read and write to repositories, folders, subroutines or files that the lock affects. Other developers however can only copy and read the component that the lock has been applied to. [2]

3.2.9 Stability

Stability is a big concern when it comes to working with software development. When working with a subroutine it is seldom only that particular subroutine that needs to be tested, often a set of subroutines has to be tested to ensure that they are compatible and work together. For example a program might consist of subroutines A, B and C, each being developed by different developers. This is a problem for developers since they need the subroutines to be static during the time they are working with and fixing them. At the same time other people cannot be asked to stop working and developing other subroutines as that would mean dead-time.

Sometimes developers need the database to be stable and at other times they need it to be changeable. The problem is that since every member in a project is working against one repository with different subroutines, they all want different parts to be stable and changeable at different times. Using global and local repositories allows each developer to decide for themselves which part they want to copy and maintain stable in their personal workspace. Problems can however still occur as we will describe in *Example 1*.

A developer is fixing a bug reported in a program which consists of several subroutines, A, B and C, after analysing he finds that the bug is located in subroutine B. The next step is to copy the subroutines to his local repository, with a lock on subroutine B since this is the code he intends to alter. At the time the developer does the check-out all the subroutines are at version 2.0. He now works for two weeks fixing the bug and testing at the same time that the main program still works with the fix. After the two weeks he does a check-in with the new version of subroutine B (version 3.0), now with the new fix. Unfortunately sometime later he starts getting calls that nothing is working anymore and that it is to do with the new subroutine he has just fixed. He is very confused since he has followed all the rules and regulations concerning CM.

Example 1: Stability problem.

The problem that has occurred is that during the two weeks it took him to fix the bug a newer version of the other subroutines has been developed and added to the main repository. Remember that to ensure stability the developer copied all the subroutine to his local repository at beginning as he should. The mistake he made was that at the end of the testing he did not update the subroutines A and C before checking in the new code and therefore he never made sure that the bug fix for B worked with the latest versions of those subroutines.

This is a big dilemma and there is no good way of overcoming this except for using some basic work techniques. For instance, always checking out the latest versions from the global repository and testing against them. Some CM tools also allow e-mail notifications to be sent out when a specific file is altered and checked in to the global repository. This is a great help to counteract the problem in *Example 1*. [2]

3.3 What is a CM tool?

The answer to the question “What is a CM tool?” is more complicated then it seems. In *Chapter 3 Theory: Configuration Management* we discussed what the functional areas of CM are. It is easy for someone that is not well informed of the functional areas of CM to think that it is all about version control and this is often the way companies look at it. This is however not the case [1][4]. The reason that version control often gets mixed up with CM is probably due to the wrong definition of version control. Version control is a part of the CM model but when people talk about version control they are actually referring to CM.

As previously mentioned CM covers four basic areas: Identification, Control, Status accounting, Audit and review. When looking at what CM covers it is easy to get overwhelmed by the amount of information. Still when choosing a tool you should look at what aspects of CM that are interesting for your company and then choose a tool and strategy based on that.

It is a good idea to do a thorough evaluation of the company to find out exactly which features and aspects that are needed or wanted by the end-users and by the company management. This will make it much easier to select which tools to evaluate as many CM tools have different features.

A good CM tool has much functionality. It can handle files and manage the changes made to them. This means to store them in some kind of repository, track changes made to the files and make audit logs on events. Many tools also include branching, merging and features for comparing different versions of a file.

Some CM tools also allow documentation of and control of overall changes. This means setting up rules and regulations for how changes are going to be implemented. The tool identifies changes and allows workflows to be set up for implementing them and tracking them throughout their change-lifecycle. In addition, the tool also report on the status of the repository.

There are also tools that allow management and the creation of the development environment for an object. This means that the objects executables are reproduced, i.e. all the “raw” tools for the object are created. Some tools also allow the movement and installation of end products into a production system.

The version control aspect of the CM model is the main focus of this thesis and will be described in detail in the next chapter.

3.4 Version Control

Version control refers to a way of managing files and directories. The main purpose of version control is to manage and keep new versions of a file saved in a structured way. The idea is to allow the users to have access to old versions of a file to make it easy for them to recover any revision of their choice.

Everyone has done some kind of version handling either knowingly or unintentionally. And when it comes to companies that are involved with development of software they all have some kind of version control. If the version control system is good or bad depends on to what extent the issue has been driven within the company and the amount of time spent on finding a good solution. Not too far ago most companies had someone responsible for keeping software projects under version control, i.e. it was done “manually” and under supervision of one responsible employee. [1][5]

Lately the strategies for managing projects under version control have been implemented in software programs and there is no longer any need for a supervisor. As these programs have become more and more advanced, so have the features they offer. The features of course vary between different software but some common ones exist.

Some of the common features are:[6]

- **Backup and Restore –** files are saved when they are edited.
- **Synchronization –** share files and get the latest version.
- **Long-/Short-term undo –** revert to an earlier version of a file.
- **Track Changes –** leave comments/messages explaining why changes took place.
- **Track Ownership –** see who made a specific change.
- **Sandboxing –** check-out files and alter the content without changing the “main” file.
- **Branching –** branch a file and keep track of its changes separately from the main file.
- **Merging –** a branched filed can later be merged back into the main file.

Apart from having different features built in, there is also a big difference in how the version control systems are structured. There are two ways to structure a version control system. They can either be centralized or distributed and both have their benefits and drawbacks. Choosing which of these two systems to implement is not something that is done with ease. Many aspects have to be taken into consideration. For instance, how the company is structured and how the development procedure is structured just to name a couple.

[6] Azad, K.(2007) ”Intro to Distributed Version Control (Illustrated)”,
URL: http://betterexplained.com/articles/a-visual-guide-to-version-control/

Before deciding which structure to use it is a good idea to get an overview of the work procedures and also ask the developers what their day-to-day work looks like. Doing this will give a sense of which structure that best would suite the company. Sometimes you might work in centralized way even though a decentralized approach might be more efficient. The issue of choosing either a centralized or distributed approach will be discussed more in *Chapter 3.4.3 Centralized or Distributed?*. [5]

3.4.1 Centralized System

A centralized system uses a central server which holds a repository with all the data. Clients then connect to it and retrieve data from the repository and make changes to it. By then committing these changes back to the repository on the server, any other user can gather the latest version of the data. Systems that are structured in this way are simply called Version Control Systems, VCS.

The centralized system works in a master-slave relation. In this relation the global repository is referred to as the master and the slaves consist of the local repositories. Most of the times the setup in a centralized system consists of one global repository and multiple local repositories, one for each developer, see *Figure 3.1*. The global repository holds the main trunk of the file, this is always the latest and most updated version of it.

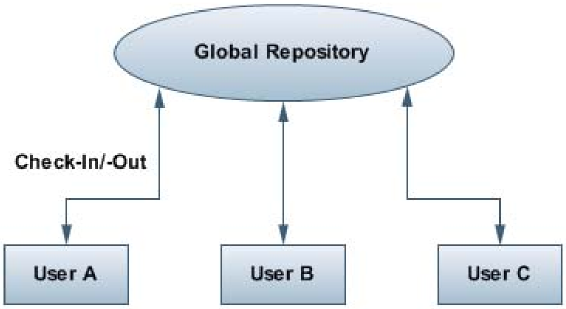


Figure 3.1: A centralized systems configuration of repositories.

The developers do a check-out of the global repository, either in its entirety or of individual files within it, depending on what the particular tool allows and which files the developer needs access to. These checked-out files now make up the local repository. Note that some applications do not automatically lock the global repository; instead the lock has to be applied manually by the person that made the check-out. If a lock is not applied this will mean that the global repository still can be checked-out by someone else.

Once a file is checked-out to the local repository it is free to be altered in any way that the developer wishes. When the developer is done working with a file, it is simply checked back into the repository, i.e. a check-in is performed. Each time an item is checked-in a change message has to be typed to go along with the check-in. The item also gets a new revision number each time it is checked-in, see *Figure 3.2*.

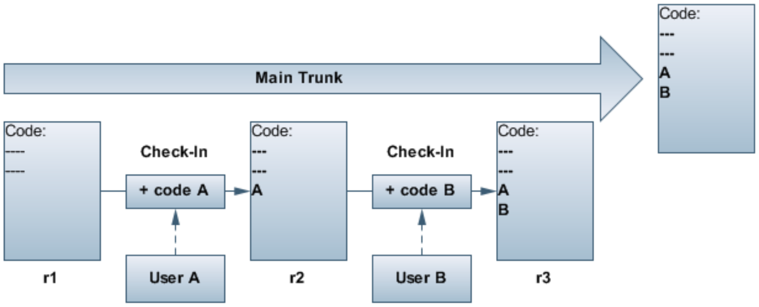


Figure 3.2: Basic work method for centralized system.[5]

At any time the latest version in the main trunk can be released. Releasing a version is often called tagging or labelling. Instead of the present revision number, the developer can tag the file as a release e.g. Release 1.0.

When working with a centralized system conflicts will occur if a file is not locked when checked-out and more than one person works with this file at the same time. If for instance developer “A” checks-out a file an add the “code A” and does not lock the file while doing the check-out, nothing hinders developer “B” to do a check-out of the same file and add the “code B”, see *Figure 3.3*. The changes that will be committed to the main trunk will be the ones made by the developer that first checks in his work. The developer who is “late” will get a conflict when trying to do the check-in since the file he has worked with no longer exists in the main trunk. Instead the main trunk will contain the file that the other developer checked in. This is the reason why it is very important to lock files whenever a check-out is performed and several developers can work with the same file at the same time.

It is also very important to unlock files when checking them back in. If this is not done, the file will remain locked and inaccessible to the other developers in the team. Unless the tool has a very good conflict resolver that allows easy merging or the development team is very comfortable with doing merges and resolving conflicts manually, locks should always be applied when checking out files. Some tools require that the locks are applied and removed manually and some tools do this automatically.

[5] Azad, K. (2007) “A Visual Guide to Version Control”
URL: http://betterexplained.com/articles/a-visual-guide-to-version-control/

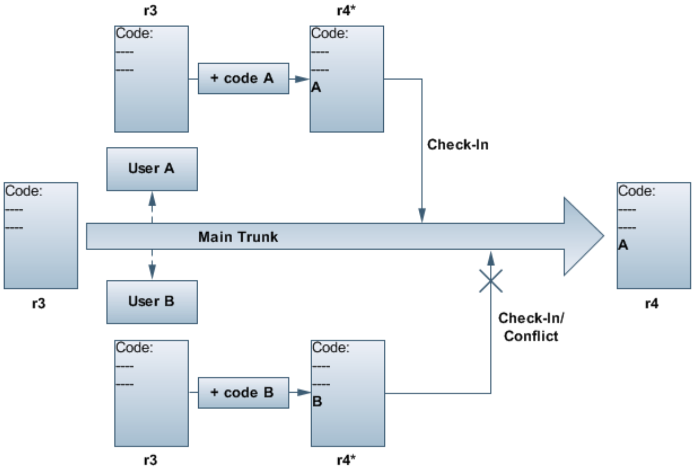


Figure 3.3: The figure shows how a conflict can occur.[5]

Branching is typically done when working with new ideas that are not in direct contact with the main code but still can have an effect on the performance of the main program. For instance, when developing an extra feature that could be included in a later release. By branching, the developer has the possibility to execute any changes and to experiment with code freely without worrying about the consequences. After working with a branch successfully the next natural step is to merge the branch back into the main trunk. Merging is such a special operation that it cannot be explained in broad terms more than the fact that is done to join different versions of files. The merge tools in the software programs all differ depending on the files that are supported by the application. The merge feature is an advanced functionality that for most tools only works with pure text files.[5]

3.4.2 Distributed System

In a distributed system each client has its own repository in which any changes can be made. These changes do not necessary need to be committed directly back to a main server or global repository, they can instead be shared to other clients directly. This structure is referred to as Distributed Version Control Systems, DVCS.

The distributed model differs very much from the centralized, not just in the set up of the repositories but also in the way in which work is carried out. Instead of the master-slave approach in the centralized system, the distributed has no hierarchy built-in so to speak. The repository of each developer is instead equal to each other, see *Figure 3.4*.

[5] Azad, K. (2007) “A Visual Guide to Version Control”
URL: http://betterexplained.com/articles/a-visual-guide-to-version-control/

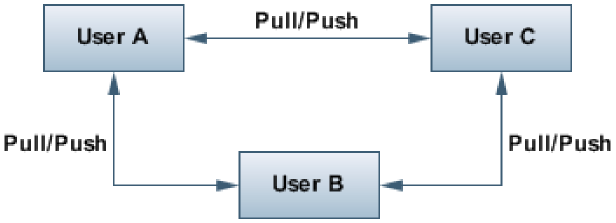


Figure 3.4: A distributed systems configuration of repositories.

Each developer has a global or local repository depending on how you wish to see it. The developers are free to share work with anyone else at any point in time, see *Figure 3.5*. No check-outs or check-ins are performed, instead pushes or pulls are preformed. Push means that you send files to a repository and Pull that you get files from one. In the distributed systems each developer works separately with their own repository and each developer's repository has its own revision history and log. There is the possibility to somewhat mimic a centralized system and this will be explained at the end of this chapter.

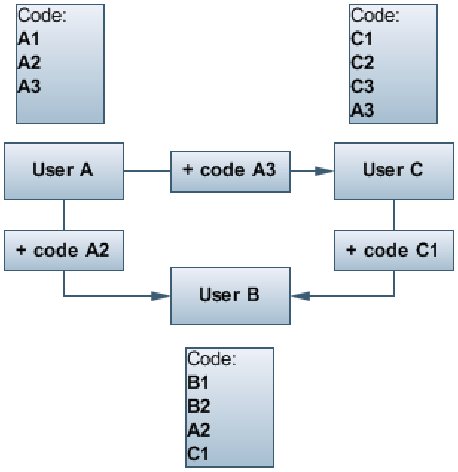


Figure 3.5: Basic work method for distributed system.

When pulling or pushing, two things can occur depending on if the file is new or if it has been used before. If the developer simply pushes a new file to someone else's repository, a simple update is performed. If instead a new version of a file is being pushed, a merge has to be performed. The update and the merge will also occur if a pull is performed instead of a push. Notice that every Pull is equivalent to a branch and every Push is equivalent to a merge when looking at it from a VCS point of view.

Working in a distributed manner can be very complicated. There is no latest version available in the same sense as it is in centralized system. It can also be very confusing to keep track of who made what. There is fortunately an easy solution to this problem which is to structure the system in a way that allows it to keep the flexibility of a distributed system and at the same time get the organized aspect of a centralized system. By having an “extra” repository to which only approved changes are pushed, you can get the system to mimic a centralized model, see *Figure 3.6*. The decision of which changes that should be approved should consequently be made by someone with an overview of the task at hand e. g. project supervisor. The extra repository does not necessarily have to be placed in a server; it can just as well be the repository belonging to the person that has to approve the changes. [6]

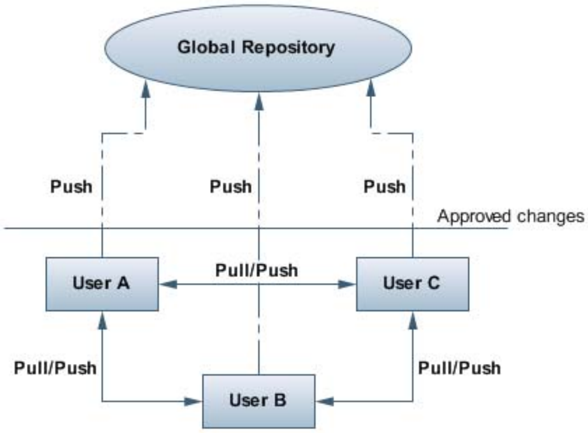


Figure 3.6: How to mimic a centralized system in a distributed system.

3.4.3 Centralized or Distributed?

An advantage of using the distributed structure is that the clients do not need to be connected to a server to share updates made to a file or project. On the other hand, since clients are free to share changes as they want, it can be hard to track what version of the files that are the latest at any given time. This can be resolved by having two repositories, one with a working version and one with a test version, where the working version is continuously updated as features that are added or changed by clients are approved.

Which of the systems to use varies from time to time and it depends entirely on the projects and the way in which the developers work. When working with a big scale project with many clients and numerous files, a distributed approach might preferable. If the workload is small and only involves few developers that are located in the same area, a centralized system might work better. It is all a matter of taste and demands. [6]

3.5 Quality Assurance and Quality Control

Quality Assurance (QA) and Quality Control (QC) are two terms that are often mixed up with each other and misused. The main mistake is that very often QC is mistaken for QA. Most people refer to QA as testing and making sure that product has reached a sufficient level of quality. This is however not QA at all but instead QC.

QC refers to testing and investigating if products fulfil their specifications and inspections of work products like code, design, requirements and documents. Most people would incorrectly put one or all of these things as a part of QA.

QA has actually nothing to with validating the end product, whatever it might be. QA instead refers to the measures and steps taken to assure that a product ends up with a certain quality. This includes everything in the work process that leads up to a test. One way of putting it is that "if QC is about detecting defects, QA is about avoiding them!"[7]

In essence QC is about checking and evaluating the results of a process while QA is about checking and evaluating the process itself. Both parts are necessary for ensuring a good end-product but the better the QA is, the less work needs to be done in the QC stage.

When it comes to QA and QC in regard to Configuration Management there is a mixed opinion whether they should be included or not. It seems as QA and QC very well would fit within the CM model since the goal of CM is to make sure that work is carried out correct and that the end product is good enough. [7]

There is no feature in any CM tool that directly deals with the QA and the QC parts of the CM model but using a tool will automatically ensure that QA is implemented since the tool makes sure that the right processes are followed and that the correct code is released. With CM companies will get more control over software assets and common mistakes that can be made without CM will be easier to avoid. CM ensures that the right version gets released and that no rogue files end up in a build. [1]

Using a CM tool will also cut down the amount of software bugs. "Fewer bugs are found in released products …, typically, 50% of the bugs that customers find are bugs that could have been avoided had there been good CM in place."[1]

[1] Dart, S. (2000) Configuration Management: The Missing Link in Web Engineering. Page 86

[7] Koch, A.S. (2006) "Testing vs. Quality Assurance"
URL: http://www.cmcrossroads.com/content/view/6782/120/

4 Tetra Pak

Tetra Pak is a part of the Tetra Laval Group. The group consists of three companies; Tetra Pak, DeLaval and Sidel, see *Figure 4.1*. This is the structure of the company, not the official names of the different departments. Tetra Pak Processing Systems where this thesis has been written is located within Tetra Pak Processing Solutions.

DeLaval is a full service supplier to milk producing farmers and offers complete systems for milk production and animal husbandry.

Sidel is one of the leading companies in the world for packaging solutions for liquid food, such as carbonated drinks, water, cooking oil, milk, tea, fruit juices and beer.

Tetra Pak is a developer, manufacturer and promoter (marketer) for complete process, packaging and distribution systems for foodstuff. One may at first just think of liquid foods like milk and cream when thinking of Tetra Pak packages. The fact is that foods like ice cream, cheese, dry foods, fruits, vegetables and pet food are all food products that are processed and packaged by Tetra Pak manufactured machines. Even though the market has been broadened, the dairy products still dominate it and accounts for 2/3 of the foodstuffs packaged by Tetra Pak.

Beside this the company also offers a range of equipment such as separators, homogenisers, heat exchangers but also conveyors, tray packers, film wrappers, roll-containers and so on. In addition to this, there is also a software side that offers services for factory planning, control and monitoring of plants, computerized logistics studies, training, follow-up service and marketing assistance. [8]

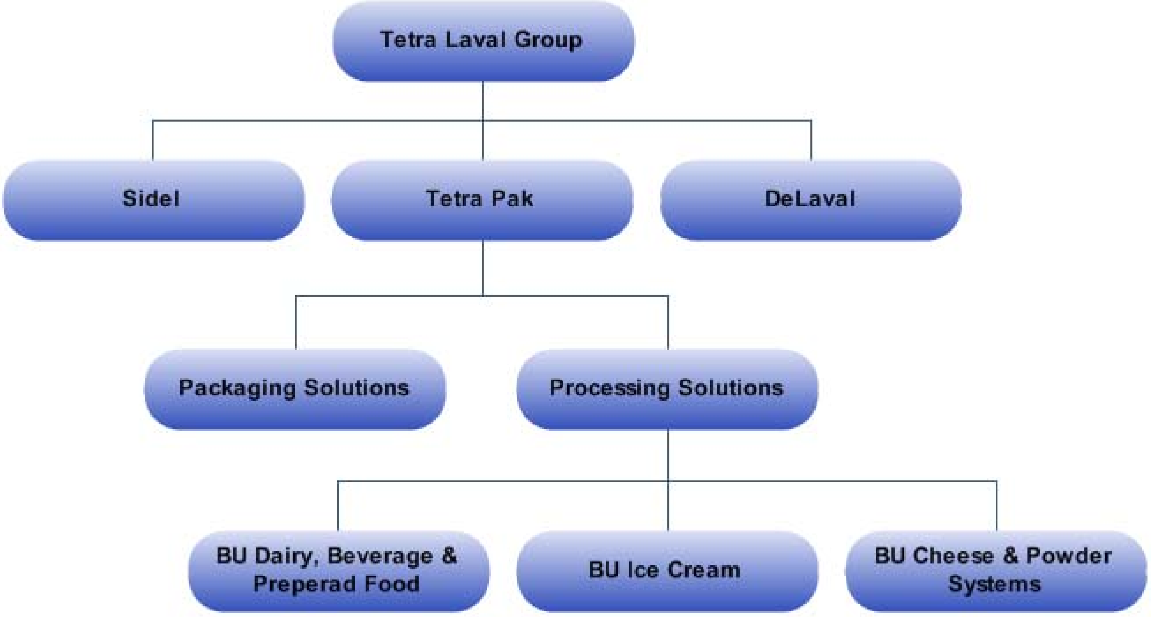


Figure 4.1: Structure of the Tetra Laval Group and its industries.

4.1 Tetra Pak History

The company AB Tetra Pak was created in Lund 1951 by Ruben Rausing and Erik Wallenberg as subsidiary company to Åkerlund & Rausing.

Åkerlund & Rausing was a packaging company, and it was within this company that the first directions were taken to develop and produce a milk package.

Figure 4.2: Rausing with his sons.

The idea was to create and develop a milk package that required a minimum of raw materials and had optimal hygienic. It was with this in mind that the idea of a tetrahedron package took were formed. During the years 1944 - 1951 development proceeded. Among other things a new way of belaying paper with plastic was created and also a new process for sealing the package beneath water level.

The 18th of May 1951 Tetra Pak presented a new packaging system and it got a lot of publicity straight away. The fist machine for tetrahedron packages was delivered the following year to a dairy plant in Lund. In 1954 the first machine was delivered abroad to Hamburg, Germany and in 1959 the production capacity for packages surpasses 1 thousand millions.

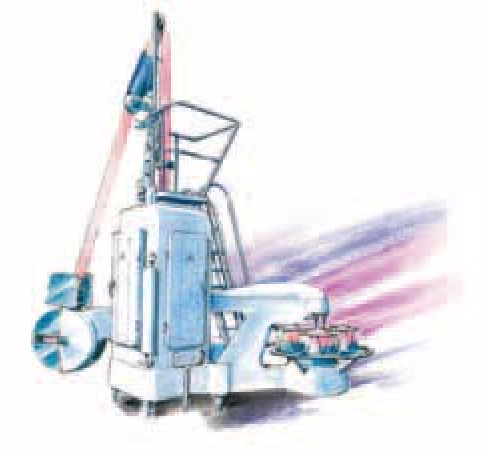


Figure 4.3: One of the first packaging machines for tetrahedron-shaped cartons.

The development of the first machine for aseptic packaging started in 1956 and 5 years later the machine was presented on a press conference in Thun, Switzerland.

In the mid 60’s it became common with milk with long durability as a complement for pasteurized milk.

The parent company Åkerlund & Rausing was sold in 1965 but AB Tetra Pak was kept and in the year 1970 there was a restructuring throughout the company and Tetra Pak International AB was created. AB Tetra Pak became the production company and AB Tetra took care of sales within Sweden. The following year the production of Tetra Pak packages reached 10 thousand millions.

A decade later in 1981 the executive board of the group moved to Lausanne, Switzerland and a new international headquarter was created, Tetra Pak Rausing SA. The international headquarters name was changed to Tetra Pak International SA in 1989. The company AB Tetra Pak in Lund stayed as the corporate technological head office of the group.

The 10th of August 1983 the founder of Tetra Pak – Ruben Rausing passed away.

In August of 1991 Tetra Pak's acquisition of Alfa-Laval was seen trough and in 1993 after some restructuring the two companies came together under the common name Tetra Laval and in the same year the total production of packages exceeded 60 thousand millions. [8]

4.2 Business Unit Dairy Beverage and Prepared Food

As shown in *Figure 4.1*, Tetra Pak Processing Solutions is dived in three subdivisions, one of them is Business Unit Dairy, Beverage and Prepared Food. The structure of BU DBF is shown in *Figure 4.4*. [9]

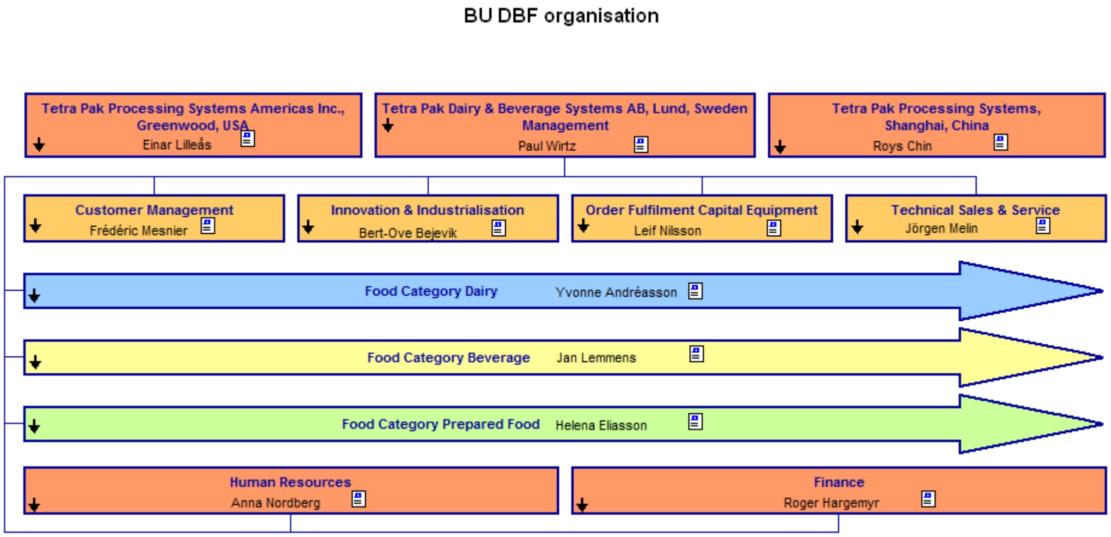

BU DBF organisation
Tetra Pak Processing Systems Americas Inc., Greenwood, USA
Einar Lilleås
Tetra Pak Dairy & Beverage Systems AB, Lund, Sweden Management
Paul Wirtz
Tetra Pak Processing Systems, Shanghai, China
Roys Chin
Customer Management
Frédéric Mesnier
Innovation & Industrialisation
Bert-Ove Bejevik
Order Fulfilment Capital Equipment
Leif Nilsson
Technical Sales & Service
Jörgen Melin
Food Category Dairy
Yvonne Andréasson
Food Category Beverage
Jan Lemmens
Food Category Prepared Food
Helena Eliasson
Human Resources
Anna Nordberg
Finance
Roger Hargemyr

Figure 4.4: Structure of the BU DBF organisation.

4.2.1 Flowcharts

The flowcharts were constructed together with our supervisor to help us understand the chain of commands within BU DBF when it comes to software development.

In *Figure 4.5* the ownership and availability of documents and templates are shown. The chart also explains the way in which a product is produced and which departments and employees that owns the different product assets, see

Figure 4.6 explains the order in which request for change (RFC) handling should be carried out.

The charts should be more or less self-explanatory but to clarify the RFC chart, ABEO is the system being used to report and follow up the requests for change within BU DBF.

Both of the flowcharts were handed out along with the survey and discussed during the interviews. The flowcharts presented below have been slightly revised from the ones handed out. A few spelling errors have been corrected and the name of ART has correctly been changed to AR. The original flowcharts can be found in *Appendix 2*.

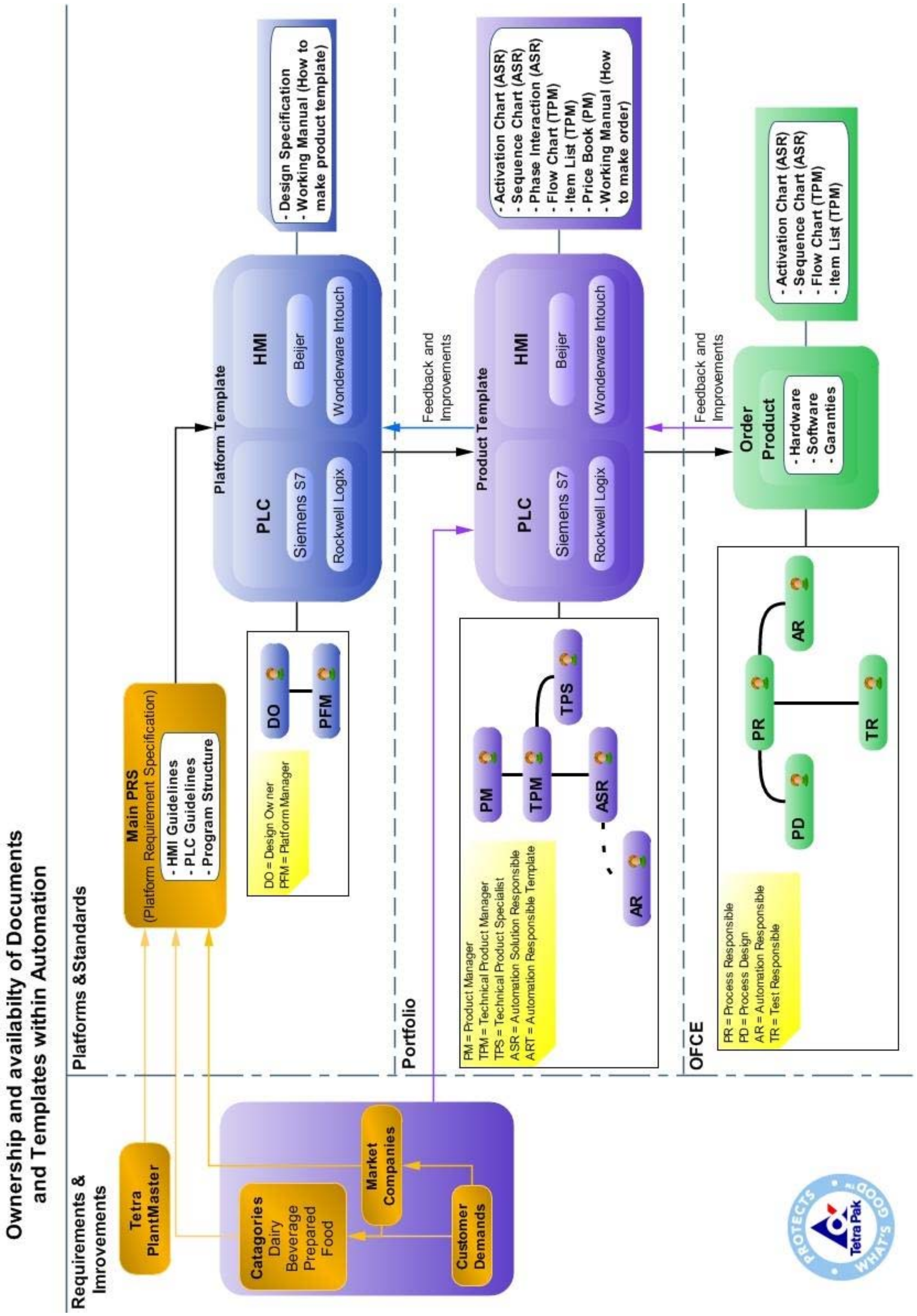


Figure 4.5: Ownership and availability of Documents and Templates within Automation.

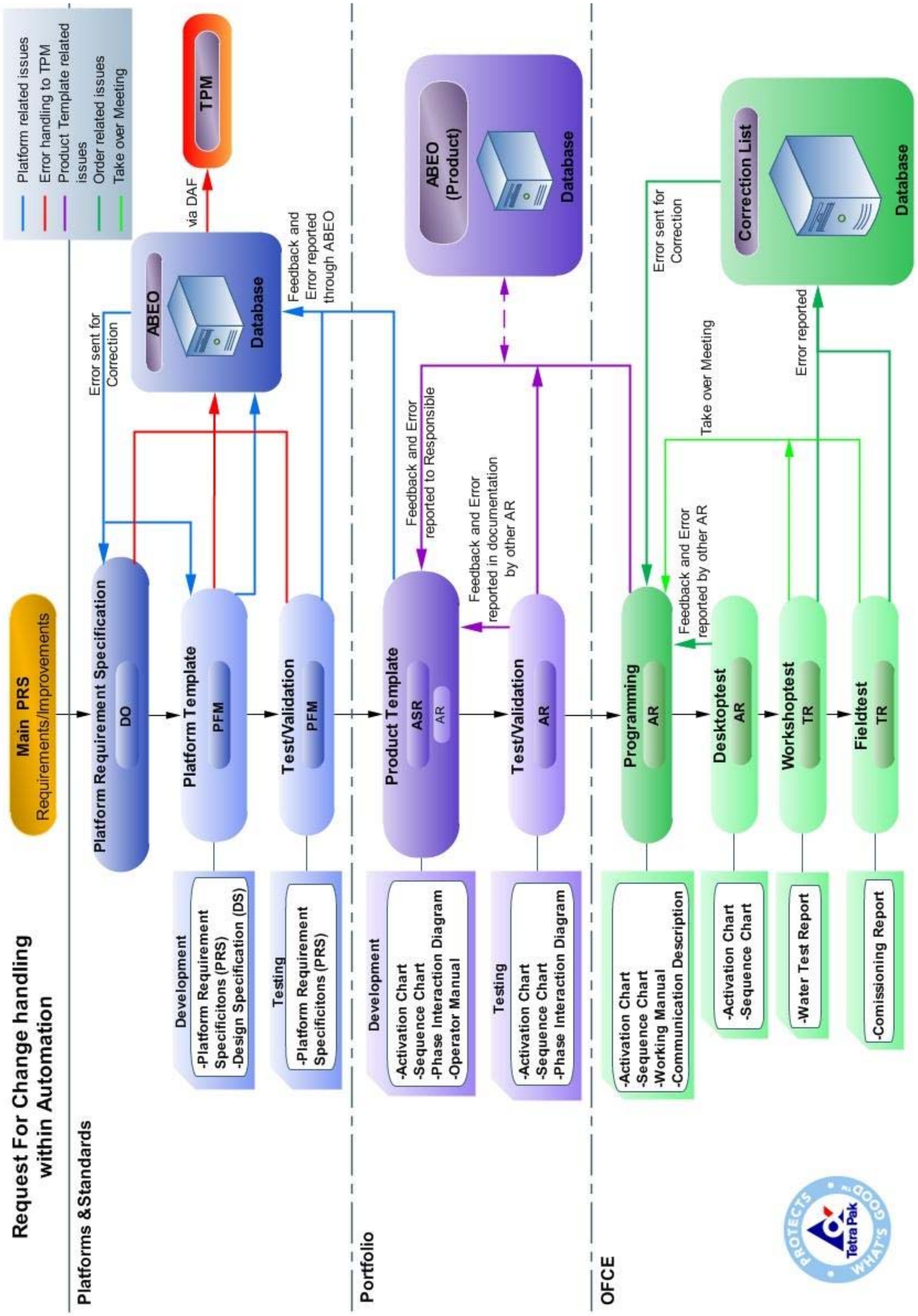


Figure 4.6: Request For Change handling within Automation.

4.3 Platforms & Standards

Platforms & Standards is a cross category division and this means that it is found within all the food categories; Dairy, Beverage and Prepared Food. Platforms & Standards are maintaining four platforms – Platform Heat, Platform Storage, Platform Siemens S7 and Platform Allen-Bradley Logix. Platform & Standards is located within Innovation & Industrialisation and the purpose of the division is to standardize these platforms. The work of the department includes finding common elements and interfaces for the different platforms. The common features from the categories and platforms are then maintained and developed in templates by the Platform & Standards department. [9]

4.4 Employees

Within BU DBF the employees have different roles. The hierarchy of their positions can be viewed in *Chapter 4.2.1 Flowcharts* and the meaning of their roles is explained in this chapter.

4.4.1 Design Owner BU DBF

The design owner works in the Platform & Standards area, see *Chapter 4.2.1 Flowcharts*. The design owners' responsibility is to evaluate the efficiency for all software platforms, product templates and to collect and set the platform requirements. The efficiency is measured in regards to product performance and manufacturing costs. The role consists of support, quotations (pre-sales) in large projects where many branded processing units are integrated.

4.4.2 Platform Manager

The Platform Manager, PFM, works in the Platforms & Standards area, see *Chapter 4.2.1 Flowcharts*. The PFM's role is to evaluate the efficiency for their platforms i.e. automation and product platform templates. The efficiency is measured in regards to product performance and manufacturing costs. The role also consists of maintaining, improving and developing the BU DBF platforms. This is done in agreement with the users of the platforms. [9]

4.4.3 Automation Solution Responsible

The Automation Solution Responsible is just like the name suggests responsible for automation solutions. This means to develop and maintain the solutions for the product portfolio in complience with Platforms & Standards and BU DBF management system. The ASR is also the one that receives and manages the feedback for improvements that regard the Portfolio management. [9]

4.4.4 Automation Responsible

Within BU DBF there are approximately 40 people employed as AR (Automation Responsible) but the exact title can vary between categories. The AR is delegated from the ASR to help maintain product templates. These are the employees who create the software for the PLC and HMI, both in the Portfolio and OFCE (Order Fulfilment Capital Equipment) area. The AR works within the OFCE sector of the product lifecycle, see *Chapter 4.2.1 Flowcharts*. The AR's role consists of making the necessary customisations to software and hardware to meet valid demand specifications. The AR also works with quotations, product support and development projects. [9]

4.4.5 Test Responsible

As the name suggests the Test Responsible has the responsibility of performing some of the tests in the OFCE sector, see *Chapter 4.2.1 Flowcharts*. The role consists of preparing the tests according to the valid routines, perform the test against established check lists and verify performance criteria of modules in regard to technical specification and report feedback to the organisation. [9]

5 Empirical Foundation

The data for the thesis was collected through a survey, interviews and by testing various CM tools. The way in which the survey, interviews and software testing specifications were constructed will be explained in this chapter.

5.1 Survey

To get a good view of what the company is missing within CM we decided to hand out a survey.

5.1.1 Quantitative study

First of all we wanted to learn more about what people actually knew about CM. To do this we created a survey targeted at ARs and TRs. The total number of ARs and TRs was about 45 and we felt that conducting oral interviews with this many people would not fit within the time limit for the thesis. On some questions we also made space for the surveyees to write their own comments.

5.1.2 Creating the survey

The questions we wanted to get answered were mostly about CM in general.

- How well familiar are people with CM and are they using it?
- Are they working in a structured and organized way making it possible to manage the workload?
- How well are the version control working?
- Would they like to see any changes in the CM routines?
- Etc. etc.

The questions were formulated from our own knowledge about CM and inspiration was gathered from Dart S., Configuration Management: The Missing Link in Web Engineering [1]. We also searched for similar surveys to see what type of questions might be suitable.

The questions were discussed together with our supervisor. This was done to make sure that the questions were relevant and stayed within the focus of the thesis assignment. The answering alternatives for each question were thoroughly discussed and formulated. The alternatives vary from “Yes or No” to grading answers on a scale. We wanted to keep the layout of the answering alternatives as unanimous as possible but in cases where we felt that a simple “Yes or No” would be good enough, we settled for this mainly to make it easier to come to concrete conclusions.

On the first page we decided to have a short description of CM. Filling out a survey about something you do not know anything about is pointless but at the same time we did not want to give to much away. The survey was constructed to find out what people already knew about CM, not how much they were able to learn from a longer description.

5.1.3 Handing out the survey

We reasoned that the best way to hand out the survey was to personally contact all the ARs and TRs. By doing this we could introduce ourselves and explain briefly the purpose of the survey. Hopefully this would make the ARs and TRs more willing to help us in our work. Sending the survey by mail would make it easier to ignore it, especially since everyone seems to be very busy all the time. Unfortunately we could not find everyone that we were looking for, even after trying for several days. To hand out the survey to the persons we could not find, we placed it in their mailboxes and asked them to fill out the survey within a week.

5.1.4 Analysis of the survey results

We inserted all the data from the surveys into Microsoft Excel and made graphs to the questions that had answering alternatives. After that, we speculated about the distribution of answers and made a few comments about it. The comment was then complemented with a conclusion. We also tried to give an explanation to why the distribution of answers looked the way it did.

5.1.5 Presentation of data

The results will be presented in *Chapter 7.1 Survey* according to the following model:

Question
The survey question to be evaluated.

Diagram
Diagram of the answers. The numbers of answers are also included since some people chose to leave some blank answers. The scale on the Y-axis is in percent and each answering alternative correlates to one block in the diagram.

Comment
Comments and speculations about the result.

Conclusion
Conclusions made from the diagram and the speculation. In some cases there are suggestions on how to improve the distribution of answers in a more favourable way.

5.1.6 Reliability of the survey

The survey was handed out to over 40 Automation Responsible developers within the company. Due to recent restructuring the lists of name we had at our disposal were a bit out of date. Some of the people were not employed anymore or had other duties. Some were on business trips or working elsewhere at the moment and therefore not possible to get a hold of. We waited two weeks from the day we started handing out the survey until we decided that the deadline had passed. This left us with 22 surveys filled out and this must be considered a fairly good foundation to be able to come to essential conclusions.

We soon realized that the overall knowledge of CM was very low. This resulted in a lot of "uncertain/don't know" replies on many of the questions. This definitely made it harder to come to good conclusions on some questions.

Some of the developers we handed the survey to showed unwillingness to fill it out with the motivation that it felt like some sort of a test. This was obviously not our intention and that was the main reason to why we decided to keep the surveys anonymous. It still probably led to loss of answers on some questions or even loss of some whole surveys.

Some surveys seemed to be filled out in a rush with no elaborations whatsoever on any questions but they were few in relation.

Most surveys seemed to be filled out with reflectance of what the questions meant and some elaborations were really good and provided us with very useful information.

Based on these facts the overall reliability of the survey can be ranked as good, especially on questions where the majority has left the same reply.

5.2 Interviews

To complement the survey we performed interviews and the purpose of the interviews was to get more qualitative answers to questions about CM. The length of the interviews varied between 15 minutes and an hour depending on how elaborate the interview objects were.

5.2.1 Finding interview objects

The interview objects were ASRs (Automation Solution Responsible) and PFMs (Platform Managers). These PFMs work in the Platform & Standards department and instead of working with product and order specific applications, they work with platform templates that are later being used in the Portfolio and OFCE groups. Having a working CM implementation would ensure better control of all template versions. Therefore the input from these people was extremely important.

5.2.2 Half structured layout

The same questions as on the survey were asked but the interview objects were asked to answer freely. No predefined answering alternatives were given. Complementary questions were improvised and asked as the interviews went along. This could in some cases give us a deeper understanding of, for instance, the level of security and how files are being managed. Questions about quality assurance were also added.

5.2.3 Conducting the interviews

We brought recording devices to each interview but not everybody felt comfortable with the idea of being recorded. Even though the recordings only were meant for our personal use to be able to better analyse the result, we had to accept the interview objects' wishes. A total of 7 people were interviewed and 5 of those agreed to be recorded.

5.2.4 Analysis of the result

To analyse the result we listened to the recordings and read through our own notes from the interviews to try to find new approaches and ideas.

It did not take many interviews until we realized that the answers were more ore or less the same to all questions. The overall knowledge about CM was low and most of the interview objects did not have much to say. A few interview objects were familiar with CM, were very elaborate and provided us with a lot of useful information. The answers from those people covered most of the answers of the ones that had little experience of the area and therefore not that much to say.

5.2.5 Presentation of interview results

The result from the interviews is presented as a bullet list with statements from the interviews, divided into four categories – General work routines, Quality assurance and quality control, Wanted features in a CM tool and suggested improvements in the CM routines and Other. It is important to point out that the statements are not quotes. Due to the similarity of answers from the different interviews we have tried to formulate the statements to cover answers from multiple interviews. Many of the findings were, as stated before, expected and has been left out since other parts in this paper cover them. Only answers that have contributed with useful information and new input have been included in the result.

The result from the interviews has been kept in mind while performing the software testing and also when formulating *Chapter 7.4 General Discussion*.

5.2.6 Reliability of the interviews

The reliability of the interview must be considered high. The main problem you would expect to encounter when performing interviews in Swedish and presenting the result in English is translation errors, but this is not the case here. Only the most essential information has been included and the translation did not cause any difficulties.

5.3 Finding a CM tool

The main purpose with the thesis is to find a CM tool for handling of PLC, HMI and the documentation that are related to them. This includes finding and evaluating different tools for the Version Control aspect within Configuration Management. Since the tools have different key features and the fact that they support different file formats, means that finding a tool that meets the requirements could only be done through various tests.

The stated file formats that should be supported are the following:

Rockwell RSLogix 5000	.ACD
Siemens Simatc S7	.zip
Beijers E-Designer	.mpa
Wonderware Intouch	.zip
AutoCAD	.dwg
Microsoft Word	.doc
Microsoft Excel	.xls
Microsoft Notepad	.txt

5.3.1 Software Testing Specifications

To use a CM tool it must be compatible with all the development tools. In one way all the CM tools are "compatible" since they can be configured to disregard which file type they are working with but the drawback with this is that some features that the tool offers for known file types will be unavailable. There are on the other hand more requirements to the application. Therefore a series of tests were done to analyse the most basic and common functions.

The tests performed were the following:

- Create user groups and add users to ensure a safe system with possibility to have limited access to certain folders or files. If there are other features in regard to safety settings that the application offers, these will also be evaluated.

- Compare a newer revision of a file with a previous one graphically, either with a built-in compare tool in the CM tool or with a plug-in. If the development application has a separate compare tool, the possibility to use this from within the CM tool will be tested.

- Conflicts can occur if there is a possibility for several users to check out the same file from the repository. Even though the preferred way of working, according to the interviews, are to only work one person at a time on a file, it still might happen that several people work in the same file at the same time. Therefore the CM tools that can create conflicts must have good conflict resolving features. These will be investigated.

- The overall configuration possibilities will be important. It is obviously a drawback if it is complicated to create folders and moving files around. Simple tasks like these should be smooth and not take any longer than doing it manually in the Windows Explorer.

- The audit log will be one of the most used sections of the CM tool. This is where you look up the changes made between different revisions and locate older versions of sought after projects. The possibility to search the log and overall appearance will be investigated.

- Our first impressions of the application will be taken into account. Being used to work with computers and computer applications every day, the first feeling you get when exploring a new program often tells you a lot. Everything from appearance to user friendliness will be considered. It is important that it is easy to get started with the CM tool and the learning curve cannot be too steep.

Some flaws in the CM tool might be compensated with good features in other sections. If there for instance is no way to graphically compare different revisions, a good audit log might be able to provide you with enough information to dismiss this feature.

There are more test cases that could be performed, such as branching and tagging (or labelling). If any of the combinations of development software and CM tool do not pass the most basic tests, there is no need for further investigation because the tool will not be used anyway.

5.3.2 Presentation of Data

The results will be presented in *Chapter 7.3 Software Testing* according to the following model:

Discussion

A discussion with our impressions and results from the testing in relation to the test specifications in the previous chapter, and with the wishes of features from the survey and the interviews in mind, will be presented in this area.

Compatibility Table

Application	Filetype	Supported	Ability to show differences
Application name	File suffix	Yes / No	Yes / No

The table consists of four columns.

Application contains the application name and *Filetype* the file suffix for the application.

By *Supported* we simply mean if the CM tool has built-in support by default for the tested application. This means that the CM tool recognizes the specific filetype and not just treats it as file regardless of the suffix.

With *Ability to show differences* we refer to some sort of built-in feature in the development application that the CM tool utilises to compare different revisions, or some feature in the CM tool that can perform this operation.

5.3.3 Software Testing Limitations

All the tested CM tools had different system requirements and acquiring the correct versions of all those components was challenging. The freeware applications were easy to locate but hard to set up and the commercial tools were harder to get a hold off but easier to set up.

During the time we had scheduled for testing we could not learn how to use all the features of the applications. For instance, the manual of the freeware applications are all close to 200 pages and reading those would not fit within the time limit.

We were not able to test any of the CM tools with AutoCAD since we did not get a hold of the software and also because we reasoned that the focus should be on PLC and HMI software.

6 Configuration Management- and Software Development Tools

In this chapter the nine CM tools which have been tested, will be described. Four CM tools are freeware and five are commercial CM tools.

The software development tools used by BU DBF that the CM tools shall support will also be described.

6.1 CM tools

The CM tools that will be evaluated are:

- Subversion/TortoiseSVN
- Subversion/RapidSVN
- Mercurial/TortoiseHG
- Bazaar/TortoiseBazaar
- Proficy Change Management
- VersionWorks
- FactoryTalk AssetCenter
- Microsoft Office SharePoint Server 2007
- AutoSave

6.1.1 Subversion / TortoiseSVN

TortoiseSVN is one of the most widely used GUIs for Subversion today. It is implemented as a Windows shell extension which makes it completely integrated with the explorer. Even though the application is developed with focus on handling pure text documents, it is not an integration for any specific IDE. This makes it possible to use TortoiseSVN with any development tool. TortoiseSVN is developed under GPL which makes the program as well as the source code completely free. This means that you can use it to create commercial applications or just locally within a company, without any restrictions.[10]

The TortoiseSVN development team consists of three people and the project received the 2007 Community Choice Award from SourceForge as the best Tool or Utility for Developers. [25]

Information	
Maintainer	The TortoiseSVN team
Price/Cost	Freeware
Repository model	Centralized
Source Code	C++
Type	Desktop integrated client
Operating Systems	32-bit MS Windows (NT/2000/XP),Vista, Win2K, WinXP
Server	Microsoft Windows Server 2003
Tested Version	1.5.5 (Released 24. October 2008)
Supported Manufactures	All (disregards filetype)

6.1.2 Subversion / RapidSVN

RapidSVN is an open-source freeware GUI for Subversion and it is developed under the GNU GPL. The big difference between RapidSVN and other GUIs is that it is a stand-alone tool, which has its benefits and drawbacks. The GUI build-up resembles and feels like any typical windows program. The application uses add-on software to merge, to show differences and to edit files. [11]

The development team for RapidSVN consists of two persons. This can be considered a small number but since it is an open-source program. There are numerous plug-in software available with new features. [12]

Information	
Maintainer	Tigris.org
Price/Cost	Freeware
Repository model	Centralized
Source Code	C++
Type	Stand-Alone Client
Operating System	All 32-bit MS Windows (95/98/NT/2000/XP), All POSIX (Linux/BSD/UNIX-like OSes), Linux
Server	N/A
Tested Version	0.9.6 (Released 3 March 2008)
Supported Manufactures	All (disregards filetype)

6.1.3 Mercurial/TortoiseHG

Mercurial is a freeware application for distributed version control of files. It uses the command prompt but one can also use the TortoiseHG GUI. The TortoiseHG GUI is totally integrated with MS Windows, as it is a shell extension. As the name suggest TortoiseHG is derived from TortoiseSVN and consequently looks similar to it and has many of the same features. [13]

Information	
Maintainer	The TortoiseHg team
Price/Cost	Freeware
Repository model	Decentralized
Source Code	Python
Type	Desktop-Integrated client
Operating System	Windows XP
Server	N/A
Tested Version	0.6 (Released 18 January 2009)
Supported Manufactures	All (disregards filetype)

6.1.4 Bazaar/TortoiseBazaar

Bazaar is a distributed version control application that uses the command prompt for executing the different commands used with the application. Since this is not very user-friendly a GUI was used.

There are many different GUIs for Bazaar. Using one of these is an easier way to work with the tool. One of the most common and familiar GUIs is TortoiseBazaar. This is a GUI for Bazaar derived from TortoiseSVN. Most of the functions found are similar to those in TortoiseSVN. TortoiseBazaar is a Windows shell extension and integrated with the explorer. [14]

Information	
Maintainer	Canonical Ltd.
Price/Cost	Freeware
Repository model	Decentralized
Source Code	Python
Type	Desktop-Integrated client
Operating System	GNU/Linux, UNIX, Windows and OS X
Server	N/A
Tested Version	1.8 (Released 16 October 2008)
Supported Manufactures	All (disregards filetype)

6.1.5 Proficy Change Mangement

Proficy Change Mangement (PCM) is a CM tool developed by GE FANUC. It is built on Microsoft Visual Source Safe but requires only one license of Source Safe since it is only installed on the server. Proficy Change Mangement uses concurrent licenses.

The key-features of the tool are version control, security, audit-trail, scheduled backups and compares, electronic signatures and configurable systems.

A big difference between PCM and other CM tools is that it utilizes the built-in compare functionality in the development tools when comparing different revisions of a file. [15]

Information	
Maintainer	GE FANUC
Price/Cost	See Appendix XX
Repository model	Centralized
Source Code	N/A
Type	Stand-Alone Client
Operating System	Windows NT/200/XP, Internet Explorer 5.5 or higher
Server	Windows NT/2000/XP/2003, TCP/IP Networking, IIS if using HTML interface
Tested Version	5.80 Build 4541
Supported Manufactures	ABB, AutoCAD, Cimplicity HMI, Concept, GE Fanuc CNC, Intellution iFIX, LM 9030/9070, MS Excel/Word, ProWORX, RSLogix 5/500/5000, Siemens S5/S7, Wonderware etc.

6.1.6 VersionWorks

VersionWorks is a CM tool developed by GepaSoft, a German company situated in Karlsruhe. According to their homepage their annual turnover growth is clearly continuously in the two digit range and they are market leaders in Germany.

The company was recently bought by Rockwell who already has a similar tool called Asset Centre. These two products will be integrated and released under a new name within the next two or three years.

The software has all the specifications to handle CM for PLC and HMI and a unique feature with VersionWorks is the use of checksum when checking out/in files from the server. This guarantees that a file will never be corrupt. Another benefit is the possibility to make comments on specific changes within a file, not just the on the whole file.

VersionWorks also uses a “smart compare” function to visually display changes between different revisions of a PLC or HMI program.[16]

Information	
Maintainer	Gepa / Rockwell
Price/Cost	No information has been given. Seems to depend on wanted features and supported components.
Repository model	Centralized
Source Code	N/A
Type	Stand-Alone Client
Operating System	Windows XP Recommended hardware: Pentium 2,5 GHz+, 1 GB+ RAM
Server	Windows 2000 Server/Professional (SP 4 or later), XP Professional or Windows Server 2003 (SP 1 or later)
Tested Version	4.60
Supported Manufactures	SIMATIC, Rockwell, Schneider, Codesys, TwinCat, PCS 7, WinCC, InTouch, Citect, Kuka, All PC-based data types (Word, Excel, etc.)

6.1.7 FactoryTalk AssetCentre

This tool is developed by Rockwell Automation. AssetCentre has many different features besides version control. Some of the features are secure access to the control system, security and protection of files and processes, audit-trail, version control, checking which version that runs on a specific device and finding the latest version of your program, track users' actions and configure process instruments.

Today the application lacks support for Siemens but this will be fixed in a couple of years when AssetCentre and VersionWorks are integrated.[17]

Information	
Maintainer	Rockwell
Price/Cost	N/A
Repository model	Centralized
Source Code	N/A
Type	Stand-Alone Client
Operating System	Windows XP, 1Ghz+ Processor, 1GB RAM, 1 GB HD
Server	Microsoft® Windows® 2003 Server OS (w/SP1) *or* Microsoft Windows 2003 Server OS (w/R2) *or* Microsoft Windows 2000 Server OS (w/SP4) Microsoft SQL 2005 server
Latest Version	N/A
Supported Manufactures	Rockwell, Intouch, Fanuc, Motoman

6.1.8 Microsoft Office SharePoint Server 2007

This is a multi usage web-based application with many different features. The tool helps people within companies to share information in numerous ways. SharePoint can be seen as a webpage or portal builder but it offers much more than that. SharePoint is used to share files and documents, creating wiki's, using forums or sending e-mails. When it comes to sharing files the application includes a version control feature that allows files to be checked in and out, thus restricting and limiting the accesses for the files. It also has a notification option that send e-mails to users when a changes are committed.[18]

Information	
Maintainer	Microsoft
Price/Cost	Depends on version. Price list can be found on http://office.microsoft.com/en-us/sharepointserver/FX102176831033.aspx and the price varies between $4424 and $57,670 for the Server application and between $54 and $187 for the client access licenses
Repository model	Centralized
Source Code	N/A
Type	Web client
Operating System	For Windows systems: Microsoft Internet Explorer 6.0 or later; Windows Internet Explorer 7.0 or later, Firefox 1.5 or later; Mozilla 1.7 or later; Netscape Navigator 8.1 or later For UNIX/Linux systems: Firefox 1.5 or later; Netscape Navigator 7.2 or later For Macintosh systems: Firefox 1.5 or later; Safari 2.0 or later
Server	Single Server Deployment: A computer with processor clock speeds of 2.5 gigahertz (GHz) or higher; a minimum of 1 gigabyte (GB) of RAM, 2 GB of RAM recommended; disk space up to 2 GB for installation minimum, 5 GB or more minimum for data Server Farm Deployment: Computers with processor clock speeds of 2.5 gigahertz (GHz) or higher; a minimum of 1 gigabyte (GB) of RAM, 2 GB of RAM recommended; SQL Server 2000 SP3 (or later) or SQL 2005 system with dual processors of 2.5 GHz and 2 GB RAM minimum; disk space up to 2 GB for installation minimum, 5 GB or more minimum for data.
Tested Version	SharePoint Server 2007
Supported Manufactures	Mainly Microsoft Office files but also other common types of files

6.1.9 AutoSave

AutoSave is developed by MDT Software and it is a pure version control application. The tool allows management of many different types of files and documents. The management consists of protection, save, restore, discover and track of changes in regard to programmable industrial devices. AutoSave also has comparability features for different versions of a file by using the built-in compare functionality in the development tools.[19]

AutoSave is used widely within the automotive industry, especially in the USA. In this industry a plant often consists of hundreds of PLCs and the need to control each one of these and be able to tell which versions are running on which PLC is necessary. This is the reason that the price of AutoSave does not depend on the number of users but on the number of supported modules within the application.[27]

Information	
Maintainer	MDT Software
Price/Cost	Depending on number of modules
Repository model	Centralized
Source Code	N/A
Type	Stand-Alone Client
Operating System	Windows
Server	SQL Server, Windows 2003
Tested Version	5.04.01.05
Supported Manufactures	ABB, Bosch Rexroth/Indramat, Citect, Document Support, GE/ GE Fanuc, Gidding and Lewis, Kuka, MDT Software, Mitsubishi, Omron, Rockwell Automation/Rockwell Software/Allen-Bradley, Schneider Electric/Telemecanique, Siemens, Wonderware, Toyoda

6.2 Development Tools

The software development tools used at BU DBF and that needs to be supported by the CM tool are:

- Rockwell RSLogix 5000
- Siemens Simatic S7
- Beijer Electronics E-Designer
- Wonderware Intouch
- AutoCAD

6.2.1 Rockwell RSLogix 5000

Logix is used for creating software applications for Rockwell PLCs. It supports relay ladder, structured text, function block diagram and sequential function chart editors for developing application programs. [20]

Information	
Maintainer	Rockwell
File type	.ACD
Tested Version	16.03.00 (CPR 9)

6.2.2 Siemens Simatic Step7

Step 7 is the basic tool for creating PLC programs for SIMATIC S7, SIMATIC C7 and SIMATIC WinAC automation systems. Step 7 organizes all the programs written by the user and the data required for those programs in blocks. Blocks can for example be alarm-driven, time-driven and called or interrupted by other blocks. This facilitates the maintenance and the organizational clarity of the PLC programs. [21]

Information	
Maintainer	Siemens
File type	.zip
Tested Version	5.4 + SP2 (Revision Level: K5.4.2.0)

6.2.3 Beijer Electronics E-Designer

E-Designer is a graphical development tool for programming HMI software. The tool consists of many predefined block-libraries but new functions can easily be created. Personalized catalogues can then be built with different blocks and the symbol handler allows regular pictures to be imported to projects.

When programming with E-Designer, the blocks are simply dragged and dropped from the libraries onto the workspace. E-designer consists of a simulation environment that allows the user to test the program without having to download it into the actual hardware. [22]

Information	
Maintainer	Beijer Electronics
File type	.mpa
Tested Version	7.30 (Build 272)

6.2.4 Wonderware – Intouch

Intouch software is a tool for visualization and supervision of HMI's. The tool includes a multi-user development and editing environment. It also has a predefined symbol library that uses ArchestrA graphical symbols with integrated script and animation capabilities. These features are however not limited to AcrhestrA symbols. The script and animation features also support user-defined symbols. The software has the capability to directly gather information from I/O sources and display it in real-time [23]

Information	
Maintainer	Wonderware
File type	.zip
Tested Version	9.5.001 1101.0110.0018.0002

6.2.5 AutoCAD

AutoCAD is a computer-aided design program for development of blueprints. It comes with many predefined building-blocks for doing a complete working blueprint. It also allows the user to design 3D-model blueprints with features like layer-building. The 3D-model can then be viewed and explored. [24]

Information	
Maintainer	Autodesk
File type	.dwg
Tested Version	Not tested, see *Chapter 5.3.3 Software Testing Limitations*

7 Results and Discussion

This chapter will present and discuss the results from the survey, the interviews and the software testing.

The survey was handed out to approximately 45 Automation Responsible and Test Responsible employees within BU DBF. 22 surveys were returned and the result gave a hint about the knowledge level about CM within BU DBF.

Platform Managers and Automation Solution Responsible employees were interviewed. A total of 7 interviews were conducted and the result provided more qualitative information about the CM routines within BU DBF.

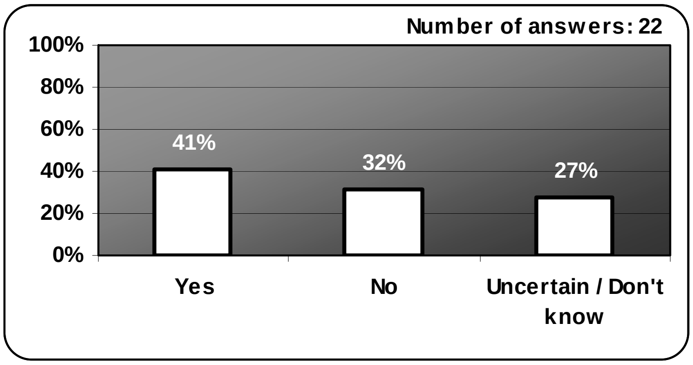
The software testing was performed on the tools described in *Chapter 6.1 CM tools*. The testing was the most time consuming part when building the empirical foundation and it involved both freeware and commercialized CM tools.

7.1 Survey

1. **What is your title? How long have you worked here? Any previous experience from this kind of work?**

The average automation engineer has worked at Tetra Pak for approximately 10 years. This should make the survey results reliable since the engineers should be well aware of the work process today.

2. **Do you think the flowcharts are representative of how you work?**



Comment

The majority thinks that the flowcharts are representative. The people that did not agree had some suggestions for correction. These suggestions are however based on how they work today and necessarily not the "correct" way to work. This can of course be due to the fact that the theoretical guidelines differ so much from reality that they are impossible to follow.

Remarks on the ownership chart were:

- The ARP should be replaced with ART (Automation Responsible Template) or at least have a connection between the ARP and ART in the cases where there are an ARP.
- Market Companies and Customers affect Customer-specific machines. There could be a direct connection between those boxes.
- Add ART and connect to ASR and TPS.

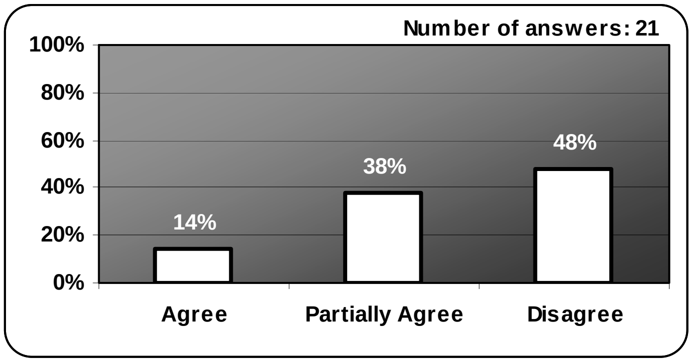
Remarks on the RFC chart are:

- There are no routines or it is very uncertain how the "Feedback and Error reported to responsible" between the AR in OFCE and the ASR in Portfolio works.
- The Programming and the Desktop test in OFCE are most of the time done by the same AR.
- The "Error sent for correction" in OFCE is really an active action committed by the AR. Nothing is sent.
- The "Test/Validation"-box in Portfolio is diffuse.

Conclusion

Most of the employees agree with the flowcharts, at least theoretically. There are some minor changes like spell correction that has been made and the ARP has been replaced with ART. To make any greater changes to the charts, it is probably best to present them on an AR meeting for a deeper discussion.

3. **I am familiar with CM (except from the short description on the first page).**



Comment

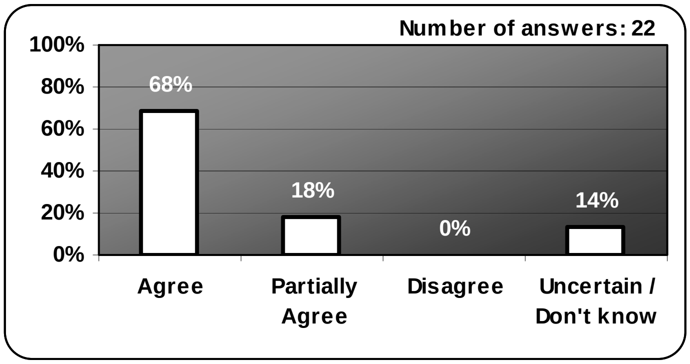
This result was pretty much expected. Most engineers use some sort of CM in their work. They document changes and store different versions of their' projects but the term CM is not used. This is probably due to the lack of a CM tool.

Conclusion

To fully appreciate and understand why work should be carried out CM tool, the employees must be well acquainted with the area. Otherwise people will find ways to work around it or set the wrong expectations on what the CM system will accomplish.[1]

One comment on the survey stated that they have always worked with CM but never used the term CM when referring to it.

4. **I think that CM in general is important for our company.**



Comment

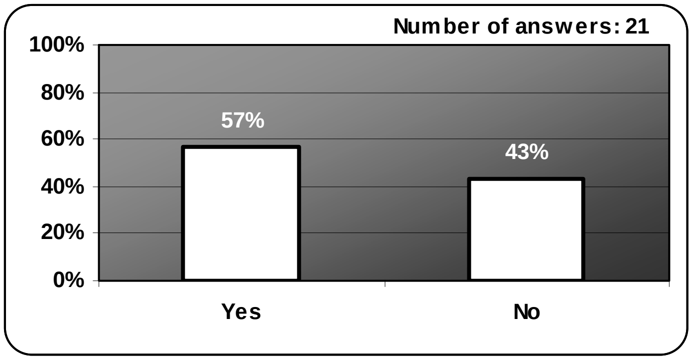
It is positive that the majority think CM is important. In fact, CM is very important and not having a tool produces a lot of manual work. Copying of files, pasting files, creating manual backups and so on is unnecessary work when there are tools that can automate this process completely, thus saving time and letting the developer do his actual task.

Conclusion

The fact that 68% feel that CM is important and that only 14% are familiar with it (see question 3) strongly suggests better CM routines and more education about the area.

More people than the ones using CM (see question 5) think CM is important. This can only mean that those people feel the need for a CM tool or better routines.

5. **I use CM in my work.**



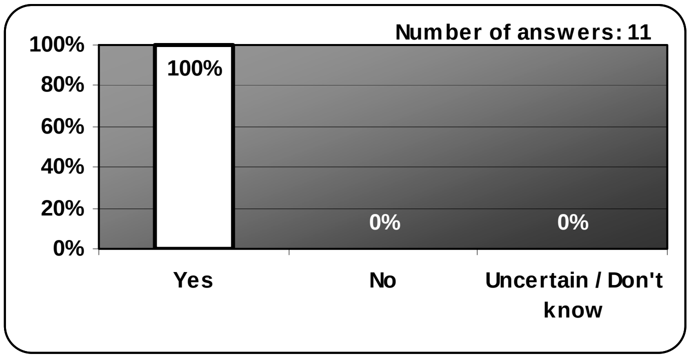
Comment

Only 57% claims to be using CM and it is not good enough. Ideally the result should be 100%.

Conclusion

These figures might be a little misguiding. We know for a fact that a lot more than 57% use CM, they just don’t know it since the term “CM” is rarely used. As soon as you store a project and label it with a version number you are doing basic version control. If you write a comment about the new version in a text file that goes along with the new version you keep an audit log of your work. Both are actions that a CM strategy applies and that a CM tool will facilitate.

If yes, do you think CM plays an important role in your work?



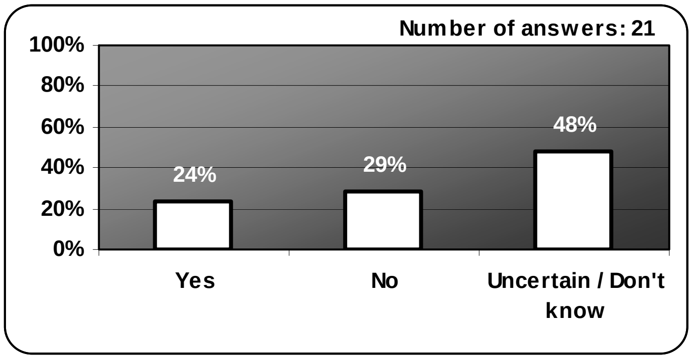
Comment

Clearly those who use some sort of CM think that it is important. This shows that by using a CM strategy you easily see the benefits with it.

Conclusion

Encouraging the people that do not think CM is important or that do not use it (see question 4 and 5) to work with CM would hopefully contribute to more yes-answers on this question. The result also shows that it might be hard to appreciate something you have never tried when looking at question 4. The benefits of working with CM should be pointed out before "forcing" anyone to use it.

6. Do you have any official CM-routines?



Comment

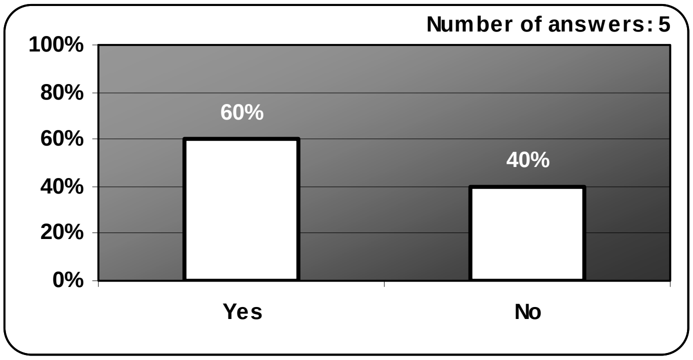
To work without any CM routines is not good. This means that employees store their projects and document their work in a fashion that best suits their own needs, instead of the common good of all.

Conclusion

When demand for faster releases and efficient use of resources increase, the need for good CM routines and disciplines are essential. This is not the same as adopting a CM tool. The principles of CM must be introduced to the developers on a project at the right time and in a way that make them see the benefits with CM. [3]

We know that there are some CM routines or at least official guidelines for how to manage files. Unfortunately it seems as many employees are not aware of this, but it might also be due to not knowing the meaning of what CM is.

If yes, do you work according to it?



Comment

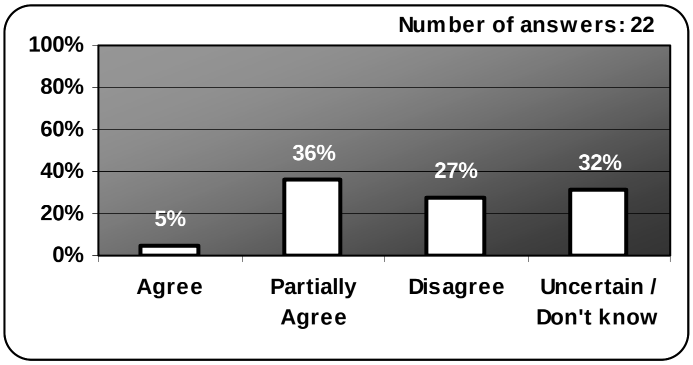
Only five answers might be too few to put any significance on the question. A comment from one of the employees that responded no is that there are so few members in his development team that their somewhat simplified routines work satisfyingly well.

Conclusion

There is really no reason to not follow the CM routines if they are working properly. If the routines are too complicated or in some way inaccurate, which makes them impossible to follow, they need to be reviewed and changed. The routines should never be constructed in a way that makes work easier if you avoid them, on the contrary the routines should make the work easier, more organized and unanimous throughout all departments.

If there are only a few members on the team, it is understandable if they do not feel the need to document or save files in a way that match the specified disciplines. But what happens if the team expands or if a team member retires or gets replaced? In a case like that, having common CM routines would make it easier for the new member(s) to get integrated with previous work.

7. **I feel that today's CM-routines are working properly within BU DBF.**



Comment

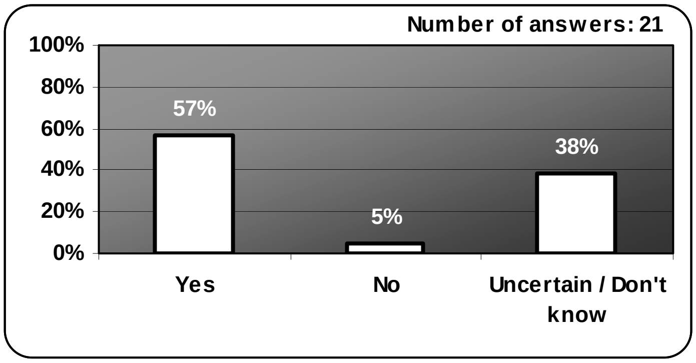
41% are more or less satisfied with the way the CM routines are working. This is inconsistent with question 6 where only 24% states that they have any CM routines.

32% are uncertain or do not know. This probably means that they have limited knowledge of the area, otherwise you would know.

Conclusion

One way to interpret this result is that there are some employees that are satisfied with not using CM. Another way to interpret this might be that some employees are satisfied with creating and following their own routines. These opinions would most certainly change if the right tool was implemented and the right CM disciplines were taught. Still the overall conclusion is that far too many people think that the routines are not working in a way that they feel is satisfying.

8. **I would like to see changes in the CM-routines within BU DBF.**



Comment

57% would like to see changes in the routines. The result was expected considering question 7.

Improvements people wanted to see were:

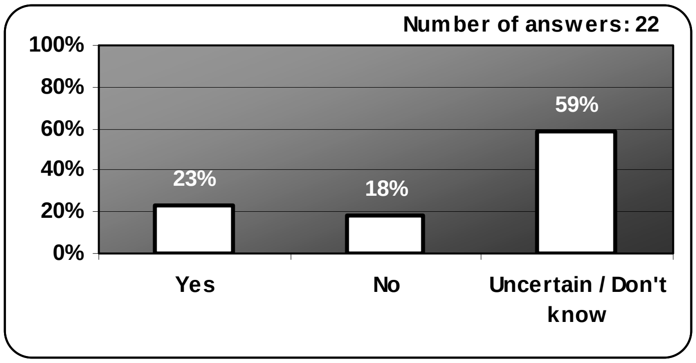
- More automatic handling of files
- Better way of handling variation orders
- Better communication between developers to speed up the process of updates and error correction
- Everybody should work in the same way
- Easier work process and more information and education

Conclusion

It is pretty obvious that people would prefer to use a tool that automates the whole process of managing files. One remark however was that this tool should not be too complicated to use since this would create a lot of administrative work. Someone also stated that there might actually not be anything wrong with the routines today. The problem might rather be how these routines are interpreted and carried out. This clearly indicates that there is insecurity about the routines and that more education is needed, either in how to implement the routines or about the subject in general.

Both of these statements are true. The tool should be user-friendly and simplify the work. The better a CM system works, the less you notice it [3]. It is also important to make sure that everyone knows what to expect from a CM system. The working routines associated with the tool should be easy to understand and consistent among all the users. Education within the area is essential; otherwise the users will not appreciate the benefits of CM and find ways to work around it.

9. Do you have anyone responsible for CM?



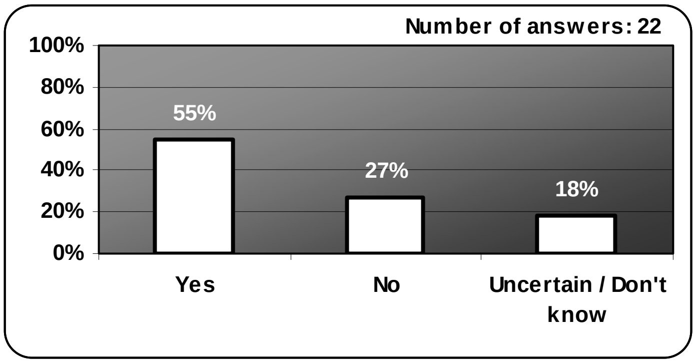
Comment

The 23% that claims to have a person responsible for CM refers to the person that have come up with the work routines on how to store projects, document changes, naming files etc. within their development team.

Conclusion

With a functional CM system up and running everybody is responsible for CM but it is a good idea to have someone with an overall responsibility to make sure that the correct CM routines are carried out [2]. Everybody should document their changes and follow the correct procedures.

10. Do you keep track of all variant releases?



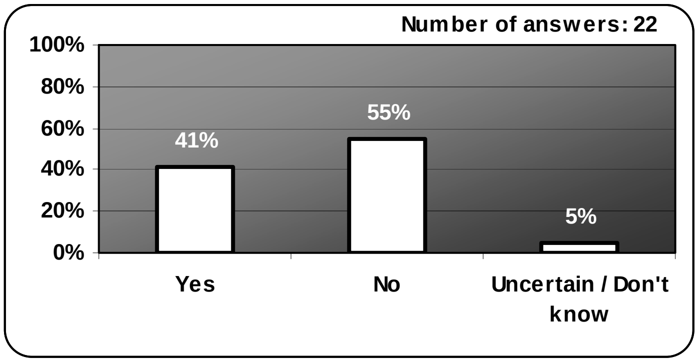
Comment

A variant release is a release that is essentially the same release as its predecessor but with some minor differences, see *Chapter 3.2.3 Variants*. Keeping track of these releases is good since a major bug or need of an update for the predecessor will most likely affect a variant release too.

Conclusion

Variant releases should always be tracked and this is something that a good CM tool will handle, see *Chapter 3.2.1 Storage*. This of course requires that the routines for storing variants are well thought through, making it easy to locate them.

Do you know which of the delivered machines that share the same code?



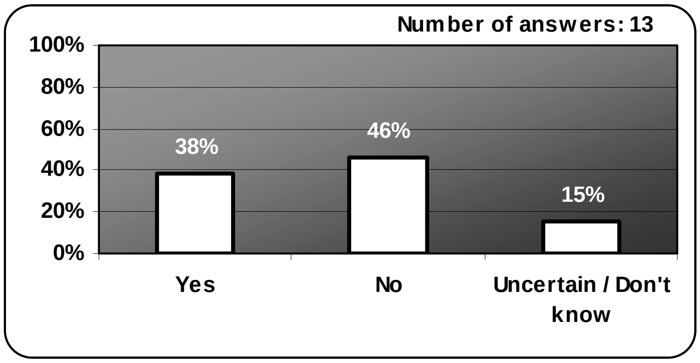
Comment

The distribution of answers should be more or less identical with the previous question since variant releases share the same foundational code.

Conclusion

This question may have caused some confusion in case you do not know what a variant release is and the definition of variants alter between different literatures. A variant is essentially the same release as its predecessor but with a minor difference, see *Chapter 3.2.3 Variants.* A variant machine can for instance be changed in such a small way as having a different colour or a display with a different language. The functionality of the variant is however the same.

If no, can you easily find out?



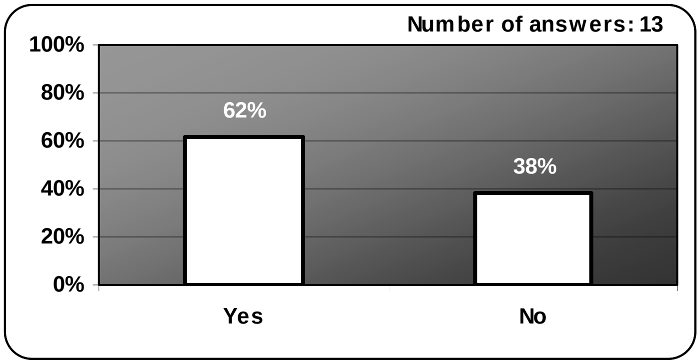
Comment

If you do not know which machines that share the same code and you do not know how to find out which do, you can face problems if the shared code is faulty. If shared code is in need of urgent updates or corrections, the process of finding out which machines that are affected must be easy, fast and efficient.

Conclusion

This is no problem with a good CM tool. It will track all the variants and shared assets. This can also be done without a tool if the routines are working well as long as the developers document their work in a way that makes it possible to see which templates their programs originate from. Doing this will make it possible to find the source of the problem but it will still be easier to use a tool since the process of finding the source will be reduced to a simple search in the history log.

11. **I can handle the speed and volume with which RFC (Request For Change), feedback and improvements to my programs occurs.**



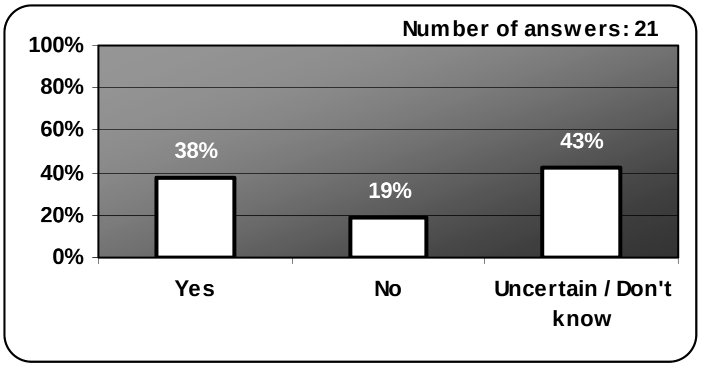
Comment

Only 13 people answered this question. The rest left it blank or said that they did not know what RFC stands for. RFC is Request For Change which means exactly what it says. It is a widely used term in programming environments and an abbreviation we definitely thought people were familiar with.

Conclusion

Even though there were only 13 answers it is not good that 38% claims they cannot handle the workload. You can never say for sure that a CM tool would change this situation but it might be able to help. The reason to why people cannot handle the workload should be investigated further. If it is due to time delays caused by working with the wrong code or the wrong setup of customer specifications for instance, a CM tool would most definitely help.

12. **I think that having specified release deadlines would be of help in my day to day work.**



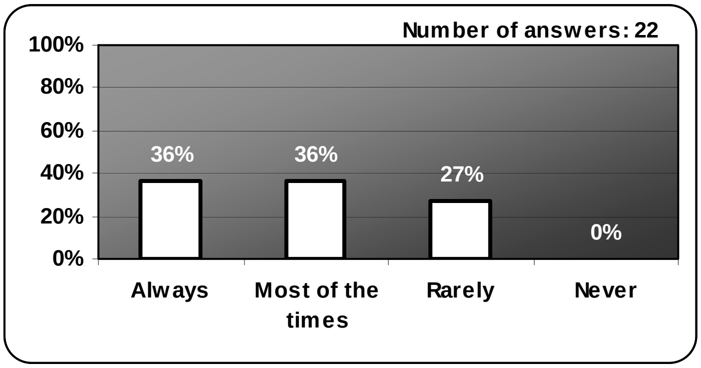
Comment

This is a question that was brought up during the AR meeting in November. The opinions during the meeting seemed to be evenly distributed and the result from the survey verifies this.

Conclusion

There can be both advantages and disadvantages with release deadlines. Some prefer deadlines to be able to better plan their work, others only see it as a stress factor. Having release deadlines is completely up to the management of the company.

13. I document all changes to my programs.



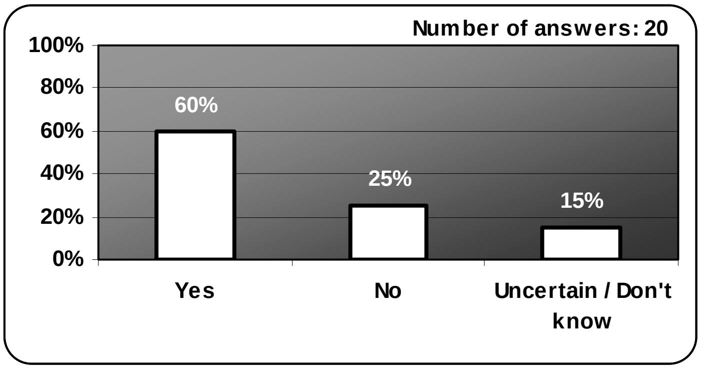
Comment

It is positive that no one replied Never but it is not good that 27% claims to Rarely document changes.

Conclusion

Documenting changes and keeping a log is a fundamental part of CM and should be carried out for almost all changes. Most changes made respond to a Platform Requirement Specification, an RFC or fill some other functionality. When saving the day’s work it should be implied that the developer makes a few notes in the log about what changes have been made and in response to what PRS or RFC. In a fully functional CM environment all RFC's should be labelled with a version number of their own. This number should be included in the log to make the traceability easier [1]. In most CM tools there is the option of making it mandatory to fill in a log message after making changes to files.

Is it easy for other people to read and understand the changes just by using this documentation?



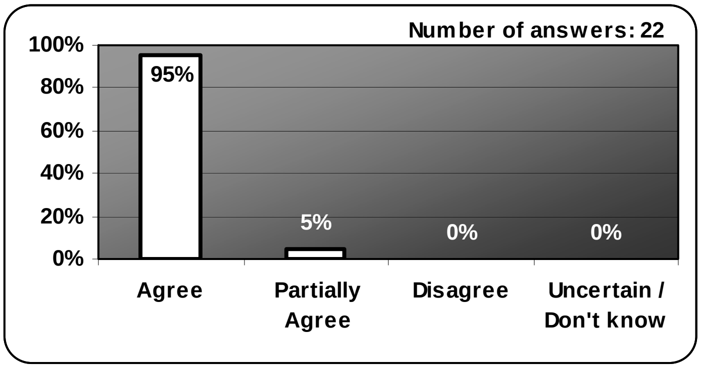
Comment

It should be easy for everyone to be able to understand changes made just by reading the log. There is no point in keeping a log if no one understands its content. A good log should consist of the parts we mentioned in *Chapter 3.2.4 History Log*. If you do not use a CM tool the log should be easy to locate and preferably stored in conjunction with the file it refers to.

Conclusion

Only 60% say that the log is easy to understand. Not all departments have clear rules about how the log should be written. The biggest reason for the log not being easy to understand is probably due to the fact that some developers only keep a log to organize their own work. This means that the log was never created with the intention of anyone else reading it.

14. I store my programs in an organized way.



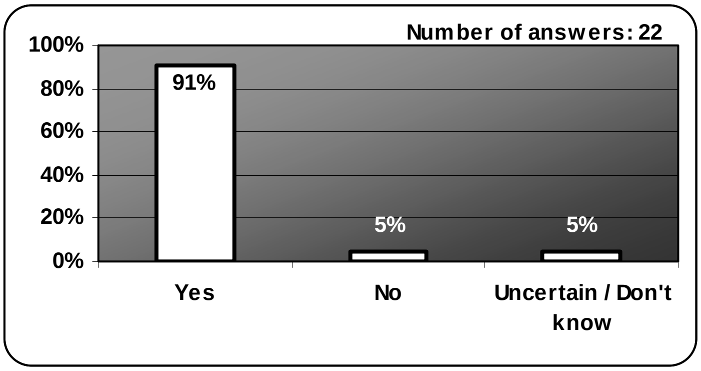
Comment

This was a very positive outcome and it could not be much better.

Conclusion

It is self-explanatory that you have to be organized. Not only for your own sake but for your co-workers too, in case they would like to have a look at your projects.

It’s easy for other people to find these programs.



Comment

The programs are stored on a local server which makes them accessible for everyone. This alone however does not mean that the programs are easy to find, a common tree structure for how projects should be stored would facilitate finding sought after versions of various projects.

Conclusion

Today the server is being backed up every night. Mistakes such as accidental deletion of files or user slip ups in relation to drag and drop of files and folders can still mess up the file structure of a project. Things like these are hard to avoid but can be controlled more efficiently if the read and write permissions to each project are configured properly.

For example, an unauthorized user may want to have a look at a program from another department and should be able to copy the program to his local computer. If he chooses to do this by drag and drop and accidently places the folder in the wrong directory, this should not be any problem if he does not have write permission. Some CM tools also prevents actions such as drag and drop.

How do you name your files/folders? ***(for example: filename_ver_XXX.XXX_XX or similar)***

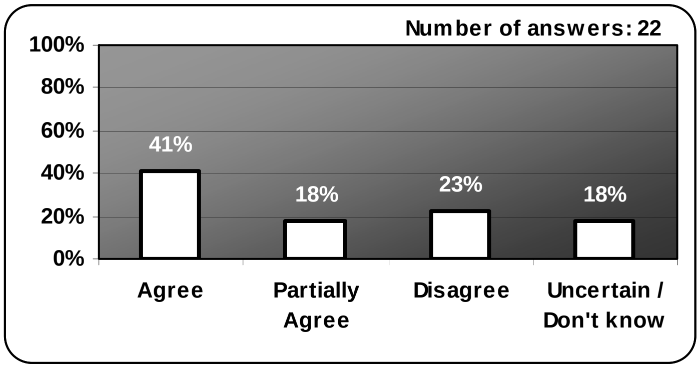
Comment

The naming of files seems to differ between different departments. Most template names today includes a version number but still puts a lot of responsibility on the user since the naming is done completely manually. Some answers also stated that there is no standardized way to name files.

Conclusion

The file name should be made from the same foundation for all projects, a standard should exist [2]. This will make traceability easier, avoid unnecessary confusion and give a better overview of the projects. The file name do not necessarily needs to contain the version number or date. Most CM tools lets you label or tag your file with a version number.

15. I know who to contact or which documents to look at to find out how long it typically takes to make certain type of changes in my programs.



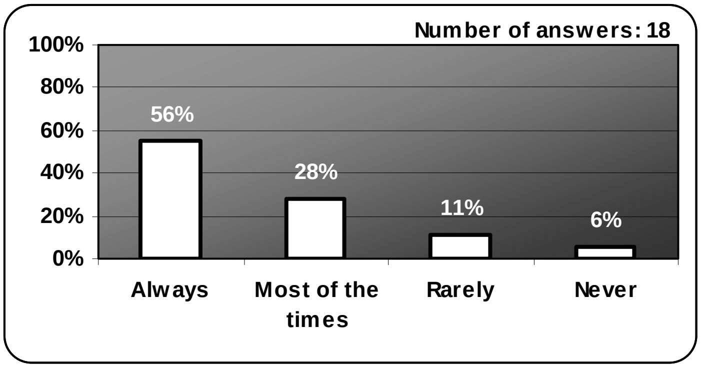
Comment

Most developers seem to have some clue about where to look or who to contact when facing different programming problems.

Conclusion

If the entries in the log tied to each project are made correctly this could be of great help when trying to figure out how long it will take to make a certain type of change. By searching for similar problems in the log, the developer can with some luck find the same fix in another project and reuse that code or at least find something similar. If he for some reason do not understand the changes or have other questions, the responsible programmer will also be listed.

16. **I am reusing bug fixes across releases rather than wasting time fixing the same bug that was fixed in a previous release.**



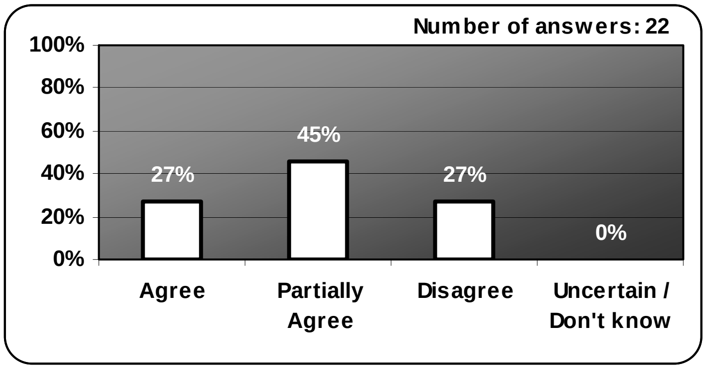
Comment

This question is tied to the previous one. You cannot reuse bug fixes unless you know where to find them or who to contact. It is positive that the majority are reusing bug fixes and there is no reason why you should not.

Conclusion

Once again the audit log will play an important role. By connecting RFC with bug fixes in the program and documenting these in a predefined way it will be easier to find fixes for the bug the developer is facing. Even if a developer has made an identical bug fix in another project, the process of finding it will be faster if a search in the log is all it takes. In addition, all bug fixes should also be given a version number and be tracked. [1]

17. **I know for sure that everything that is released has gone through the necessary testing.**



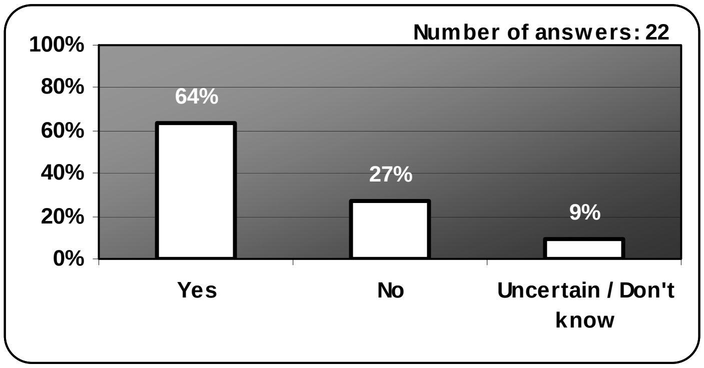
Comment

It should be enough that the one writing the program make sure that it fulfils the product specification or solves an RFC. Still only 27% fully agreed with the statement.

Conclusion

This question falls under the Quality Control and Quality Assurance area of the CM model. Releasing a version without knowing if it is good enough will probably generate RFCs that could have been avoided if the necessary testing had been made. Formulating clear steps on how to test a program and trying to avoid misunderstandings by applying effective Quality Assurance would cut down the workload, see *Chapter 3.5 Quality Assurance and Quality Control..*

18. **I can roll back a program to an earlier version to recover from a bug in a short amount of time.**



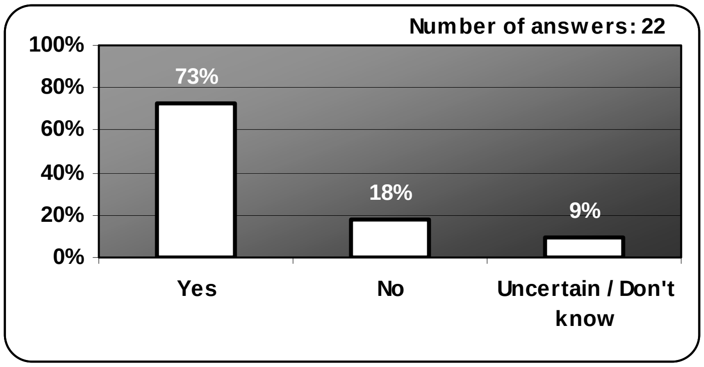
Comment

Being able to roll back a program could turn out to be important in case of an emergency of some sort.

Conclusion

Roll-back is something that all CM tools support. It simply means that you revert your program to an earlier version and this is important in case of severe bugs. Today it is possible to roll back a program to an earlier version since these are stored on the server. It is also possible to revert to a version that was between releases by calling the support and ask them to recover a backup from the server. A CM tool simply lets you choose a version in the audit log and make that version the latest if you want to. This is a nice feature when you have to recover from a bug and the latest version is not always the correct version.

19. I know which developer is editing code or content and why they are doing it.



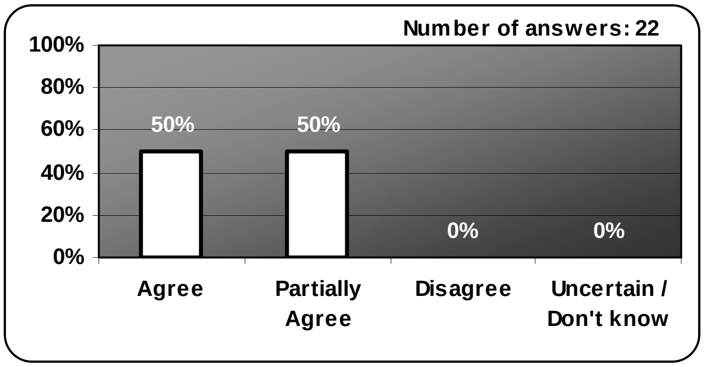
Comment

This seems to be functioning relatively well considering that 73% answered yes. This is probably due to the fact that the development teams are fairly small and most of the communication is done orally.

Conclusion

If the development teams are big and if someone is waiting for another member of the team to finish a change before he can do his part this process might not work as smooth. Most CM tools offer the possibility to write a check out message. This message can for instance contain info about why the file has been checked out and approximately how long it will take to finish the programming. In the case that someone has forgotten to do a check-in, the audit log lets you know who has it checked out so that you can contact the person in question.

20. I think that I work in a structured and organized way.



Comment

It is true that everyone work in a structured way. Every team has their own procedures that they follow. The work might be structured and organized, but different in every department.

Conclusion

A CM tool organizes your work and makes little room for individuals to make their own structure depending on the configuration of the tool. Introducing a CM tool and implementing correct CM procedures would ensure unanimous work throughout all departments, which is one of the requests from the employees, see question 22.

21. **Which is the most common problem you encounter in your work with regard to CM? Is it in the process, the CM tool, the program, trouble finding files, etc?**

Comment

Some of the answers were:

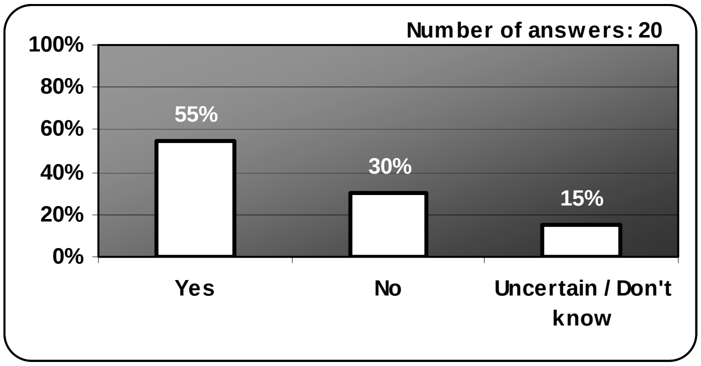
- Overkill of rules and rule changes
- Hard to find a software bug in the logic because the logic is very complex for certain process modules
- Some units have no templates and the developer are therefore forced to copy old projects with special end customer adaptations needed
- There is no CM tool
- Routines are not followed. This might be due to lacking information, no follow-up and shortage of time
- A lot of manual work and this takes time
- It is a problem to follow procedure in a variation order
- No knowledge about CM
- Getting the changes made verified on site in a safe manner
- Working different between different machines and categories is an obstacle
- Finding specific files can be quite difficult
- Tracking changes

Conclusion

Most of the above mentioned problems could be solved with a good CM tool. The ones that cannot be solved by a tool is the lack of knowledge about CM and not following routines.

The user must have knowledge about CM to appreciate it and this can only be achieved through education. Not following routines when using a tool will cause problems but it will be easier to set up more structured routines when integrating a tool in the development.

22. Are you missing anything in your daily work when it comes to keeping track of versions and changes, looking in between products and categories?



Comment

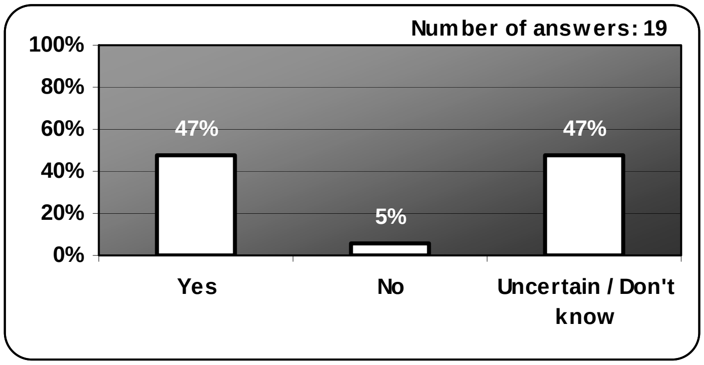
The majority are missing something in their work and some of the requests were:

- History of special versions
- Summaries
- A CM tool
- Working with the same structure in all the categories
- Using the same version control in all departments
- An easy way to find orders with a specific software version
- Better exchange of information and know-how
- Bug fixes are not shared in a efficient way between products and categories
- Better descriptions of changes between versions

Conclusion

Just as the previous question a CM tool will fix most of these requests. The audit log will play an important role and better descriptions of changes between versions will only be achieved if the developer documents the changes properly. A CM tool will provide a better structure of projects but if the proper documentation is not done, it will still be hard to find what you are looking for.

23. Are there any particular features you would like to have in a CM tool?



Comment

47% have some feature they would like to see. The 47% that are uncertain at least do not exclude the possibility of introducing new features.

Some of the requests were:

- Possibility to search for changes
- A good editor which makes it easy to add comments about the changes that were made
- Possibility to store or link describing documents in the most common formats (doc, xls, gif, jpg, pdf etc.)
- Possibility to tag or label a file
- Built-in function for comparing code
- A log for registration and description of bugs with built-in trigger for sending feedback to the template responsible
- Possibility to version control complete special programs, for instance customer specific ones, not only templates
- See changes between versions
- Better feedback tool
- A way to make sure that the one who need a change and the one making the change can be absolute sure that the change is verified in a safe way
- User friendly CM tool

Conclusion

These requests can hopefully be fulfilled with the use of a good CM tool.

7.2 Interviews

The interviews gave some more elaborate answers to the questions we thought were important but after only a few interviews we realized that we probably would not get anymore input than we already had. The answers were more or less the same to all questions and most of the responses were expected. Compared with the survey, the interviews were conducted with the hope of getting more detailed information and for some interview subjects this was the case. Unfortunately the overall knowledge about CM within the company is so limited that only a few interviews provided us with any elaborations at all.

The interesting findings from the interviews are presented below.

General work routines

- There are no rules or requirements for how to document the code changes done in the programs. The least you could demand are some guidelines for how to document changes and in what editor you should do it. Everyone do it in their own way today but everybody would benefit if the rules were clearer and unanimous through all departments. Even though each department follow a certain layout when documenting changes and everyone knows how to do it in similar way, there are no official guidelines that are documented anywhere.

- To document changes in programs, some use a combination of MS Word documents, MS Excel Sheets and even take screen shots to graphically show differences. Some store this documentation in the same folder as the project files. Others create a folder named “Changes done from previous version” or similar and store the documentation there.

- Old projects and ongoing projects are stored on a server that everybody has access to and no passwords are required. Releases are stored in the same location.

- Testers, up-starters and customers would have much easier to follow documentation if it was done in a structured way.

- Even though templates are used, the process of tracing specific releases can be hard since most machines are constructed with many customer demands.

- The work approach with the usage of templates is relatively new and there are no clear rules for Quality Control.

- There is no easy way to find out who works on which files, except for those that work in small teams, since there are no check-in and check-out system and no possibility to lock files.

- Roll-back is possible but it requires that you call the support and ask them to recreate files from the backup server. This must however be done in a fairly short amount of time from the day that the error that caused the need for rollback was detected. This is due to the limited time that the backup is stored before it gets deleted. Once roll-back is done, it can still be complicated to proceed. It requires that you manually locate the previous version and make all the code changes except for the ones that created the need to do the roll-back in the first place.

- Conisio is a version-handling tool that was in use a short while but disposed of since it crashed when handling RSLogix files. Another problem encountered when trying the application was the lack of knowledge of such a system among the developers. They did not understand the purpose of checking in and out files and apparently work was lost since people did not check in the files believing that they were backed up on the server automatically. To implement a tool with a check-in/check-out feature it would require education.

- OFCE seems to be the only department that keep a consistent layout of their file structure.

Quality Assurance and Quality Control

- To perform Quality Control and Quality Assurance a tool called Mercury Quality Center is used within Plant Autmation and in some projects within BU DBF. This system of setting up test cases is apparently very well documented to ensure easy repeatability of the tests and overall it is working very well.

- The programs are tested in a desktop test, a workshop test and finally a field test. The field test is done at the customer site and most of the times long after the workshop test and long after the code have been released. This requires that you can go back and correct the bugs that are reported at the start up.

- The water tests in the workshop are not completely reliable due to several factors. One is of course the viscosity of water in relation to the fluid that the machine is designed for. Another is the fact that all processes are tested with only water. If you run a process and see that there is water coming out, how can you say for sure that it is not a left over from, for instance, the dish program that was tested earlier?

Wanted features in a CM tool and suggested improvements in the CM routines

- Higher security with the use of check-in and check-out of files.

- The tool should log who access which files and who has done what.

- Since everyone uses their own system for storing files it can be hard to find the logs belonging to some projects. The log should be easy to access.

- Possibility to lock files to ensure that no one else uses it while it is under development.

- The same work procedure in all departments.

- Possibility to create workflows that, for instance, automatically sends e-mails when a new version is released would be a nice feature.

- A CM tool should be able to make the roll-back easier and have the ability to compare files graphically.

- It is very important that the tool is easy to use and not to administrative. It should not slow down the day-to-day work just to gain a few advantages in something that are rarely used. Also, the licenses of the CM tool must be concurrent in case of computer failure. Moving a license from one mac address to another is a time consuming process that will cause lost of many work hours.

- In the OFCE folders you get information in a structured way. You can see the program before commissioning, after commissioning when some bug fixes and customer specific requests have been implemented and which actual programs that were delivered with the finalized order. But there is one drawback. There is no way to see which actual revision that was delivered. Nor is it possible to see which version of Tetra Plantmaster the revision is compatible with. By putting a release label on the folder with the delivered software it would be easy to open up your CM tool and see what revision it was. Then it would also be possible to see which improvements that have been done with the program since that specific revision. If for example the customer get in touch and requests upgrades on a specific functionality it would be easy to see if these requests already have been implemented in a later release.

- Instead of as today use a product key, which in reality is a MS Excel sheet with a lot of columns listing the different versions of software in a product you could simply use CM. All you would have to do is to make the changes needed and label the file with a revision stamp and you are done. The factor connecting the different components of the machine would then be the revision stamp, which is searchable within the CM tool, instead of the product key, i.e. a MS Excel sheet.

Other

- It might be out of the frames for our work but the Correction List in OFCE should maybe be disposed of and replaced with another system, preferably ABEO. This is to limit the number of tools and use the same system through the whole organization.

7.3 Software Testing

The software testing was performed against the test specifications in *Chapter 5.3.1 Software Testing Specifications* and the results for the different CM tools are presented in this chapter.

7.3.1 Test results of Subversion/TortoiseSVN

The security of TortoiseSVN can as in any other CM tool be customized. It is a little troublesome since there is no administrative tool and it is not as customizable as the commercial tools that have been tested. The creation of users, user groups and the passwords are done through .txt files that are located in the repository. This means that the process of adding users to projects cannot be done quickly. The administrator has to access the repository on the server and manually type them in to the text files. This approach also limits the flexibility of Tortoise. Many settings are just not there. It might for example be a good idea to assign the possibility to override checkouts to the administrator in case some forgets to check a file back in but this option is unfortunately not available. The only administrative settings that can be altered are user access and user passwords to different repositories.

TortoiseSVN is able to keep track of revisions for all the tested file types since it does not regard the file type. When working with binary files TortoiseSVN has some minor bugs like making Windows display dialog boxes with error messages, when in fact no error has occurred. This is nothing that will affect the files in any way but it is still very confusing and gives the impression of TortoiseSVN being a bit unstable.

The compare function works well for pure text files and even for MS Excel files and MS Paint images. When trying to compare PLC and HMI files an error that states that the files are binary are displayed. This was expected because TortoiseSVN uses incremental saves. This means that it only saves the changes between the different revisions, not the complete files.

The inability to compare binary files can be bypassed by proper use of the audit log. The log is good, easy to overview, it is searchable and it is possible to open any version of a file with the corresponding development tool. The search function is very useful since it lets the user search for anything from filename and users to check-in comments.

At first glance, TortoiseSVN looks very appealing. The complete integration in the Windows Explorer could not be better. It is easy to manage files and folders. The work procedure is as working without any CM tool at all. You create, copy and paste your files as usual. The only difference is that you right-click and commit the files to the repository. This use of the tool makes version control fast and easy. Unfortunately Tortoise is developed mainly to handle pure text files and for this purpose it is probably the most favourable tool available.

To use Tortoise on a daily basis when working with binary files is not an option. There are too many flaws and since there is no support for PLC files you encounter other problems, for instance when working with RSLogix 5000 files. When saving a file RSLogix automatically creates a backup which has to be manually deleted since it is a new file, unrecognized and useless, to TortoiseSVN. Furthermore when opening Rockwell files through the log you will encounter strange error messages when there in fact is no error at all.

Except from confusing error messages and minor bugs when performing actions on binary files, TortoiseSVN is quite good. The version control works flawless.

Using a freeware application tool like Tortoise will require the application to be updated frequently and from the day we started evaluating the tool till today, 3 new updates have been released. You will also be forced to keep the version of Subversion running on the server updated or else you will encounter compatibility issues when upgrading. This will without a doubt create a lot of administrative work.

Application	File type	Supported	Ability to show differences
Rockwell RSLogix 5000	.ACD	No	No
Siemens Simatic S7	.zip	No	No
Beijer E-Designer	.mpa	No	No
Wonderware Intouch	.zip	No	No
Microsoft Word	.doc	No	Yes (with add-on software)
Microsoft Excel	.xls	No	Yes (with add-on software)

7.3.2 Test results of Subversion/RapidSVN

RapidSVN did not have built-in support for any other types than pure text files. Rapid did manage, as expected, to keep all the tested file types under version control and the version managing worked just as well in RapidSVN as with for example TortoiseSVN.

A drawback with the application is the fact that it uses add-on softwares for the comparison and merging of files e.g. for MS documents. The add-on programs we tested were WinMerge and WinDiff. The add-on programs can be chosen freely by the user and RapidSVN can then be configured to use those programs as default when doing merges or compares of files. Configuration of the add-on programs is done through the menu of RapidSVN, but note that the programs must be installed on each computer that wants to use them. Using add-on programs means that you have to install and manage three applications instead of one which is a huge inconvenience. The situation might occur that a newer version of any of these three applications may lead to a compatibility issue and this is a major disadvantage for the tool. Also it has to be said that the usage of these add-on programs were somewhat difficult at first. The programs were not easy to use and it took some attempts before we got it right.

The audit log shows the wanted information about the version of the file e.g. comments, date and who has made the latest check-ins and check-outs. To use the audit log as a way of looking at the differences between versions of a file can be a good idea if you do not want to bother with the installation of extra programs. The information in the log is presented in a structured way and it is easy to get an overview of it.

RapidSVN looks like any MS Windows program. As mentioned in *Chapter 6.1.2 Subversion/RapidSVN* it is a stand-alone application, which means that the user has to start the application every time it shall be used, but MS Windows can be configured to automatically start RapidSVN every time the computer is turned on. The menus are easy to use and navigate through and the application as a whole is user-friendly.

An inconvenience that we found was that you cannot create a repository through the GUI, instead the command prompt and a Subversion command line had to be used. On the other hand once the repository was created the tree structure of the folders could be set up in any way that was wanted through the GUI. One observation that was made during the set up of the repositories was that they could not be created on a network driver but were instead required to be created on the server by typing the URL in the command prompt.

Since RapidSVN uses Subversion for setting up permission for users, the configurations has to be made, in the same way as for TortoiseSVN, in a text file located in the global repository.

In regard to the goal set up by the company RapidSVN does fulfil the version control aspect of managing files. It does however not fulfil the requirement of being able to compare differences between revisions for a file graphically. Using RapidSVN with any of the development tools was an effort and it is not recommended. This tool is better suited for working with pure text documents.

Application	File type	Supported	Ability to show differences
Rockwell RSLogix 5000	.ACD	No	No
Siemens Simatic S7	.zip	No	No
Beijer E-Designer	.mpa	No	No
Wonderware Intouch	.zip	No	No
Microsoft Word	.doc	No	Yes (Windiff)
Microsoft Excel	.xls	No	Yes (Windiff)

7.3.3 Test results of Mercurial/TortoiseHg

Mercurial with the TortoiseHg GUI is a very easy to use version control tool. If it is used only as a version control tool it works perfect. Creating repositories, adding files, cloning repositories and so on is done with no effort through the menu commands. The TortoiseHg GUI is in itself very simple and easy to get started with but the look of it feels somewhat dull. However this is just a question of aesthetics and has nothing to do with the performance of the application.

The fact that the application does not have support for showing differences by deafault, neither graphically or text based is a major set-back. The application does however allow third-party software to be used to merge and to show differences. As mentioned in *Chapter 7.3.2 Subversion/RapidSVN*, this might give rise to compatibility issues and it consequently might become a problem when using the add-on programs with the tool.

The audit log allows the user to see the change comments for each revision. Whenever a file is committed the user is prompted to make a change comment for the file and these are the comments that are shown in the audit log. In many cases this can be sufficient to understand what changes that were made and thus a compare function might not be needed.

Being a distributed tool means that locking and a restricting access to folder and files do not exist. As we mentioned in *Chapter 3.4.2 Distributed System* this is the way in which a distributed system works. The idea is that any user should be free to push or pull data to and from any repository. The distributed system can be set up to mimic a centralized system as we discussed in the same chapter but TortoiseHg still does not have the option of locking files.

Since the tool works in a distributed manner the situation might occur that a file has been altered at the same time by different users. When this occurs and a push is attempted to the “shared” repository, an error message is displayed. The message consists of a question asking the user if he/she has forgotten to do a merge. For text files the add-on programs can be used to merge the files but for binary files the merge has to be done manually. There is also the option of doing a forced push to the repository but this means that the data in the repository will be overwritten.

When looking at TortoiseHg from the company’s point of view of what the wanted application should be able to perform it is very obvious that this application does not meet the requirements. Another aspect that makes this application badly suited for Tetra Pak is that it works in a decentralized/distributed manner. Since all work within the company is done in a centralized way, TortoiseHg is not appropriate for the task at hand.

Application	**File type**	**Supported**	**Ability to show differences**
Rockwell RSLogix 5000	.ACD	No	No
Siemens Simatic S7	.zip	No	No
Beijer E-Designer	.mpa	No	No
Wonderware Intouch	.zip	No	No
Microsoft Word	.doc	No	Yes (with add-on software)
Microsoft Excel	.xls	No	Yes (with add-on software)

7.3.4 Test results of Bazaar/TortoiseBazaar

TortoiseBazaar works with any file when it comes to version handling. However, we found that it was generally more suited for working with pure text files and not the files that the development tools we have looked at generate. As a tool for version handling it is very easy to use and the menu context is very straight forward. Only executable commands are shown when working with the files and folders.

The Bazaar tool does not support the ability to compare and show differences of binary files. The only way to get a sense of the differences between different versions of a file is to view the user comments in the audit log. TortoiseBazaar does have a built-in diff feature but it only works with text files.

The audit log displays all necessary information about the revisions and at times more than what is actually wanted. A good feature that the audit log has is that it is searchable. The search of different key words can be done either by searching for change comments, authors, revision IDs or revision numbers. The search feature can come to good use when wanting to find a revision fast as you avoid searching the entire audit log list.

Bazaar is a distributed system and does not support restricting the access to folders and files. Like we mentioned before the distributed approach of working with files implies that all users have access to each other’s repositories and that the work is free to be shared in anyway wanted, as explained in *Chapter 3.4.2 Distributed System*.

Configuring tree structures and setting up folders is done with ease. It is completely up the user in what way the folders should be grouped. There is no predetermined tree structure when a new repository is created and since it is easy to add folder and files to a repository the whole process is effortless.

Being a distributed tool means that conflicts might take place when working with Bazaar. If for instance two users work on the same file and then try to commit the work to a shared repository, a conflict will occur and the application will complain. The way that TortoiseBazaars solves this is by creating two extra files and altering the name of the file you are working with. The files consist of the name and type of the file plus an extra suffix. The added suffix depends on where the files originate from. For .doc files the new names become “name.doc.BASE”, “name.doc.OTHER” and “name.doc.THIS”. The BASE is the file in the shared repository, THIS is the file in your repository and OTHER is the file that was committed by the other user. The user then has to open these files manually and make the necessary changes to the content and decide what should be kept to resolve the conflict. If it is .txt files the diff function can be used to see the differences.

As mentioned before the Bazaar tool works in a distributed way and that together with the fact that it does not fulfil the requirement of displaying differences graphically, which was a desire from the company, Bazaar does not fulfil the test specifications.

Application	File type	Supported	Ability to show differences
Rockwell RSLogix 5000	.ACD	No	No
Siemens Simatic S7	.zip	No	No
Beijer E-Designer	.mpa	No	No
Wonderware Intouch	.zip	No	No
Microsoft Word	.doc	No	No
Microsoft Excel	.xls	No	No

7.3.5 Test results of VersionWorks

We did not get a chance to test this application until the last weeks of our work. In the beginning of our work we had a Webex meeting with GepaSoft where we got an introduction of the application and a live demonstration. During the demonstration it seemed like a really good application but the price for a pilot install was approximately 5000 Euro since GepaSoft would have to send someone to our office to do the installation. The installation is apparently difficult and may take a while. On the other hand, an introduction course for the application would be given. The price tag for the pilot installation was too big but the contact with GepaSoft was kept and finally we agreed-on sending an external hard drive with a Windows Server 2003 image. A week later we got a copy of VersionWorks, free of charge. Only referring to the Webex meeting would have led us to the conclusion that VersionWorks would probably be the best suited tool for the task at hand but it always feels better to try it out by oneself.

When you start using the application you soon realize why GepaSoft offers an introduction course. The first time it is not that easy to understand and there are a lot of buttons and choices for the user, but once you get acquainted with the structure it is pretty straightforward.

The client application is called the Commander and when starting VersionWorks you get two choices of how to display it – Power mode or Easy mode. There is also a Classical mode, which reminds of the MS Explorer and it is a bit trickier to use. The preferred layout is the Power mode since it provides you with all information you can possibly want about a file, but if you just want to make a quick change to a file, the Easy mode is probably the best suited.

Graphically the application looks really good. The icons and buttons are similar to those in MS Windows and the navigation through your projects is done in the same manner as in the Windows Explorer. There are a lot of clickable items and choices for the user but the most essential information is clearly displayed and it is easy to get a good overview of it.

You can open the “Project Administration” from within, or outside of, the Commander. Here you will find more configuration options than in any of the other tested CM tools. Setting permissions, adding unsupported types of files, configuration of the server and mail etc. are done from here and everything seems to be self-explanatory or done with the help of a wizard.

Once you get acquainted with VersionWorks the application is very easy to use. As soon as you get a feel for the construction of the application it is not much more difficult than using the Windows Explorer to organize your work. You can set up the tree structure as you wish and drag and drop files freely. A slight difference from the Explorer is the three types of folders used in VersionWorks – regular folders, project folders and component folders. The regular folder is, as the name implies, a regular folder and these are used to organize your project folders. The project folders are used to organize your component folders. In other words, you can only add a component to a project and you can only add a project to a regular folder. The icon for the regular folder is a plain folder, the project folder has "prj" written on it and the icons for the component folders take the shape of the icon for the component that the folder is meant to contain. This gives a clear and good overview of your projects.

The "smartcompare" that Gepasoft demonstrated during the Webex meeting and spoke warmly of seemed like an interesting feature. Unlike some other CM tools, VersionWorks does not use the built-in compare functionality of the development tools. It is a unique feature and it turned out to work pretty well.

For RSLogix the compare worked flawless. It displayed the code changes graphically and also detected changes in the configuration of the Logix program.

For Siemens it did not work as well. It was able to detect which block that had been changed but it could not display the changes graphically. It was however able to display changes done to the hardware configuration, connections and symbols. This is something that not even the built-in compare in Simatic is able to do.

Differences in Intouch applications could not be displayed graphically but just as for Siemens, VersionWorks was able to detect which windows that had been altered and even display changes in the configuration of the application.

Microsoft Office was not installed on the Server image and this caused an incompatibility with MS Word and MS Excel documents. The compare for these applications shall however work if they are installed.

The log in VersionWorks does not look very detailed at first glance but it displays the most essential information about the files. When checking in a file the user can write a comment about the change, a reason for the change and even select keywords from a list that can be attached to the revision and thereby making it easier to locate it in the log. By switching to ProjectState you will find out that the log in VersionWorks is the most detailed of all the tested applications. There are numerous ways to display it and possibility to use filters to only show wanted information. There are a few predefined filters that can be useful, for instance only to view checked-out files or files that have been altered since a specific date or time. The log provides complete overview of all projects under version control in VersionWorks and is easy and fast to browse through.

There is much more to VersionWorks than just version control. It is for instance possible to create workflows and put files "Under Construction" to indicate that they are accessible by anyone else than the developer that have checked them out. When a file is under construction it is marked with an icon in the log and the reason why it has been locked is displayed. This makes the file stand out of the log, making it easy to see which files are being worked on at the moment. It is also possible have automatic e-mail dispatches to users that are concerned about alterations to projects that are part of different workflows.

It should be pointed out that since there are so many ways to configure VersionWorks, it is hard to explore them all when not being given a proper introduction to the application. When buying the application the introduction course is included and attending this is necessary. Using VersionWorks as a simple version control tool would not require any education about the application but since it offers so much more it would be foolish not to use it to the fullest.

The overall impression of VersionWorks is very good. The application is fast, easy to get started with and feels very robust and well constructed. Despite the fact that it does not support graphical display for Simatic S7 files, VersionWorks would probably satisfy all the needs of the company when it comes to a CM tool.

Application	File type	Supported	Ability to show differences
Rockwell RSLogix 5000	.ACD	Yes	Yes
Siemens Simatic S7	.zip	Yes	Yes (but not graphically)
Beijer E-Designer	.mpa	No	No
Wonderware Intouch	.zip	Yes	Yes (but not graphically)
Microsoft Word	.doc	Yes	Yes
Microsoft Excel	.xls	Yes	Yes

7.3.6 Test results of AssetCentre

The main window gives a feeling of being very simple. The number of clickable items is relatively few and if this is all that is needed for AssetCentre to be a successful CM tool, it sure must be user-friendly. And it is. When working with the files under version control you simply right-click on them and only the available commands are displayed. If you want to add files or folders you change to "Design"-mode and the right-click menu will provide you with other options such as adding files. This makes AssetCentre easy to get started with.

When it comes to comparing revisions there seems to be no such feature. This is of course a drawback but older revisions can be easily accessed from the audit log. If the comments made when checking the files back in are good enough, the required information about the changes can be found in the log where the different revisions can be opened with their associated applications.

The log is divided into an audit log and an event log. The event log contains information about the files in the repository. It shows which files have been added, checked in and checked out. The audit log contains additional information, such as which users have logged in and out of the system and when someone has performed an action that caused an error. Compared with the other CM tools that have been tested, the log in AssetCentre is one of the more detailed.

Setting permissions and configuring groups and users are done in the FactoryTalk Administration Console, which is an independent application that cannot be accessed from within AssetCentre. The Administration Console is easy to use and lets you configure everything from rights to override checkouts to which actions that should be displayed in the audit log. AssetCentre is thanks to the variety of choices in the Administration Console highly customizable and flexible.

The biggest drawback with the application is the lack of support for Siemens and Beijer. They can however be added as custom files and since there is no possibility to do a graphical compare anyway, and if the need for this feature can be overlooked AssetCentre might be able to function as a pure version control application. However, it feels like the lack of support for most of the tested file types make AssetCentre unfit for Tetra Pak.

We had to use the lab in the Platform Solutions department to try AssetCentre since there were a limited amount of licenses for the application. Unfortunately Simatic S7 and E-Designer were not installed on the computers with Asset Centre. Therefore we cannot say anything about how the version control of those types of files work and since Simatic S7 differs from other PLC software (using nodes for example) there might be compatibility issues when trying to run Simatic S7.

Application	File type	Supported	Ability to show differences
Rockwell RSLogix 5000	.ACD	Yes	No
Siemens Simatic S7	.zip	No	No
Beijers E-Designer	.mpa	No	No
Wonderware Intouch	.zip	Yes	No
Microsoft Word	.doc	No	No
Microsoft Excel	.xls	No	No

7.3.7 Test results of Proficy Change Management

The first impression you get when opening Proficy Change Management is that it is an ambitious program. Creating users and groups are easy and the security level seems to be highly customizable.

The application uses check-in, check-out and locking of files. Therefore there are no risks of ending up in conflicts.

As mentioned in *Chapter 6.1.5 Proficy Change Management*, the application uses the built in compare functions of the different development applications. This has however not functioned as well as we were hoping.

When trying to compare RSLogix 5000 files, PCM only lets you know that the two different versions being compared are binary and of different size. Even with the Rockwell comparison tool installed on the computer Proficy Change Management failed at showing the differences graphically. Our contact at GE Fanuc raised this as an issue at their GlobalCare technical support team but they have not succeeded at giving a solution to the problem.

For Siemens it works better but to be able to make the compare functionality work, you must choose which Simatic Stations you have in your program. It seems as the name Simatic 300 Station is the default name, but if you have another name for your station in your program, this name must be typed in manually. If you have several stations in your program you must add these and give them the correct name. This feels unnecessary since it should not be that hard for Proficy Change Management to import the names automatically.

When comparing Siemens files Proficy Change Management uses the built in compare function in Simatic. This built-in compare feature can only detect differences in the code and will not notice if you add new hardware, new connections or change the icon library. This is a huge drawback. If someone for instance adds or changes the connections and do not mention this in the log when checking the file back in, there is really no way to know that these changes were made.

Proficy Change Management has no support for Beijer but this can be added by using the Custom Project function. This allows you to add support for any type of files you want but if the added program has a built-in compare tool, this functionality can only be achieved through a script that GE Fanuc is willing to construct.

For the more common files like MS Word and MS Excel the compare does not work. Even though MS Word has a built in merge tool the option to compare different versions is not available when right-clicking on the file.

When using Proficy Change Management with Wonderware files we encountered a strange error. Adding the files to Proficy Change Management works fine and checking out them too. After altering the file and trying to check it back in, a dialog box tells you that the file no longer exists on your local computer and that you have to undo the check-out and start over by checking out the latest version from the server. When undoing the check-out you get yet another error message telling you that the operation failed, but somehow it seems as if the file is checked back in. After this you can check out the file from the server but the option to open it with the editor is faded and unavailable. We tried in every way possible to get it to work with the same result as described. Interesting is that the option to compare appears when right-clicking on the Wonderware file. Unfortunately, as described, we were not able to create a second revision of the file and were thereby unable to try the compare function.

The overall version control seems to run perfect and the log is easy to read and also gives the user the possibility to label files with a version number. The option to label a revision and have it visible in the audit log is a good feature. This makes it possible to exclude the date or the version number in the file name.

A huge drawback which more or less makes this program useless to Tetra Pak is that there is no way to change the tree structure of the folders. It is for instance not possible to create a folder, with a name of your choice that can contain files from different development tools. This means that all Rockwell files under version control has to be in the Rockwell folder, all Siemens files in the Siemens folder and so on. When working with numerous projects, all the files are spread out in the different folders in the tree and mixed together. This will certainly lead to confusion and if there are MS Word- and MS Excel files associated with project files, it will be impossible to keep the work organized.

Our contact at GE Fanuc informed us that we could use the Factory Layout to achieve the file structure that we wanted. The Factory Layout displays the project as HTML pages in the InfoViewer pane and you navigate in your project through these. This use of PCM requires some HTML programming and do not give a good overview of the project. It also seems to suffer from some small bugs that make Proficy Change Management freeze and sometimes shut down. This happens especially after modifying the source code and choosing generate preview. After you have created the HTML page you have to publish it on the server. This seems to require Administrator rights to the server and working in this way would make simple tasks as creating folders and files tedious.

The main window of Proficy Change Management is simple, easy to get an overview of but graphically it feels a bit dull. For example, all the files stored on the server have the same icon and the folder icons seem outdated. There are not that many options and clickable items and the layout is a bit strange. For example, to change the preferences for the applications, you have to check out the “Custom Projects Settings” file for that application. Then you have to alter code in the script file. There is no easy guide describing how to do this or wizard that leads you through it, the scripts are simply opened through the MS Explorer with Notepad or the editor you prefer. This set up requires you to contact GE Fanuc if you want to tweak the preferences and this adds to the feeling of PCM being too rigid. The impression you get after using Proficy Change Management for some time is that the application is deficient at fulfilling the test specifications even though it would probably fulfil the version control requirements after some alterations in the configuration script files.

Application	File type	Supported	Ability to show differences
Rockwell RSLogix 5000	.ACD	Yes	Yes, but it does not work properly
Siemens Simatic S7	.zip	Yes	Yes
Beijers E-Designer	.mpa	No	No
Wonderware Intouch	.zip	Yes	Unclear
Microsoft Word	.doc	Yes	No
Microsoft Excel	.xls	Yes	No

7.3.8 Test results of AutoSave

The layout of AutoSave is very similar to Proficy Change Management. The biggest difference is that it is easier to get an overview of and it is possible to create a tree structure freely.

When starting the application you will notice that there are not many icons or clickable items. The interface is simple and easy to get acquainted with. This is positive since the learning curve is short, but this also means that are options that are left out.

The tree structure of folders is easy to set up. For every new folder, or Area as it is called in AutoSave, you get the possibility to set permissions for users and user-groups. These settings can be altered at any time by choosing properties for the area you are working with. Applying restrictions for user groups and even single users is easy and fast. All settings in AutoSave and all documentation about files are stored in the SQL database and are easy to restore in case of a crash or similar malfunction. The projects under version control are stored as zip-files on the server and by backing it up they can also be easily restored.

The homepage of MDT Software claims that AutoSave can perform a compare to any two versions of a file but this does not function as expected. AutoSave can compare files and tell the user that the files are different in some way. Graphical compare can only be done if this functionality exists within the development tool since AutoSave utilises this functionality if possible. The personnel that performed a demonstration of AutoSave specifically pointed out that AutoSave can not perform any compares that the development tool associated with the files can not do. AutoSave works as a shell and functionality apart from the version control aspect depends on the development tools. The only graphical compare that AutoSave itself can perform are on pure text files. The result is presented in a table in HTML format and shows the text changes and which rows have been affected. This however is extremely hard to understand especially when rows have been deleted since the row numbers between the two different versions will no longer match. When comparing text files it would be much better with a split screen window showing both versions and colour marking on the changes.

When it comes to comparing different versions of Siemens Step 7, Wonderware Intouch and Rockwell RSLogix 5000 files, the results should be the same as for Proficy Change Mangement since AutoSave also uses the built-in compare functionality of the development tools. However, we were not able to try this feature with Rockwell RSLogix 5000 or Wonderware Intouch since the modules for supporting these applications were not installed on the test computer provided by NOVOTEK. During the demonstration the personnel from NOVOTEK could not say for sure if the compare would work for RSLogix 5000 since it uses a separate compare application. Comparing Wonderware Intouch will not work since there is no built-in support for comparing files in the Wonderware application.

AutoSave does not support Beijer files, instead the universal module has to be used to add these type of files. When comparing different versions of universal files that are binary, they are simply compared bit by bit and the result just shows that a difference exists.

Conflicts can not occur since all checked out files are locked and this is explicitly displayed when highlighting a file in the tree structure. To be able to correct mistakes like forgetting to check files in you can apply settings that dispatches e-mails when a file has been checked out for a predetermined period of time. There are also settings that lift the lock. Unfortunately AutoSave does not give a warning if the application is closed while files are still checked out. It does however notify the user of checked out files when the application is started. The files in the tree structure do not have any overlaying icons that clearly show if they are checked out. To be able to see if they are, they have to be selected and the current state of the file will be displayed in the main window.

Setting up the tree structure of files and folders is fast and easy. It is very straightforward to create a wanted tree structure but moving files around to different directories is not possible once the tree structure is in place. This requires the user to place the files in the correct folder from the beginning. If the tree structure for some reason has to be altered, it is possible to create a new tree structure for a project and import the old files but this is not a good idea since the revision history will be lost.

The overall appearance was not very appealing but on the contrary, the tool is fast and stable. AutoSave does give a good overview of projects and files that are under version control and it is easy to find sought after files through the search ability. Finding wanted versions of files might be a bit more difficult since the log only can be reached through a report that is generated in HTML format and reached via the web browser. This means that there is no possibility to search the log from within AutoSave. The reports that can be generated are Client User Log, Compare Groups, Configured Agents, Locked Programs and Program Zip Files. The most interesting report is the Program Zip Files report. This report lists all the activities about the files of a certain project. Unfortunately there are no filtering options when generating the reports but a good feature is that the reports can be automatically generated by AutoSave and dispatched via e-mail.

Other tools, e.g. VersionWorks, seem to be more focused on the software development stage of a project whereas AutoSave most likely will come to its best use when a plant is up and running. It seems as AutoSave is better suited as an asset management tool than a configuration management tool. There are no problems with the version control in AutoSave but the limited amount of choices and lack of functionality makes AutoSave unsuited for BU DBF.

Application	File type	Supported	Ability to show differences
Rockwell RSLogix 5000	.ACD	Yes	Unclear
Siemens Simatic S7	.zip	Yes	Yes
Beijers E-Designer	.mpa	No	No
Wonderware Intouch	.zip	Yes	No
Microsoft Word	.doc	Yes	No
Microsoft Excel	.xls	Yes	No

7.3.9 Test results of Microsoft SharePoint

The version control features of SharePoint works very well for Microsoft applications. Using the application for managing files and sharing information proved to be an easy task. SharePoint offers numerous ways and options for displaying and sharing items but the application does not have the ability to compare versions of files.

The interesting aspect of SharePoint from Tetra Pak’s point of view is the centralized version control of the application. When it comes to handling other Microsoft developed software types, SharePoint excels in comparison to the other tested CM tools. This is however to be expected since the tool is developed by Microsoft. Using SharePoint together with the development tools that we are investigating is on the other hand a more complicated matter.

The application uses the web browser as an interface for handling the data stored on the server. This means that when a file is checked out, it is actually still located on the server and work done to the file is performed on the actual server. There is the option to download a file directly to the computer but this means that the file is not longer under version control and the user has to upload the file again after working with it. Obviously this is not the way in which we want to work as it completely misses the point of version control.

There is no compatibility between SharePoint and RSLogix 5000. When accessing the file, SharePoint recognizes it as an .ACD file. This means that the file can be checked out and then opened with RSLogix 5000 through SharePoint. When changes were made and the file was saved and checked-in, the changes were not saved to the server. This made the SharePoint useless when working with RSLogix files.

As expected the tool was not very well-suited when working with Siemens files either. The problem in this case starts when trying to “open” the .zip files. SharePoint wants to automatically start the extraction of the zip file and save its contents to the computer but Simatic S7 imports the entire zip file and no extraction of the file is needed. SharePoint does not have any operation similar to “open with” to allow the file to be opened through any other application than the one recognized by SharePoint. Since the file could not be handled in the way that we wanted and the fact that not even Simatic S7 tool could be opened trough the interface, this led us to the conclusion that SharePoint is not in any way compatible with the Simatic S7 files.

The same can be said about Beijer files. SharePoint identifies the .mpa files as .zip files for some unknown reason. Consequently the same problems were encountered, and the same conclusion can be made, as for Siemens.

Wonderware InTouch projects are saved in a folder with subfolders containing numerous files. Since SharePoint does not allow the check-in of folders, just single or multiple files, it made it incompatible with the development tool.

If a project contains many subfolders it would take a long time to recreate the wanted tree structure since all folders would have to be created in SharePoint and this is done in several steps through menus. Working in this way is very time consuming and not very user-friendly.

When it comes to the ability to granting access to folders and files to different users SharePoint is easy to configure. The application allows the user to set up rules for access restriction to folders. Also the actions that are allowed to be performed to a file can be configured for either individual users or for an entire group.

The overall experience of SharePoint is that it works perfect with Microsoft files and the application has many features that makes it very user-friendly. In appearance SharePoint is very esthetical appealing and since the web browser is used as interface, it can be altered very easy.

Application	File type	Supported	Ability to show differences
Rockwell RSLogix 5000	.ACD	No	No
Siemens Simatic S7	.zip	No	No
Beijer E-Designer	.mpa	No	No
Wonderware Intouch	.zip	No	No
Microsoft Word	.doc	Yes	No
Microsoft Excel	.xls	Yes	No

7.4 General Discussion

When starting to compare the different CM tools that have been tested there were a few aspects apart from the software testing specifications that had to be considered. The CM tool had to, except from being able to successfully version control the files, be possible to put into practice. The tested software might have been able to perform efficient version control but still be impossible to implement within the company. This is due to different reasons that, among others, will be discussed in this chapter.

When looking at the freeware applications that were tested and the commercial ones you soon realize that the freeware tools are almost equally good at performing good version control, displaying a good audit log and possibility to tag and label files. The file structure is also highly customizable and the learning curve is short. So what motivates the high price tag on the commercial tools?

It is interesting to take a closer look at the freeware programs since they are free of charge, perform the task of version handling of files but lack the ability to do graphical compare. This gives rise to the question how important the ability to graphically compare changes really is? The answer varies depending on who you ask. It is obvious that this ability facilitates the work substantially but when looking at the survey results and the interviews there were only a few persons who thought that the graphical compare functionality would be a nice feature to have. When implementing a tool one should rank the features that are wanted and choose a tool thereafter.

It is clear that the version control ability is the most important, the log has to be functional and that the other features that come with the programs should more be considered as bonuses. While testing the different tools we have looked at how reliable the version handling is, how easy it is to set up file structures, user-friendliness, how detailed and easy the log is to navigate and the ability to do graphical compares. We would argue that the order that we just mentioned the features in also reflects how they should be ranked. Even though it would be nice to have the graphical compare functionality it is close to impossible to find an application that can perform this operation on all the files used within the company. One of the reasons to why we find that this feature is not that important is the fact that it is always possible to see the change comments in the audit log. This gives the user a good idea of the changes that where made from one version to another. Considering the fact that there is always the ability to open the different revisions with their associated development tool – this makes the graphical compare feature less important. The results from the interviews and the survey also confirm that the most wanted feature by the developers is a way of managing files and work in the same way in all departments while only a few people requested the compare functionality.

When reflecting the fact that the commercial software tools are expensive to buy and implement within a large company, one might be more compelled to choose a free-ware tool. This is however not a good idea, especially within company with many users.

The reasons for choosing one of the commercial tools instead of the free-ware are many. The strongest argument for doing this is the guarantee of getting secure, established and stable software to work with. The commercial tools are being developed to be compatible with especially PLC and HMI files and this alone might be enough to motivate the price of the product.

When choosing a freeware tool the risk is taken that bugs and other more serious flaws can be found in the application. For example a repository created by an earlier version might not be supported by a newer release. This will very fast become a problem since free-ware programs a constantly updated and newer releases are made public very often. The free-ware applications also rely on the users to report bugs in the application and since it is free of charge, there are no obligations towards the users from the company developing the free-ware tools.

Implementing a freeware tool would create a lot of work and a lot of time would be spent keeping track of the latest releases, bug-fixes and on upgrading the application. At a big company like Tetra Pak this can be such a huge undertaking that it might be necessary to hire personal that only deals with the administrative part of the application. This will generate expenses and there will be no pay-back time since this is a never-ending task.

The complete opposite can be said about the commercial tools. They require few or no updates compared to the freeware applications and they will in a more effective way facilitate the developers' work. Even though the commercial software cost money, the man-hours you can save by using such a tool will compensate and provide you with a payback time relative to the buy-in price of the product.

CM tools specialized at handling pure text files are often free or at least cheaper than the ones specialized at handling binary files. A company that only deals with pure text files in their projects can settle for a cheaper application. In new companies with young employees that are acquainted with the concept of carrying out good CM, a freeware tool would probably be good enough. If the users know the tool well enough to be able to handle the speed with which updates and bug-fixes occur to the CM tool there is no reason why not to use it. During our work we have been in contact with friends working at other companies. The tools they use are not necessarily used within the whole company but maybe in minor development teams. Our contacts have informed us that TortoiseSVN are used within ÅF, Schneider Electric and TAT. AXIS uses subversion and Sony Ericsson is using Clearcase. This is the reason why TortoiseSVN was the first tool we tested and it was disappointing that it did not work as good as we were hoping, see *Chapter 7.3.1 Subversion/TortoiseSVN* for more detailed information.

The other freeware applications remind of TortoiseSVN and most of them require subversion to run. In spite of this none of them functioned as well as TortoiseSVN. Since our impression is that TortoiseSVN is the best freeware version control tool available, this ruled out the freeware applications for solving the CM problem within BU DBF and had us focused on the commercial tools. In many cases they also offer much more than just version control.

When looking at the commercial tools there are a few deliberations you have to consider. How large is the chance that an implementation of the tool would be successful? Is it user-friendly enough or would it require too much administrative work? Is the price tag of the application motivated by its functionality? This is only few aspects to consider because it is always a risk when introducing a new application that in a sense would force people to approach their work in a completely different way.

Even though it is pretty clear that many people within the company would like to see a working CM system being put in use – judging from the survey and the interviews – it still requires the full cooperation of everyone intended to use the application. There is no question about the fact that better CM routines and a tool would solve many of the problems today as long as the tool is used correctly and the routines are followed. If not, an implementation could easily cause problems. The key here is knowledge and if people do not understand the theoretical background of the CM concept, the practical outcome will not work.

A good and fully working implementation of CM would only have positive effects. The security and comfort of not being able to delete work by accident and complete and organized version history of files are the obvious benefits. However, the tool will also solve other requests if good and well-formulated routines are introduced.

There have been wishes to be able to see which versions of PLC and HMI code that is compatible with Tetra PlantMaster. Building some sort of compatibility matrix have been up for discussion, but by doing this you only fall back into old tracks and do not use the benefits of good CM. By formulating routines that for instance require the developers to label or tag a new release with a keyword that tells which version of PlantMaster the release is compatible with, the problem is already solved. Since the log in all CM tools is searchable you would only have to type the Plantmaster version you are interested in to get a list of all compatible PLC- and HMI-applications, or vice versa. This is the traceability aspect of the CM model and once again points out the importance of understanding the theoretical background.

The only thing that would cause an implementation of CM within BU DBF to end in failure is improper use of the CM tool, neglecting the CM routines or the financial aspect since licenses are expensive to buy. But the fact is that even though it is a large investment, there is a payback time. A developer can save several minutes when browsing for files with the use of a CM tool and avoid confusion about which version is the latest for instance. As soon as these saved minutes turn into hours during the course of a few months you can really see the economic benefits from using CM. By looking at it from this angle you realize that a CM tool will actually pay itself of in what can be considered a relatively short amount of time. Yet another reason to acquire a CM tool is that you are actually buying the know-how and upkeep by the developer of the CM tool. The responsibility of maintaining the tool is partly shifted from the company to the developer. This is an important aspect since the tool should facilitate the work and not obstruct it.

8 Conclusions

In the software developing industry every company that strives for being prosperous and successful should introduce or have already introduced CM in their day-to-day work. The possible disadvantages with CM are negligible in comparison with the advantages. Certain operations may consume a bit more time when using a CM tool – such as checking in and out files – but the time spent on locating sought after files will be considerable shorter. In other words, certain aspects of CM that can be considered inconvenient are by far compensated with the benefits of such a system.

After studying the results from the survey and the interviews it was obvious that the employees would like to have a CM system put in use. CM would definitely help to sort out some of the problems that exist today. The main advantages that BU DBF would get from CM, based on our case study, is:

- Easier and time reduced location of sought after files
- Structured and easy accessible documentation of changes between different versions of files
- Good labelling of files would provide full traceability and possibility to see compatibility between different software, hardware and Tetra PlantMaster
- Customizable and secure access restriction to projects
- Full backup of all versions and revisions
- If implemented throughout all categories, work would be performed and projects would be stored after the same structure
- Possibility to graphically view changes between different versions
- Better overview of work status of projects and files
- Less room for user mistakes when copying, moving, deleting and creating files and folders
- Automatic e-mail notifications when updates are performed

BU DBF would probably find more benefits with CM than the bullets listed above. It all comes down to identifying the problems and finding the best way in which CM can resolve them.

8.1 Best suited tool

In our search for a CM tool we have come across both freeware and commercial tools. As mentioned in *Chapter 7.4 General Discussion*, the freeware applications can serve as good enough tools for companies that only deal with pure text files. This is not the case for BU DBF and therefore the choice stands between the different commercial tools available on the market.

The best tool we came across in our search is, without a doubt, VersionWorks from GepaSoft. Apart from being able to successfully perform flawless version control of all the tested file types and also being able to graphically compare the differences between different revisions of a file, GepaSoft seems to be a very ambitious company. We have had several telephone conversations with the company, both from their sales and their technical assistance department, and they are eager to help in anyway possible. It has also come to our attention that GepaSoft recently closed a deal with a new customer in Denmark and that they are willing to provide Tetra Pak with information about how their implementation of VersionWorks has turned out.

It is hard to rank the other tested CM tools, mainly due to the fact that VersionWorks was far superior in every aspect. All the other tools had their strengths and weaknesses but overall none of them fulfilled the test specifications and are therefore not suited for BU DBF.

8.2 Future work

The first steps towards an implementation of CM within BU DBF can be difficult. The transition from having no prior CM tools and no official routines to a full implementation will probably be a fairly long process. The required tool for solving the task is not cheap and the economic sacrifices should carefully be compared against the possible benefits in other areas that a CM system would provide. The result of such research will probably lead to the conclusion that a CM system would be worth the expense and in the software developing industry CM is more or less required. As stated, the implementation is not easy and for many companies it has failed. It might be a good idea to study such reports to avoid possible pitfalls.

So what is the first step towards a CM implementation?

Before even thinking about implementing CM it has to be introduced. A new CM approach will force the developers to work in a slightly different way than today and this will definitely not be greeted with open arms. This is one of the main reasons for failure when implementing CM in a company [3]. The first step should be some kind of workshop or seminar where the basic concepts are explained. This is to make sure that everyone understands the reason for introducing CM and what the possible benefits would be.

Before introducing a CM tool good CM routines have to be formulated. This is something that has to be planned carefully. Unclear CM routines will lead to lack of confidence for CM and consequently the CM tool. Who would want to work in a different way than today when there seems to be no advantages?

To implement the use of a CM tool the best approach would probably be to acquire enough licenses to be able to try it out in a small development team.

Our recommendation is to hire a consultant that is specialized at working with Configuration Management to help formulating the guidelines and routines that should be carried out. The CM tool to be used should in our opinion be VersionWorks since it was the application that best fulfilled the test specifications, see *Chapter 7.3.5 Test Results of VersionWorks*.

9 References

[1] Dart, S. (2000) Configuration Management: The Missing Link in Web Engineering. Artech House Publishers

[2] Babich, W.A. (1986). Software Configuration Management, Coordination for Team Productivity. Boston: Addison-Wesley.

[3] Bendix, L.; Vinter O. (2001) Configuration Management from a Developer's Perspective.

[4] Moreira, M. (2008) "What is a CM Tool?"
cm//crossroads (Online Community)
URL: http://www.cmcrossroads.com/content/view/12643/120/
Read: 2009-01-19

[5] Azad, K. (2007) "A Visual Guide to Version Control"
BetterExplained (Homepage)
URL: http://betterexplained.com/articles/a-visual-guide-to-version-control/
Read: 2008-09

[6] Azad, K. (2007) "Intro to Distributed Version Control (Illustrated)"
BetterExplained (Homepage)
URL: http://betterexplained.com/articles/intro-to-distributed-version-control-illustrated/
Read: 2008-09

[7] Koch, S.A.(2006) "Testing vs. Quality Assurance"
cm//crossroads (Online Community)
URL: http://www.cmcrossroads.com/content/view/6782/120/
Read: 2009-01

[8] Tetra Pak. (2008) Tetra Pak i korthet.

[9] Tetra Pak
Internal website – Orbis
URL: N/A
Read: 2008-12

[10] Tortoisesvn – Project Home
Tigris.org (Open source community)
URL: http://tortoisesvn.tigris.org/

[11] Rapidsvn – Project Home
Tigris.org (Open source community)
URL: http://rapidsvn.tigris.org/

[12] RapidSVN
SourceForge (Open source software community)
URL: http://sourceforge.net/projects/rapidsvn/#item3rd-1

[13] Mercurial
Mercurial (Homepage)
URL: http://www.selenic.com/mercurial/wiki/index.cgi/Mercurial

[14] Bazaar
Bazaar (Project Homepage)
URL: http://bazaar-vcs.org/

[15] GEFANUC - PCM

[16] GepaSoft - Versionworks

[17] Rockwell Automation - Asset Centre

[18] Windows Share Point Services
Microsoft Office Online (Homepage)
http://office.microsoft.com/sv-se/sharepointtechnology/FX100503841053.aspx

[19] AutoSave
MDT Software – AutoSave (Homepage)
URL: http://www.mdtsoft.com/products/autosave/

[20] Design & Configuration RSLogix 5000
Rockwell Automation (Homepage)
URL: http://www.rockwellautomation.com/rockwellsoftware/design/rslogix5000
Read: 2008-10

[21] Description - Overview
Siemens (Homepage)
https://mall.automation.siemens.com/SE/guest/index.asp?aktprim=0&nodeID=5000153&lang=en&foldersopen=-1714-1713-1-2374-2375-2376-2377-&jumpto=2377
Read: 2008-10

[22] Beijer Electronics. 4: Th Paragraph”E-Designer – ett verktyg för alla behov”
http://www.beijer.se/web/BExFilePileAUT.nsf/0/9106E8A8A93A8152C125728E0059FEC1/$FILE/E-SERIEN_SV_BR00447_0504.pdf
Read: 2008-10

[23] Wonderware InTouch HMI
Wonderware (Homepage)
http://global.wonderware.com/EN/Pages/WonderwareInTouchHMI.aspx
Read: 2008-10

[24] AutoCAD
Autodesk (Homepage)
http://usa.autodesk.com/adsk/servlet/index?siteID=123112&id=2704278
Read: 2008-12

[25] TortoiseSVN
SourceForge (Open source software community)
URL: http://sourceforge.net/projects/tortoisesvn

[26] Configuration Management
Wikipedia
URL: http://en.wikipedia.org/wiki/Configuration_management
Read: 2009-03-19

[27] NOVOTEK
Thomas Lundqvist
Sales Manager

10 Appendix 1: Basic usage of the tested CM tools

10.1 TortoiseSVN

Four general steps for TortoiseSVN usage:

1. Create global repository
2. Checking out repository/Create local repository
3. Put files under version control
4. Sharing work

1. To create a global repository in TortoiseSVN use the command Create Repository here, see *Figure 10.1-1*. The global repository should always be created in a public location.

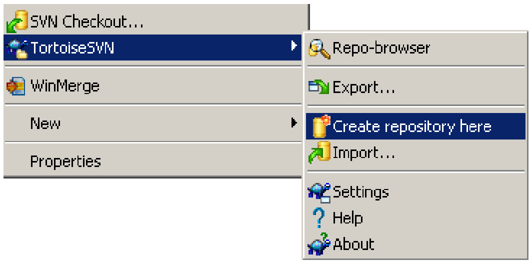
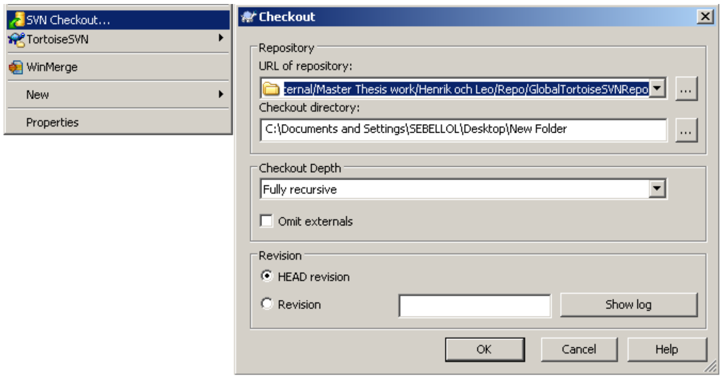


Figure 10.1-1: Create a global repository.

2. Creating a local repository is done by using the command SVN Checkout, see *Figure 10.1-2*. SVN Checkout will copy the global repository.



Repository

URL of repository: ternal/Master Thesis work/Henrik och Leo/Repo/GlobalTortoiseSVNRepo

Checkout directory: C:\Documents and Settings\SEBELLOL\Desktop\New Folder

Checkout Depth: Fully recursive

□ Omit externals

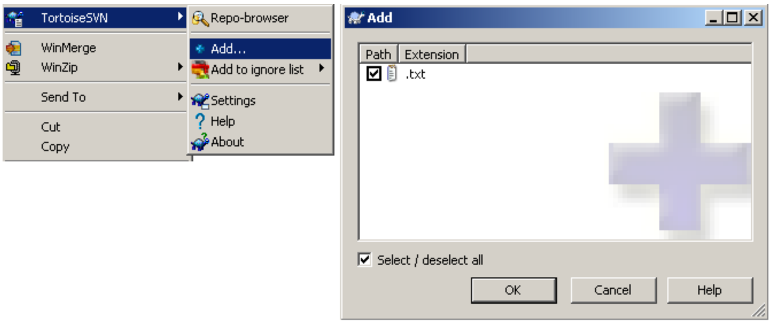
Revision

● HEAD revision

○ Revision

Figure 10.1-2: How to create a local repository.

3. To put a file under version control it must be added to make ToitoiseSVN aware of its existence. After placing the file in the local repository the Add command have to be used, see *Figure 10.1-3*.



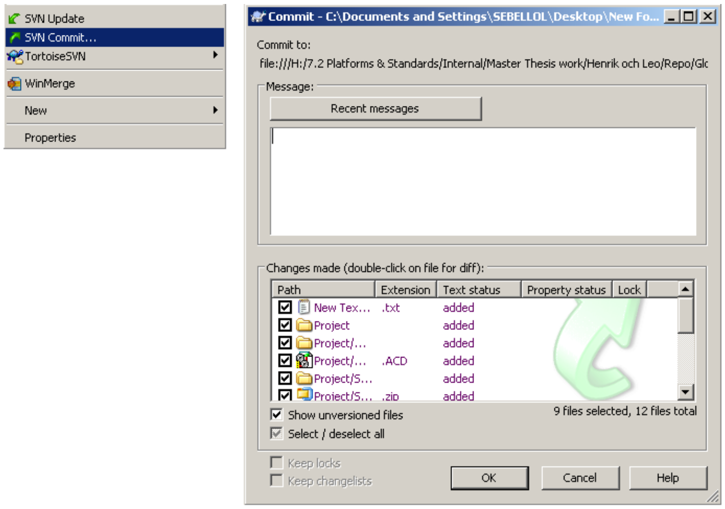
Path	Extension
☑	.txt

☑ Select / deselect all

Figure 10.1-3: Use the "Add…" command after placing a new file in the local repository.

The Add window will then appear and the files that the user wants to add should be selected.

4. To share the work in the local repository the files in it must be committed to the global repository this is done with the SVN Commit command, *Figure 10.1-4*. In the Commit window a commit message can be add and the user is asked to select the files to be committed.



Commit to:
file:///H:/7.2 Platforms & Standards/Internal/Master Thesis work/Henrik och Leo/Repo/Glc

Message:

Changes made (double-click on file for diff):

Path	Extension	Text status	Property status	Lock
☑ New Tex...	.txt	added		
☑ Project		added		
☑ Project/...		added		
☑ Project/...	.ACD	added		
☑ Project/S...		added		
☑ Project/S...	.zip	added		

☑ Show unversioned files
☑ Select / deselect all
9 files selected, 12 files total
☐ Keep locks
☐ Keep changelists

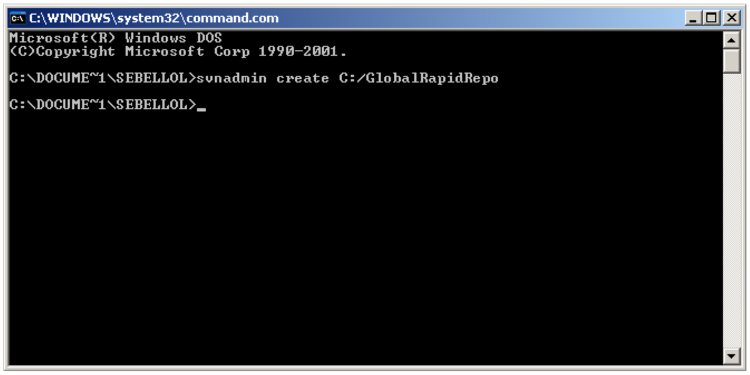
Figure 10.1-4: Share files by committing them to the global repository.

10.2 RapidSVN

Four general steps for RapidSVN usage:

1. Create global repository
2. Checking out repository/Create local repository
3. Put files under version control
4. Sharing work

1. Creating a repository is done through subversion i.e. through the command prompt. The command used is “svnadmin create c:/path_to_repo”, see *Figure 10.2-1*. Note that creating a repository on a network driver is not supported.



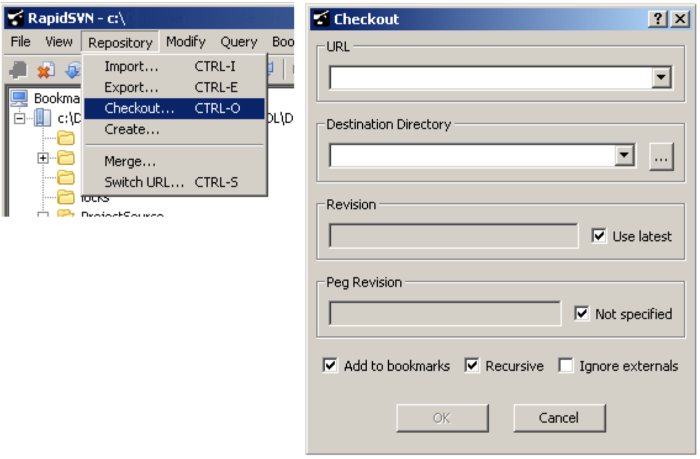
```
Microsoft(R) Windows DOS
(C)Copyright Microsoft Corp 1990-2001.

C:\DOCUME~1\SEBELLOL>svnadmin create C:/GlobalRapidRepo

C:\DOCUME~1\SEBELLOL>_
```

Figure 10.2-1: Use the command prompt to create a local repository.

2. To Check-out a repository the Checkout command is used. The next step is to fill in the URL of the server and the destination where the local repository should be placed, see *Figure 10.2-2.*



☑ Use latest

☑ Not specified

☑ Add to bookmarks ☑ Recursive ☐ Ignore externals

Figure 10.2-2: Checkout of a global repository.

3. Use the add button to put the files under version control, see *Figure 10.2-3*.

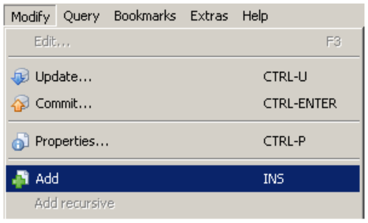


Figure 10.2-3: Use the Add command to put new files under version control.

4. To share the work from the local repository to the global repository the Commit command is used, see *Figure 10.2-4*. When committing a file a window will be prompted asking you to describe the changes that have been made to the file/files. This description is later shown in the audit log.

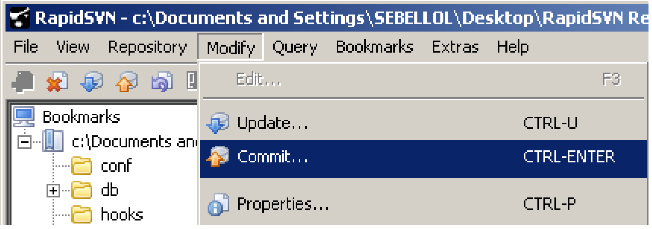


Figure 10.2-4: Share work by committing files to the global repository.

10.3 Mercurial / TortoiseHG

Four general steps for TortoiseHg usage:

1. Create Global Repository
2. Create Local Repository
3. Put files under version control
4. Sharing work

1. To create a global repository command Create Repository Here is used, see *Figure 10.3-1*. This is done by right-clicking in the folder where you wish to create the repository.

.

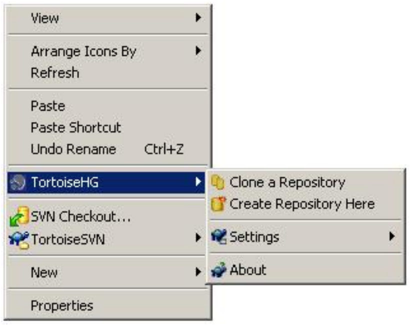
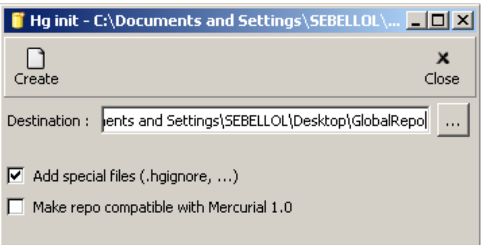


Figure 10.3-1: Create a global repository.

You will then see the window in *Figure 10.3-2* were you simply click the Create icon to create the repository. You can also change the location where the repository should be created by changing the destination. The Global Repository should be created at a shared location so all users can access it.



Hg init - C:\Documents and Settings\SEBELLOL\...

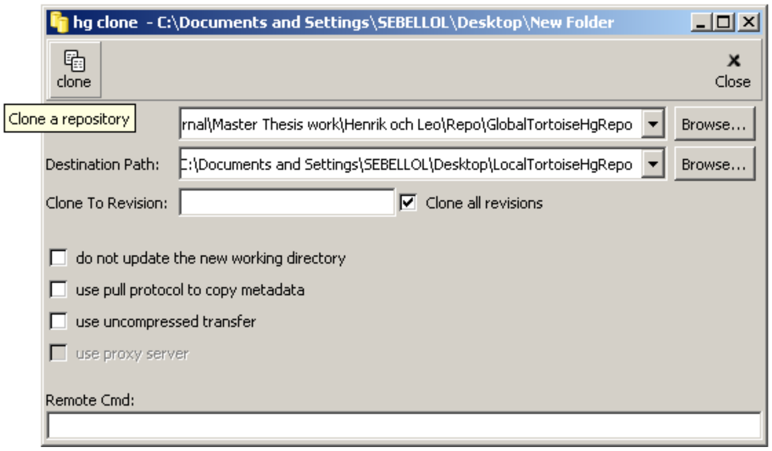
Destination : ents and Settings\SEBELLOL\Desktop\GlobalRepo

☑ Add special files (.hgignore, ...)

□ Make repo compatible with Mercurial 1.0

Figure 10.3-2: Click the create icon or change the destination.

2. The next step is to create a Local Repository and this done by using the same command as in Step 1 with the difference that it is done on your local computer. Alternatively you can clone an existing repository (i. e. a Global Repository) by using the command Clone a Repository. When doing this the hg clone window will appear, see *Figure 10.3-3*. Here you simply fill in which repository that should be cloned (Source Path) and where it should be cloned to (Destination Path) then press the clone button in the upper left.



hg clone - C:\Documents and Settings\SEBELLOL\Desktop\New Folder

rnal\Master Thesis work\Henrik och Leo\Repo\GlobalTortoiseHgRepo

Destination Path: :\Documents and Settings\SEBELLOL\Desktop\LocalTortoiseHgRepo

Clone To Revision:

☑ Clone all revisions

□ do not update the new working directory

□ use pull protocol to copy metadata

□ use uncompressed transfer

□ use proxy server

Remote Cmd:

Figure 10.3-3: Clone a global repository.

3. Adding files and folders, i. e. putting them under version control, is done by simply moving the wanted files and folder to the local repository and committing them by using the HG Commit command, see *Figure 10.3-4.*

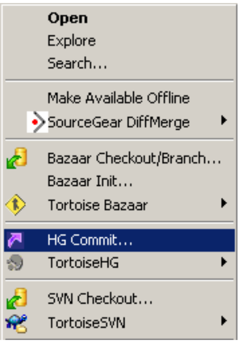
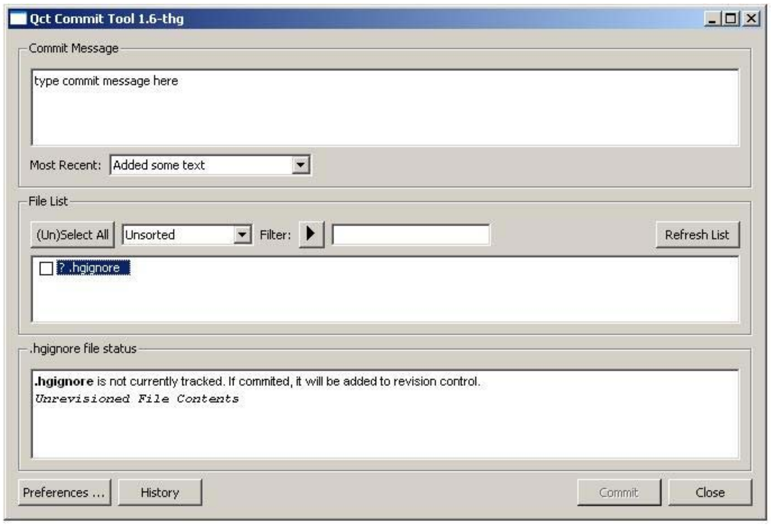


Figure 10.3-4: Committing files to global repository.

When this is done a new window will be prompted, see *Figure 10.3-5.* The user is asked to make a commit message (this text will later appear in the audit log) and choose which files that should be committed.



Commit Message: type commit message here

Most Recent: Added some text

Sort: Unsorted

□ ? .hgignore

.hgignore is not currently tracked. If commited, it will be added to revision control.
Unrevisioned File Contents

Figure 10.3-5: Enter a commit message.

Once a file committed it means that it is under version control in the repository.
Each time a user alters the contents of a file or a folder it has to be committed in order for the program to track it. The icon of a file that is not committed will have a red exclamation mark to show that its contents have been changed. After committing a file the overlaying icon will be a green check mark. The orange arrow lets the user know that there is a conflict with the file. See *Figure 10.3-6* for how the different icons look.

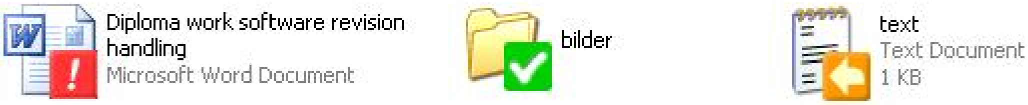


Figure 10.3-6: Overlaying icons in TortoiseHG.

4. The final step is to share the work/repository. This means sharing data either to other users or uploading it to the global repository. This is called synchronizing when using TortoiseHg, see *Figure 10.3-7.*

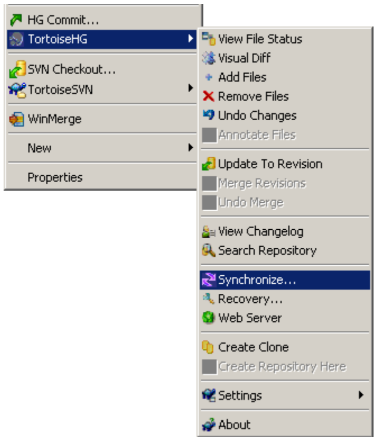


Figure 10.3-7: Use Synchronize to share work.

Using the Synchronize command will display the window in *Figure 10.3-8.*

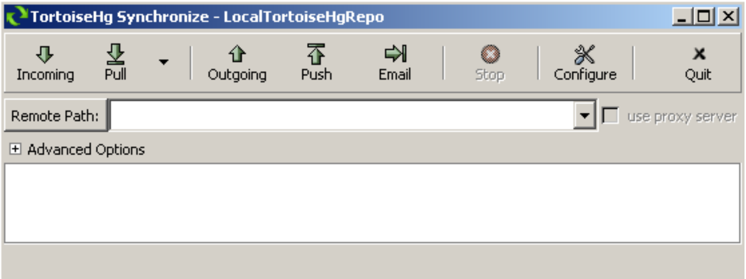


Figure 10.3-8: Synchronize window in TortoiseHG.

Pull will access the files/folders of a given repository and import them to the local repository you are working with. Incoming lets you see what files/folders you would get if you used the Pull command.

Push will share the files in the local repository to a different repository. Outgoing lets the user see what folders would be sent to the external repository.

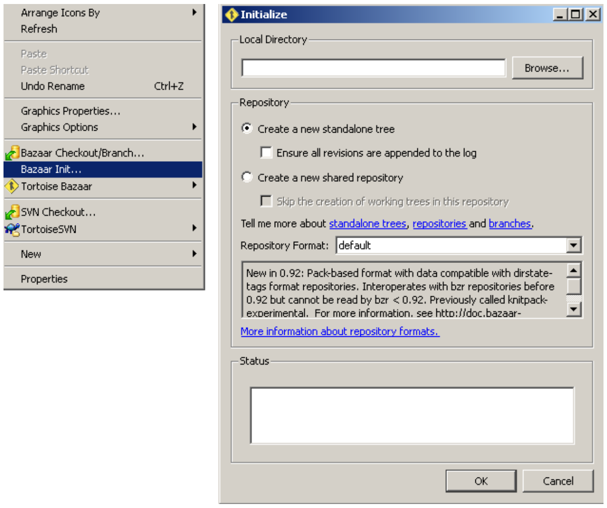
In the field Remote Path you specify the location of the repository you wish to pull from or push to.

10.4 Bazaar/Tortoisebazaar

When working with Bazaar four general steps are done:

1. Create Global Repository
2. Create Local Repository
3. Put files under version control
4. Sharing work

1. Executing the command “Bazaar Init…” will create a global repository. This will let the user access the Initialize window where different options for the repository can be selected, see *Figure 10.4-1*. In the field Local Directory the path of where the repository should be created has to be specified. The global repository should be created in location that all the users can connect to.



Initialize

Local Directory

Repository

● Create a new standalone tree

□ Ensure all revisions are appended to the log

○ Create a new shared repository

□ Skip the creation of working trees in this repository

Tell me more about standalone trees, repositories and branches.

Repository Format: default

New in 0.92: Pack-based format with data compatible with dirstate-tags format repositories. Interoperates with bzr repositories before 0.92 but cannot be read by bzr < 0.92. Previously called knitpack-experimental. For more information, see http://doc.bazaar-

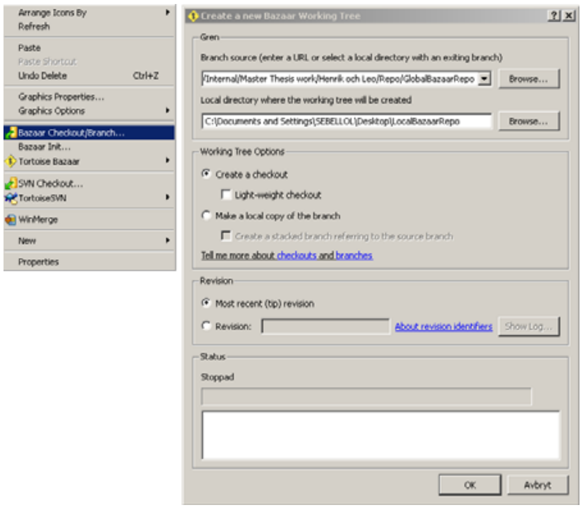
More information about repository formats.

Status

Figure 10.4-1: Choose "Bazaar Init..." to acess the Initalize window and create a global repository.

2. To create a local repository you can do as in the previous step or use the command “Bazaar Checkout/Branch…” to copy an existing global repository, see *Figure 10.4-2.*

In the “Branch source”-field the source of the global repository is specified and in the “Local directory”-field the location of the local repository is specified.



Gren

Branch source (enter a URL or select a local directory with an exiting branch)

/Internal/Master Thesis work/Henrik och Leo/Repo/GlobalBazaarRepo

Local directory where the working tree will be created

C:\Documents and Settings\SEBELLOL\Desktop\LocalBazaarRepo

Working Tree Options

● Create a checkout

□ Light-weight checkout

○ Make a local copy of the branch

□ Create a stacked branch referring to the source branch

Revision

● Most recent (tip) revision

○ Revision:

Status

Stoppad

Figure 10.4-2: Checkout or branch of a global repository.

3. To put new files under version control you put the files in the local repository and use the "Add…" command, see *Figure 10.4-3.*

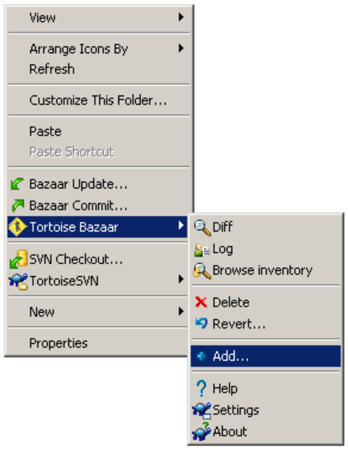


Figure 10.4-3: Put new files under version control.

Files in the local repository that are not under version control must be added and their icons are marked with a plus sign. Some of the other overlaying icons that a file might have are "one-way street" sign for files that are in conflict, green okay icon that show that the file is working properly and the exclamation mark showing that the file has been altered and needs to be committed to the global repository. See *Figure 10.4-4* for example of overlaying icons.

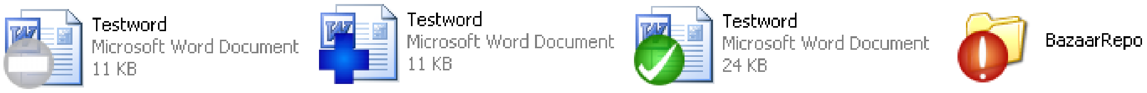
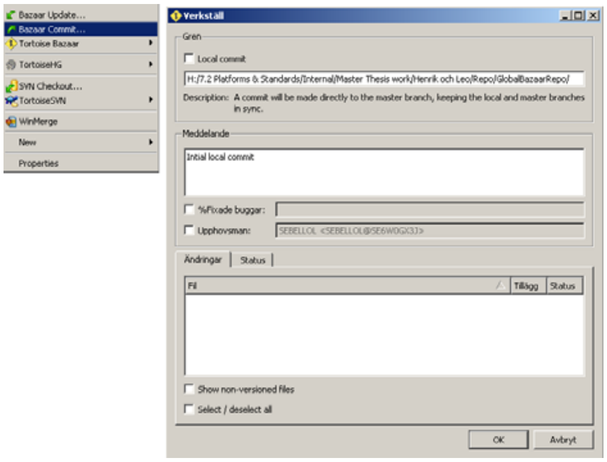


Figure 10.4-4: Overlaying icons.

4. Sharing and committing files are done with the “Bazaar Commit…” command, see *Figure 10.4-5*.



☐ Local commit

H:/7.2 Platforms & Standards/Internal/Master Thesis work/Henrik och Leo/Repo/GlobalBazaarRepo/

Description: A commit will be made directly to the master branch, keeping the local and master branches in sync.

Intial local commit

☐ %Fixade buggar:

☐ Upphovsman: SEBELLOL <SEBELLOL@SE6W0GX3J>

☐ Show non-versioned files

☐ Select / deselect all

Figure 10.4-5: Committing files with Bazaar.

Bazaar allows the user to either commit files directly to a global repository or to a local one. If the files are to be committed to a local repository, the Local commit box must be selected and the user has to fill in a message to be able to commit the files.

10.5 Proficy Change Mangement

Proficy Change Management is easy to get started with. Once PCM Server is installed you only need to install the client application on your local computer. At first logon you will be in Administrator mode and from here it is easy to add users and user groups.

The main window contains two panes – the Navigator and the InfoViewer, see *Figure 10.5-1*. By selecting to view preferences a third pane called Inspector will be shown. This window displays details about the selected file, see *Figure 10.5-2*.

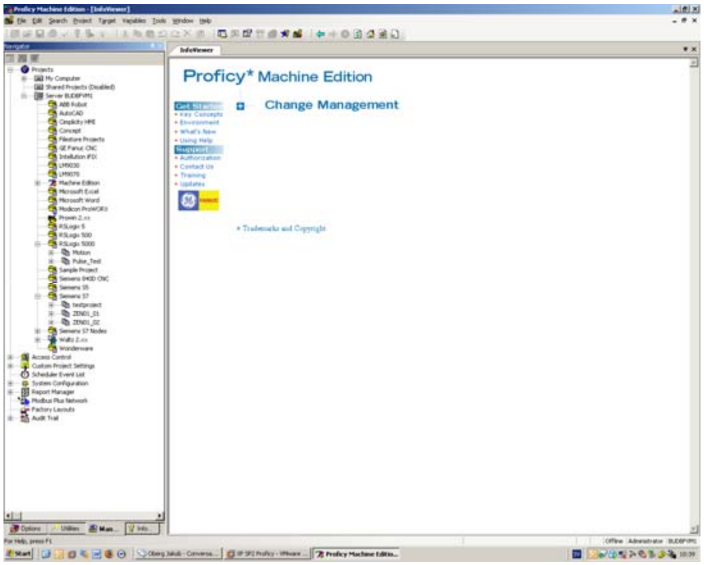
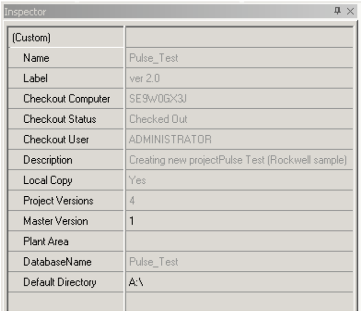


Figure 10.5-1: Main window. The Navigator pane to the left and the InfoViewer pane to the right.



(Custom)	
Name	Pulse_Test
Label	ver 2.0
Checkout Computer	SE9W0GX3J
Checkout Status	Checked Out
Checkout User	ADMINISTRATOR
Description	Creating new projectPulse Test (Rockwell sample)
Local Copy	Yes
Project Versions	4
Master Version	1
Plant Area	
DatabaseName	Pulse_Test
Default Directory	A:\

Figure 10.5-2: The Inspector.

As stated in *Chapter 7.3.7 Test Results of Proficy Change Management* there is no possibility to change the tree structure in the Navigator window. This might have to do with the inflexible file structure of Visual SourceSafe.

To put a file under version control in PCM you right-click on the folder that corresponds to the type of file that you want to add, see *Figure 10.5-3.*

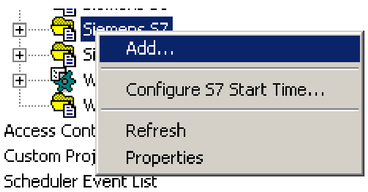


Figure 10.5-3: Add a file to PCM.

If the type of file you want to add are not listed in the Navigator, you can use a Wizard to add this type. The preferences for the type of file that is added will then be listed among the other preferences under the Custom Project Settings. From here you can alter the scripts that executes when performing actions on files in PCM, see *Figure 10.5-4*.

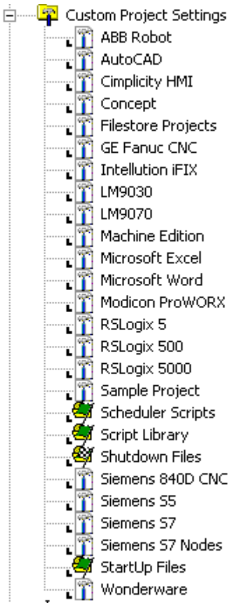


Figure 10.5-4: Custom Project Settings.

When right-clicking on the files you will only see the available commands for the selected file, see *Figure 10.5-5*. By selecting “History Report” the audit log will be displayed in the InfoViewer, see *Figure 10.5-6*.

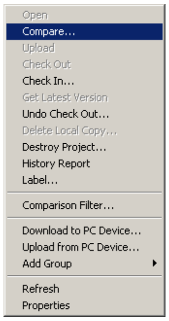
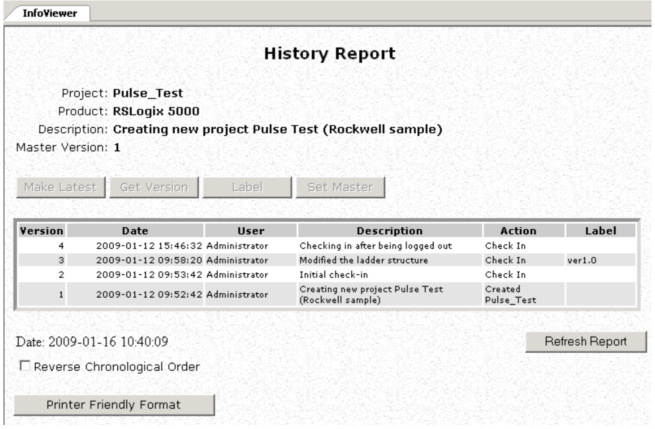


Figure 10.5-5: Right-click menu.



History Report

Project: **Pulse_Test**
Product: **RSLogix 5000**
Description: **Creating new project Pulse Test (Rockwell sample)**
Master Version: **1**

Version	Date	User	Description	Action	Label
4	2009-01-12 15:46:32	Administrator	Checking in after being logged out	Check In	
3	2009-01-12 09:58:20	Administrator	Modified the ladder structure	Check In	ver1.0
2	2009-01-12 09:53:42	Administrator	Initial check-in	Check In	
1	2009-01-12 09:52:42	Administrator	Creating new project Pulse Test (Rockwell sample)	Created Pulse_Test	

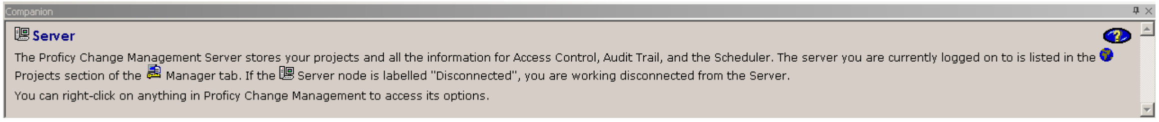
Date: 2009-01-16 10:40:09

☐ Reverse Chronological Order

Figure 10.5-6: The Audit log.

The log gives a good overview of the file history, lets you open up any revision of the file and gives you the option to label any revision with a version number for instance.

It is fairly simple to use PCM and in case there are any questions, the built-in help are quite thorough. To make the help easier to access you can choose to show the Companion pane, see *Figure 10.5-7*. This will display a basic description from the help chapter for the selected item.



Server

The Proficy Change Management Server stores your projects and all the information for Access Control, Audit Trail, and the Scheduler. The server you are currently logged on to is listed in the Projects section of the Manager tab. If the Server node is labelled "Disconnected", you are working disconnected from the Server.

You can right-click on anything in Proficy Change Management to access its options.

Figure 10.5-7: The Companion pane. In this case the Server icon in the Navigator is selected.

The basic use of PCM is straightforward and all commands are accessed from the right-click menu. Checking in and out files are easy and the checked out files will be marked with a green check mark to indicate that the file is ready to be edited, see *Figure 10.5-8.*

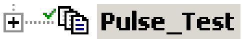


Figure 10.5-8: Checked out files are indicated with a green checkmark.

10.6 VersionWorks

When starting VersionWorks you should first switch between the Power mode and the Easy mode to get a feel for the program. The Easy mode limits the choices you have, see *Figure 10.6-1*. In this mode there are four icons and only the one that you can click are active and gets marked with a green checkmark when clicked. You can for instance not check in a file if you have not checked it out first. To access such things as the log or other information about a file, this is done through the right-click menu.

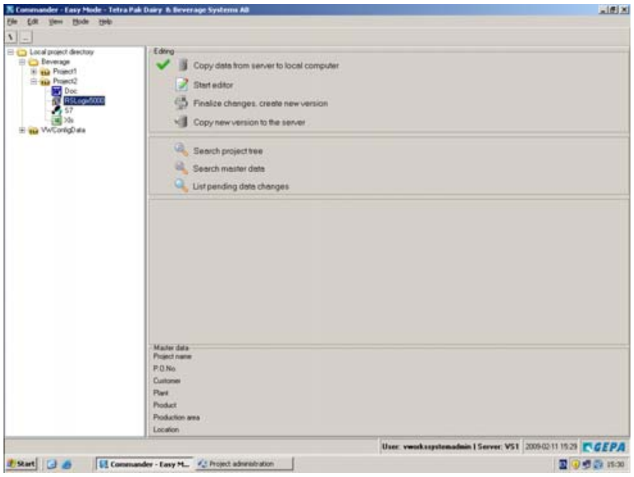
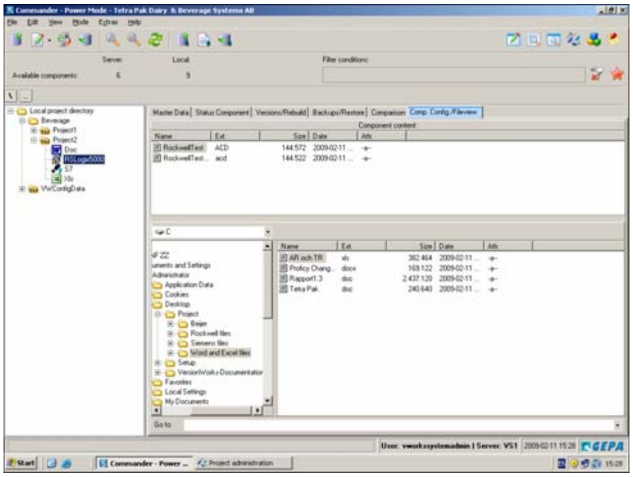


Figure 10.6-1: Easy mode.

The Power mode provides a more Windows Explorer-like layout, see *Figure 10.6-2*. The projects are, just as in the easy mode, listed in the left hand pane but the four clickable icons in the main window are now replaced with other information about the selected file. Actions you can perform on a file are listed in the right-click menu, see *Figure 10.6-3*.



Name	Ext.	Size	Date	Attr.
RockwellTest	ACD	144 572	2009-02-11 ...	-a-
RockwellTest...	acd	144 522	2009-02-11 ...	-a-

Name	Ext.	Size	Date	Attr.
ÅR och TR	xls	382 464	2009-02-11 ...	-a-
Proficy Chang...	docx	169 122	2009-02-11 ...	-a-
Rapport1.3	doc	2 437 120	2009-02-11 ...	-a-
Tetra Pak	doc	240 640	2009-02-11 ...	-a-

Figure 10.6-2: Power mode.

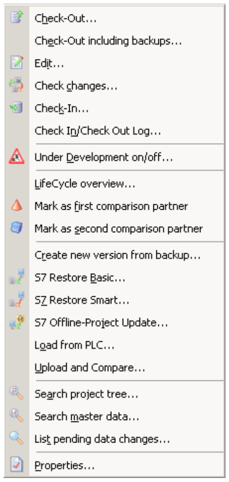
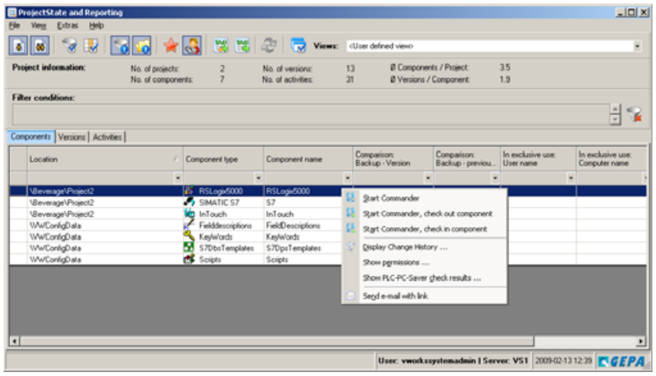

Check-Out…
Check-Out including backups…
Edit…
Check changes…
Check-In…
Check In/Check Out Log…
Under Development on/off…
LifeCycle overview…
Mark as first comparison partner
Mark as second comparison partner
Create new version from backup…
S7 Restore Basic…
S7 Restore Smart…
S7 Offline-Project Update…
Load from PLC…
Upload and Compare…
Search project tree…
Search master data…
List pending data changes…
Properties…

Figure 10.6-3: Right-click menu.

By selecting ProjectState you get access to a far more detailed audit log, see *Figure 10.6-4*. In this mode you can also create workflows to activate automatic email dispatches to people interested in changes being made to certain projects.



Project information:

No. of projects:	2	No. of versions:	13	Ø Components / Project:	3.5
No. of components:	7	No. of activities:	31	Ø Versions / Component:	1.9

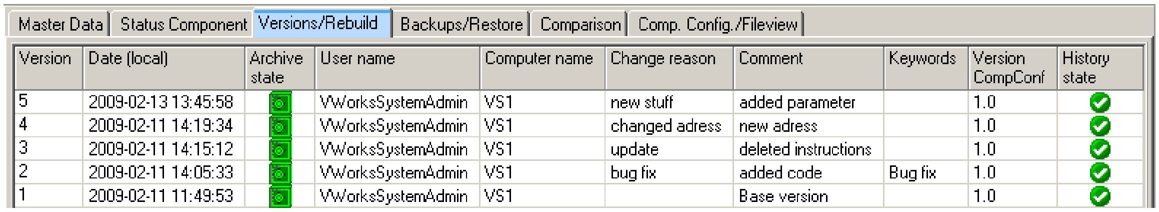
Filter conditions:

Location	Component type	Component name	Comparison: Backup - Version	Comparison: Backup - previou...	In exclusive use: User name	In exclusive use: Computer name
\Beverage\Project2	RSLogix5000	RSLogix5000				
\Beverage\Project2	SIMATIC S7	S7				
\Beverage\Project2	InTouch	InTouch				
\VVWConfigData	Fielddescriptions	FieldDescriptions				
\VVWConfigData	KeyWords	KeyWords				
\VVWConfigData	S7DbsTemplates	S7DpsTemplates				
\VVWConfigData	Scripts	Scripts				

User: vworkssystemadmin | Server: VS1 | 2009-02-13 12:39

Figure 10.6-4: The ProjectState mode.

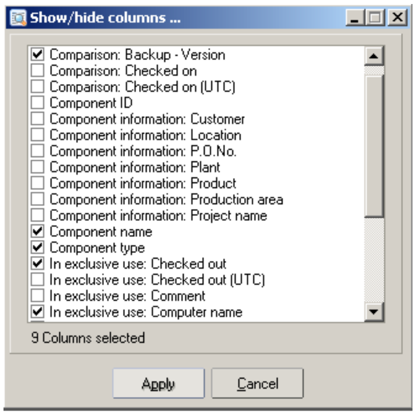
The audit trail in the Commander (see *Figure 10.6-5*) displays the version history of a file and to get more detailed information you simply open ProjectState and there you will find a lot more filter and display options (see *Figure 10.6-6*) for the log, for instance information about versions (see *Figure 10.6-7*) and an event log (see *Figure 10.6-8*).



Version	Date (local)	Archive state	User name	Computer name	Change reason	Comment	Keywords	Version CompConf	History state
5	2009-02-13 13:45:58	🟩	VWorksSystemAdmin	VS1	new stuff	added parameter		1.0	✓
4	2009-02-11 14:19:34	🟩	VWorksSystemAdmin	VS1	changed adress	new adress		1.0	✓
3	2009-02-11 14:15:12	🟩	VWorksSystemAdmin	VS1	update	deleted instructions		1.0	✓
2	2009-02-11 14:05:33	🟩	VWorksSystemAdmin	VS1	bug fix	added code	Bug fix	1.0	✓
1	2009-02-11 11:49:53	🟩	VWorksSystemAdmin	VS1		Base version		1.0	✓

Figure 10.6-5: The audit trail in the Commander.

The projects that are displayed in the ProjectState are filtered through the drop-down list in the top right corner of the window.

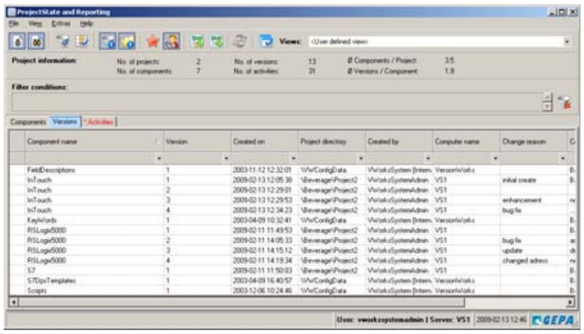


Show/hide columns ...

☑ Comparison: Backup - Version
☐ Comparison: Checked on
☐ Comparison: Checked on (UTC)
☐ Component ID
☐ Component information: Customer
☐ Component information: Location
☐ Component information: P.O.No.
☐ Component information: Plant
☐ Component information: Product
☐ Component information: Production area
☐ Component information: Project name
☑ Component name
☑ Component type
☑ In exclusive use: Checked out
☐ In exclusive use: Checked out (UTC)
☐ In exclusive use: Comment
☑ In exclusive use: Computer name

9 Columns selected

Figure 10.6-6: Display options for the log in ProjectState mode.



ProjectState and Reporting

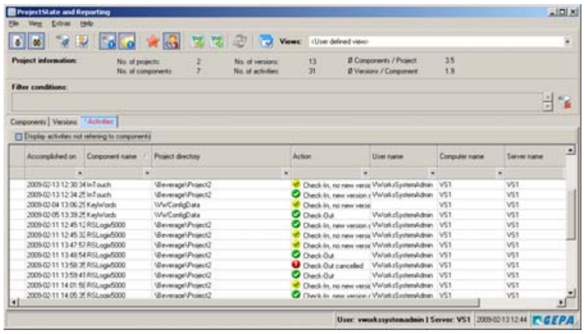
Project information:
No. of projects: 2
No. of components: 7
No. of versions: 13
No. of activities: 31
Ø Components / Project: 3.5
Ø Versions / Component: 1.9

Filter conditions:

Component name	Version	Created on	Project directory	Created by	Computer name	Change reason
FieldDescriptions	1	2003-11-12 12:32:01	\WwConfigData	\\WorksSystem (Intern.	VersionWorks	
InTouch	1	2009-02-13 12:05:38	\Beverage\Project2	\\WorksSystemAdmin	VS1	initial create
InTouch	2	2009-02-13 12:29:01	\Beverage\Project2	\\WorksSystemAdmin	VS1	
InTouch	3	2009-02-13 12:29:53	\Beverage\Project2	\\WorksSystemAdmin	VS1	enhancement
InTouch	4	2009-02-13 12:34:23	\Beverage\Project2	\\WorksSystemAdmin	VS1	bug fix
KeyWords	1	2003-04-09 10:32:41	\WwConfigData	\\WorksSystem (Intern.	VersionWorks	
RSLogix5000	1	2009-02-11 11:49:53	\Beverage\Project2	\\WorksSystemAdmin	VS1	
RSLogix5000	2	2009-02-11 14:05:33	\Beverage\Project2	\\WorksSystemAdmin	VS1	bug fix
RSLogix5000	3	2009-02-11 14:15:12	\Beverage\Project2	\\WorksSystemAdmin	VS1	update
RSLogix5000	4	2009-02-11 14:19:34	\Beverage\Project2	\\WorksSystemAdmin	VS1	changed adress
S7	1	2009-02-11 11:50:03	\Beverage\Project2	\\WorksSystemAdmin	VS1	
S7DpsTemplates	1	2003-04-09 16:40:57	\WwConfigData	\\WorksSystem (Intern.	VersionWorks	
Scripts	1	2003-12-06 10:24:46	\WwConfigData	\\WorksSystem (Intern.	VersionWorks	

User: vworkssystemadmin | Server: VS1 2009-02-13 12:46

Figure 10.6-7: Versions log.



ProjectState and Reporting

Project information:
No. of projects: 2
No. of components: 7
No. of versions: 13
No. of activities: 31
Ø Components / Project: 3.5
Ø Versions / Component: 1.9

Filter conditions:

☐ Display activities not referring to components

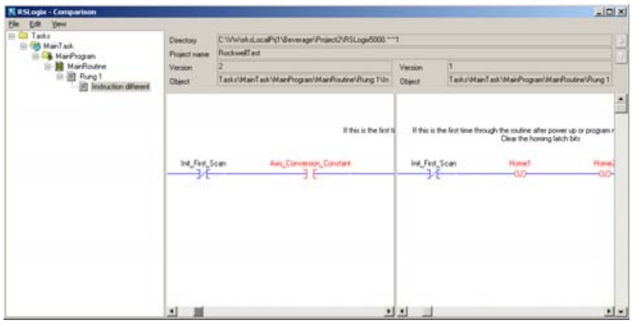
Accomplished on	Component name	Project directory	Action	User name	Computer name	Server name
2009-02-13 12:30:34	InTouch	\Beverage\Project2	✓ Check-In, no new versi	\\WorksSystemAdmin	VS1	VS1
2009-02-13 12:34:2	InTouch	\Beverage\Project2	✓ Check-In, new version	\\WorksSystemAdmin	VS1	VS1
2009-02-04 13:06:2	KeyWords	\WwConfigData	✓ Check-In, no new versi		VS1	VS1
2009-02-05 13:39:2	KeyWords	\WwConfigData	✓ Check-Out	\\WorksSystemAdmin	VS1	VS1
2009-02-11 12:45:1	RSLogix5000	\Beverage\Project2	✓ Check-In, new version	\\WorksSystemAdmin	VS1	VS1
2009-02-11 12:45:3	RSLogix5000	\Beverage\Project2	✓ Check-In, no new versi	\\WorksSystemAdmin	VS1	VS1
2009-02-11 13:47:5	RSLogix5000	\Beverage\Project2	✓ Check-In, no new versi	\\WorksSystemAdmin	VS1	VS1
2009-02-11 13:48:54	RSLogix5000	\Beverage\Project2	✓ Check-Out	\\WorksSystemAdmin	VS1	VS1
2009-02-11 13:58:3	RSLogix5000	\Beverage\Project2	● Check-Out cancelled	\\WorksSystemAdmin	VS1	VS1
2009-02-11 13:59:41	RSLogix5000	\Beverage\Project2	✓ Check-Out	\\WorksSystemAdmin	VS1	VS1
2009-02-11 14:01:5	RSLogix5000	\Beverage\Project2	✓ Check-In, no new versi	\\WorksSystemAdmin	VS1	VS1
2009-02-11 14:05:3	RSLogix5000	\Beverage\Project2	✓ Check-In, new version	\\WorksSystemAdmin	VS1	VS1

User: vworkssystemadmin | Server: VS1 2009-02-13 12:44

Figure 10.6-8: Event log.

To compare two different revisions you mark one of them as “first comparison partner”. Then you right-click on another revision and choose “Compare with selected version”.

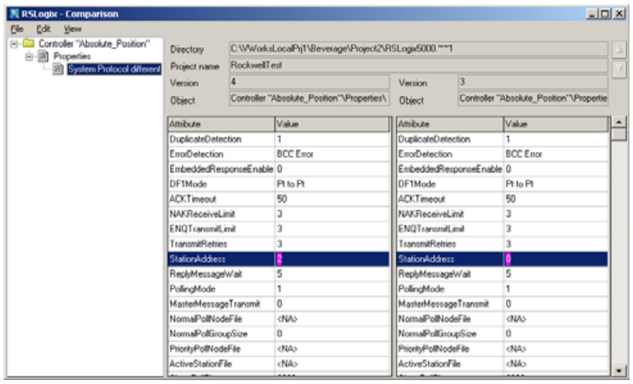
The compare result for RSLogix files is graphical and displayed in a split screen format, see *Figure 10.6-9*.



Directory: C:\WWorksLocalPrj1\Beverage\Project2\RSLogix5000.~~1
Project name: RockwellTest
Version: 2 | Version: 1
Object: Tasks\MainTask\MainProgram\MainRoutine\Rung 1\In | Object: Tasks\MainTask\MainProgram\MainRoutine\Rung 1

Figure 10.6-9: Compare result for RSLogix files.

The compare functionality also detected changes in the configuration of the project. In *Figure 10.6-10* a parameter has been changed and is indicated with a pink overlaying colour. The left pane displays the location of the file.

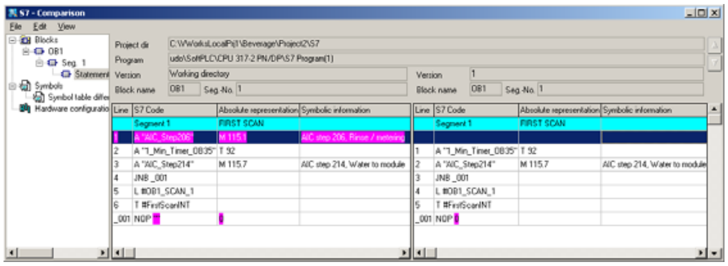


Directory: C:\WWorksLocalPrj1\Beverage\Project2\RSLogix5000.~~1
Project name: RockwellTest
Version: 4 | Version: 3
Object: Controller "Absolute_Position"\Properties\ | Object: Controller "Absolute_Position"\Propertie

Attribute	Value	Attribute	Value
DuplicateDetection	1	DuplicateDetection	1
ErrorDetection	BCC Error	ErrorDetection	BCC Error
EmbeddedResponseEnable	0	EmbeddedResponseEnable	0
DF1Mode	Pt to Pt	DF1Mode	Pt to Pt
ACKTimeout	50	ACKTimeout	50
NAKReceiveLimit	3	NAKReceiveLimit	3
ENQTransmitLimit	3	ENQTransmitLimit	3
TransmitRetries	3	TransmitRetries	3
StationAddress	2	StationAddress	0
ReplyMessageWait	5	ReplyMessageWait	5
PollingMode	1	PollingMode	1
MasterMessageTransmit	0	MasterMessageTransmit	0
NormalPollNodeFile	<NA>	NormalPollNodeFile	<NA>
NormalPollGroupSize	0	NormalPollGroupSize	0
PriorityPollNodeFile	<NA>	PriorityPollNodeFile	<NA>
ActiveStationFile	<NA>	ActiveStationFile	<NA>

Figure 10.6-10: A changed parameter is indicated with pink overlaying colour.

The compare for Simatic S7 files did not work as well as for RSLogix but it at least detected all changes that were done, see *Figure 10.6-11*, *Figure 10.6-12* and *Figure 10.6-13*. The graphical display did not work for Intouch files either, but but the changes were detected and displayed to some extent, see *Figure 10.6-14* and *Figure 10.6-15*.

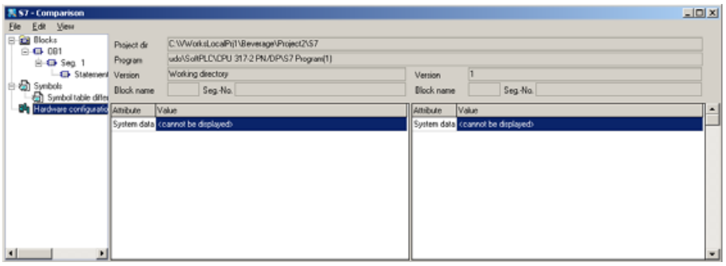


Project dir: C:\WWorksLocalPrj1\Beverage\Project2\S7
Program: udo\SoftPLC\CPU 317-2 PN/DP\S7 Program(1)
Version: Working directory | Version: 1
Block name: OB1 Seg.-No. 1 | Block name: OB1 Seg.-No. 1

Line	S7 Code	Absolute representation	Symbolic information
	Segment 1	FIRST SCAN	
1	A "AIC_Step206"	M 115.1	AIC step 206, Rinse / metering
2	A "T_Min_Timer_OB35"	T 92	
3	A "AIC_Step214"	M 115.7	AIC step 214, Water to module
4	JNB _001		
5	L #OB1_SCAN_1		
6	T #FirstScanINT		
_001	NOP ""	0	

Line	S7 Code	Absolute representation	Symbolic information
	Segment 1	FIRST SCAN	
1	A "T_Min_Timer_OB35"	T 92	
2	A "AIC_Step214"	M 115.7	AIC step 214, Water to module
3	JNB _001		
4	L #OB1_SCAN_1		
5	T #FirstScanINT		
_001	NOP 0		

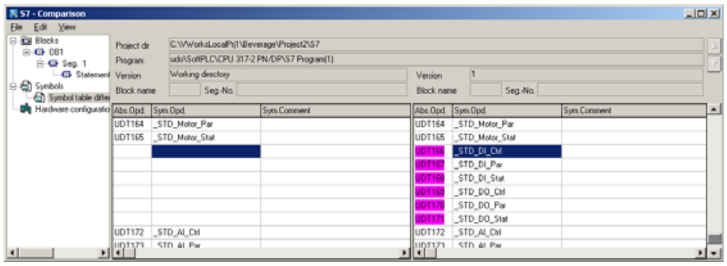
Figure 10.6-11: Changes to the ladder structure were displayed as text and it is unclear if it is possible to understand the exact changes more than where they were done.



Project dir: C:\VWorksLocalPrj1\Beverage\Project2\S7
Program: udo\SoftPLC\CPU 317-2 PN/DP\S7 Program(1)
Version: Working directory | Version: 1
Block name: | Seg.-No.: | Block name: | Seg.-No.:

Attribute	Value	Attribute	Value
System data	<cannot be displayed>	System data	<cannot be displayed>

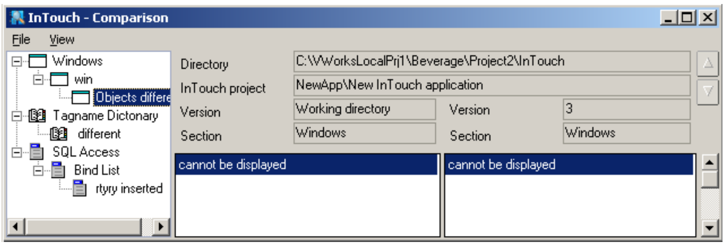
Figure 10.6-12: Changes to the hardware could not be displayed graphically but still tells the user that changes have been made.



Project dir: C:\VWorksLocalPrj1\Beverage\Project2\S7
Program: udo\SoftPLC\CPU 317-2 PN/DP\S7 Program(1)
Version: Working directory | Version: 1
Block name: | Seg.-No.: | Block name: | Seg.-No.:

Abs.Opd.	Sym.Opd.	Sym.Comment	Abs.Opd.	Sym.Opd.	Sym.Comment
UDT164	_STD_Motor_Par		UDT164	_STD_Motor_Par	
UDT165	_STD_Motor_Stat		UDT165	_STD_Motor_Stat	
			UDT166	_STD_DI_Ctrl	
			UDT167	_STD_DI_Par	
			UDT168	_STD_DI_Stat	
			UDT169	_STD_DO_Ctrl	
			UDT170	_STD_DO_Par	
			UDT171	_STD_DO_Stat	
UDT172	_STD_AI_Ctrl		UDT172	_STD_AI_Ctrl	
UDT173	_STD_AI_Par		UDT173	_STD_AI_Par	

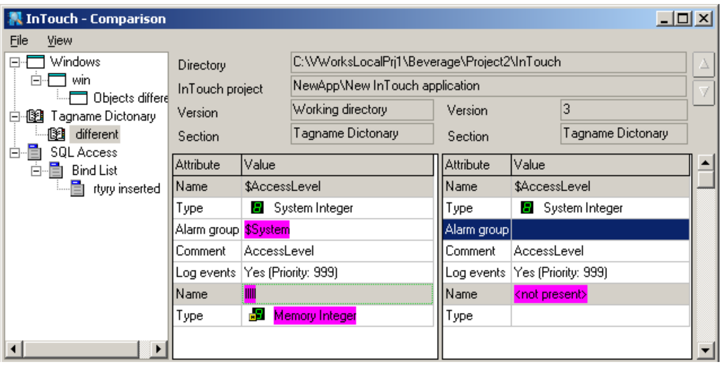
Figure 10.6-13: Altering of the symbol chart was detected and displayed flawlessly.



Directory: C:\VWorksLocalPrj1\Beverage\Project2\InTouch
InTouch project: NewApp\New InTouch application
Version: Working directory | Version: 3
Section: Windows | Section: Windows

cannot be displayed | cannot be displayed

Figure 10.6-14: Changes to windows in Intouch applications could not be displayed graphically.

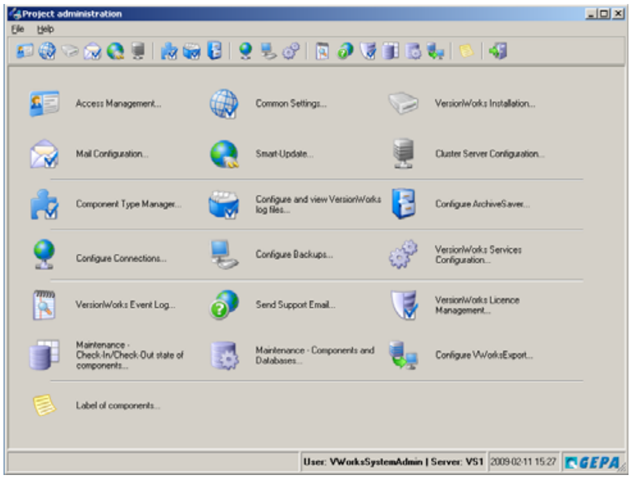


Directory: C:\VWorksLocalPrj1\Beverage\Project2\InTouch
InTouch project: NewApp\New InTouch application
Version: Working directory | Version: 3
Section: Tagname Dictonary | Section: Tagname Dictonary

Attribute	Value	Attribute	Value
Name	$AccessLevel	Name	$AccessLevel
Type	System Integer	Type	System Integer
Alarm group	$System	Alarm group	
Comment	AccessLevel	Comment	AccessLevel
Log events	Yes (Priority: 999)	Log events	Yes (Priority: 999)
Name	lllll	Name	<not present>
Type	Memory Integer	Type	

Figure 10.6-15: Modifications to parameters and configurations of Intouch applications were detected and displayed correctly.

The configuration of VersionWorks is done through the Project Administration, see *Figure 10.6-16*.


Project administration
File
Help
Access Management...
Common Settings...
VersionWorks Installation...
Mail Configuration...
Smart-Update...
Cluster Server Configuration...
Component Type Manager...
Configure and view VersionWorks log files...
Configure ArchiveSaver...
Configure Connections...
Configure Backups...
VersionWorks Services Configuration...
VersionWorks Event Log...
Send Support Email...
VersionWorks Licence Management...
Maintenance - Check-In/Check-Out state of components...
Maintenance - Components and Databases...
Configure VWorksExport...
Label of components...
GEPA

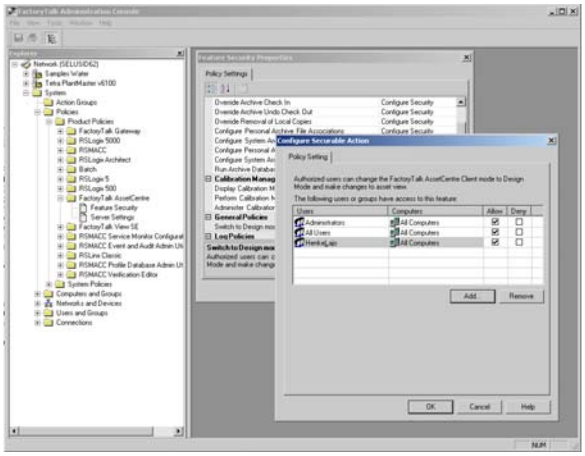
User: VWorksSystemAdmin | Server: V51 2009-02-11 15:27

Figure 10.6-16: Project Administration.

The project administration application lets the user configure permissions, mail, server backups, add new types of components that are not supported by default and much more.

10.7 AssetCentre

Asset Centre is similar to the layout of Proficy Change Management and Versionworks. The application runs on the server as a service and all configurations are made from the client computer. To add users, user groups and setting permissions you access the Administration Console, see *Figure 10.7-1*.



Override Archive Check In	Configure Security
Override Archive Undo Check Out	Configure Security
Override Removal of Local Copies	Configure Security
Configure Personal Archive File Associations	Configure Security

Authorized users can change the FactoryTalk AssetCentre Client mode to Design Mode and make changes to asset view.

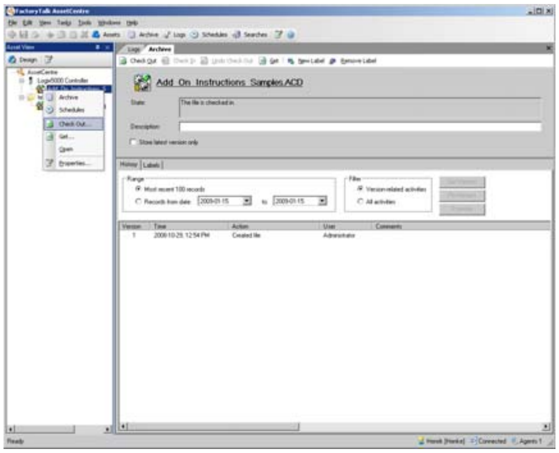
The following users or groups have access to this feature:

Users	Computers	Allow	Deny
Administrators	All Computers	☑	☐
All Users	All Computers	☑	☐
HenkeLaps	All Computers	☑	☐

Figure 10.7-1: Administration Console.

The administration console contains a lot of settings and lets you configure Asset Centre to run in a way that suits your needs.

The main window of the client application is easy to understand. The files under version control are organized in a tree structure in the left hand pane, which is called Asset View, and all commands that can be performed are reached through the right-click menu, see *Figure 10.7-2*.



Add_On_Instructions_Samples.ACD

State: The file is checked in.

Description:

□ Show latest version only

Range: ● Most recent 100 records ○ Records from date 2009-01-15 to 2009-01-15

Filter: ● Version related activities ○ All activities

Version	Time	Action	User	Comments
1	2008-10-29, 12:54 PM	Created file	Administrator	

Figure 10.7-2: The main window.

To add files you have to switch to Design mode by pressing the Design button in the Asset View pane, see *Figure 10.7-3*. This will provide you with more options such as adding, deleting and locking files, see *Figure 10.7-4*. The file formats that can be controlled with Asset Centre are shown in the dialog that appears when clicking the Add button (the green plus sign), see *Figure 10.7-5*.

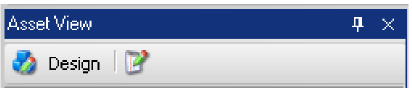


Figure 10.7-3: To add files the user must enter Design mode.

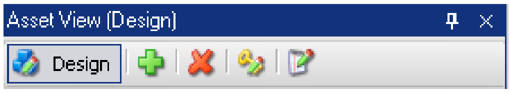
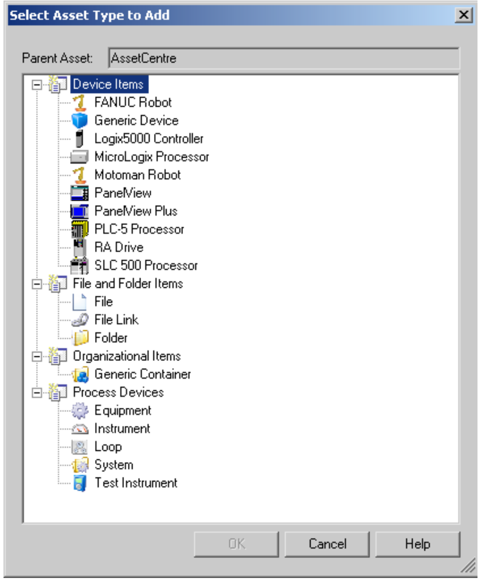


Figure 10.7-4: Design mode offers more choices for the user.



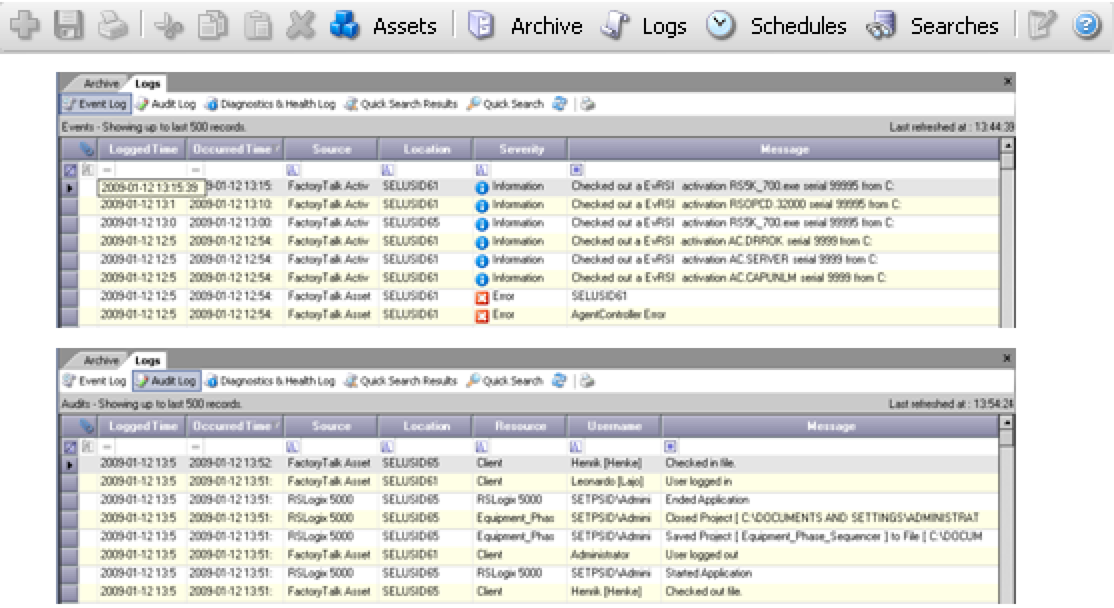
Select Asset Type to Add

Parent Asset: AssetCentre

- Device Items
  - FANUC Robot
  - Generic Device
  - Logix5000 Controller
  - MicroLogix Processor
  - Motoman Robot
  - PanelView
  - PanelView Plus
  - PLC-5 Processor
  - RA Drive
  - SLC 500 Processor
- File and Folder Items
  - File
  - File Link
  - Folder
- Organizational Items
  - Generic Container
- Process Devices
  - Equipment
  - Instrument
  - Loop
  - System
  - Test Instrument

Figure 10.7-5: Supported components in Asset Centre

The logs are accessed through the top menu and are divided into an event log and an audit log. The logs are customizable from the Administration Console and highly detailed, see *Figure 10.7-6.*



Events - Showing up to last 500 records.

Last refreshed at : 13:44:39

Logged Time	Occured Time	Source	Location	Severity	Message
2009-01-12 13:15:39	9-01-12 13:15:	FactoryTalk Activ	SELUSID61	Information	Checked out a EvRSI activation RS5K_700.exe serial 99995 from C:
2009-01-12 13:1	2009-01-12 13:10:	FactoryTalk Activ	SELUSID61	Information	Checked out a EvRSI activation RSOPCD.32000 serial 99995 from C:
2009-01-12 13:0	2009-01-12 13:00:	FactoryTalk Activ	SELUSID65	Information	Checked out a EvRSI activation RS5K_700.exe serial 99995 from C:
2009-01-12 12:5	2009-01-12 12:54:	FactoryTalk Activ	SELUSID61	Information	Checked out a EvRSI activation AC.DRROK serial 9999 from C:
2009-01-12 12:5	2009-01-12 12:54:	FactoryTalk Activ	SELUSID61	Information	Checked out a EvRSI activation AC.SERVER serial 9999 from C:
2009-01-12 12:5	2009-01-12 12:54:	FactoryTalk Activ	SELUSID61	Information	Checked out a EvRSI activation AC.CAPUNLM serial 9999 from C:
2009-01-12 12:5	2009-01-12 12:54:	FactoryTalk Asset	SELUSID61	Error	SELUSID61
2009-01-12 12:5	2009-01-12 12:54:	FactoryTalk Asset	SELUSID61	Error	AgentController Error

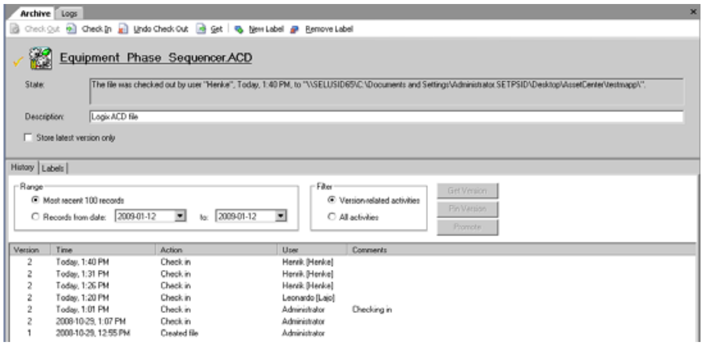
Audits - Showing up to last 500 records.

Last refreshed at : 13:54:24

Logged Time	Occured Time	Source	Location	Resource	Username	Message
2009-01-12 13:5	2009-01-12 13:52:	FactoryTalk Asset	SELUSID65	Client	Henrik [Henke]	Checked in file.
2009-01-12 13:5	2009-01-12 13:51:	FactoryTalk Asset	SELUSID61	Client	Leonardo [Lajo]	User logged in
2009-01-12 13:5	2009-01-12 13:51:	RSLogix 5000	SELUSID65	RSLogix 5000	SETPSID\Admini	Ended Application
2009-01-12 13:5	2009-01-12 13:51:	RSLogix 5000	SELUSID65	Equipment_Phas	SETPSID\Admini	Closed Project [C:\DOCUMENTS AND SETTINGS\ADMINISTRAT
2009-01-12 13:5	2009-01-12 13:51:	RSLogix 5000	SELUSID65	Equipment_Phas	SETPSID\Admini	Saved Project [Equipment_Phase_Sequencer] to File [C:\DOCUM
2009-01-12 13:5	2009-01-12 13:51:	FactoryTalk Asset	SELUSID61	Client	Administrator	User logged out
2009-01-12 13:5	2009-01-12 13:51:	RSLogix 5000	SELUSID65	RSLogix 5000	SETPSID\Admini	Started Application
2009-01-12 13:5	2009-01-12 13:51:	FactoryTalk Asset	SELUSID65	Client	Henrik [Henke]	Checked out file.

Figure 10.7-6: The top image shows the menu from which you navigate through Asset Centre. The middle image shows the event log and the bottom image shows the audit log.

To ensure the user that a file is checked out and ready to be edited the main window contains a description of the file and displays a yellow check mark beside it when it is selected in the Asset View pane. The log audit trail of the selected file will also be shown, see *Figure 10.7-7.*



Equipment_Phase_Sequencer.ACD

State: The file was checked out by user "Henke", Today, 1:40 PM, to "\\SELUSID65\C:\Documents and Settings\Administrator.SETPSID\Desktop\AssetCenter\testmapp\".

Description: Logix ACD file

□ Store latest version only

Range
● Most recent 100 records
○ Records from date: 2009-01-12 to: 2009-01-12

Filter
● Version-related activities
○ All activities

Version	Time	Action	User	Comments
2	Today, 1:40 PM	Check in	Henrik [Henke]	
2	Today, 1:31 PM	Check in	Henrik [Henke]	
2	Today, 1:26 PM	Check in	Henrik [Henke]	
2	Today, 1:20 PM	Check in	Leonardo [Lajo]	
2	Today, 1:01 PM	Check in	Administrator	Checking in
2	2008-10-29, 1:07 PM	Check in	Administrator	
1	2008-10-29, 12:55 PM	Created file	Administrator	

Figure 10.7-7: Yellow checkmark to the left of the file icon indicates that the file is checked out. The audit trail of the file is located in the bottom pane.

Basic use of Asset Centre is easy and the program is pretty self-explanatory and answers to questions that might turn up are well explained in the Help section of the application.

10.8 SharePoint

When using SharePoint you access the server through the web browser. The basic steps for the usage are: (The steps only reflect the version control aspect of the applications. These are extremely basic steps in regards to the possibilities and features that SharePoint offers.)

1. Creating workspace/folders on the server
2. Adding files to workspace/folders
3. Checking-in/-out files

1. The first step is to create a folder or a location on the server for storing files. To create this you use the action “Create” in the “Site Actions” menu in the upper right, see *Figure 10.8-1*.

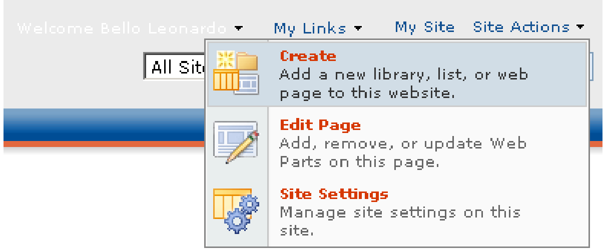
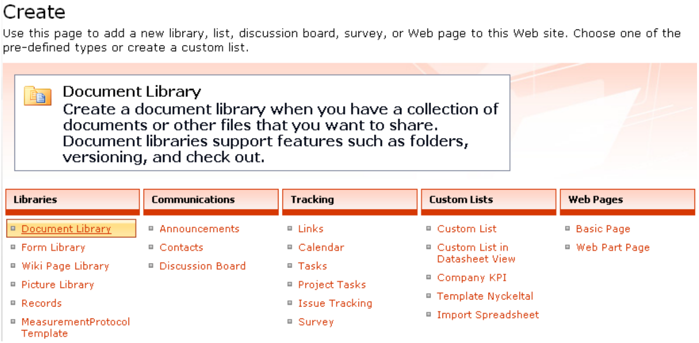

Welcome Bello Leonardo ▾
My Links ▾
My Site
Site Actions ▾
All Sit
Create
Add a new library, list, or web page to this website.
Edit Page
Add, remove, or update Web Parts on this page.
Site Settings
Manage site settings on this site.

Figure 10.8-1: Site Actions menu.

The next step is to choose what you want to create, in our case we choose Document Library, see *Figure 10.8-2*.



Create

Use this page to add a new library, list, discussion board, survey, or Web page to this Web site. Choose one of the pre-defined types or create a custom list.

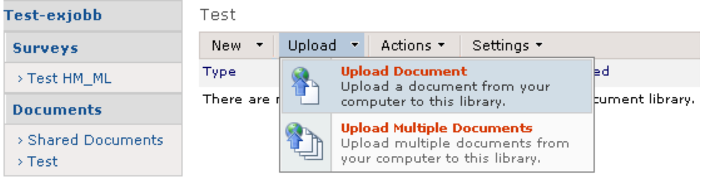
Document Library
Create a document library when you have a collection of documents or other files that you want to share. Document libraries support features such as folders, versioning, and check out.

Libraries	Communications	Tracking	Custom Lists	Web Pages
Document Library	Announcements	Links	Custom List	Basic Page
Form Library	Contacts	Calendar	Custom List in Datasheet View	Web Part Page
Wiki Page Library	Discussion Board	Tasks	Company KPI	
Picture Library		Project Tasks	Template Nyckeltal	
Records		Issue Tracking	Import Spreadsheet	
MeasurementProtocol Template		Survey		

Figure 10.8-2: Creation of a document library.

Your are then asked to specify some basic setting for the folder such as folder name, description, navigation, document version history and document template. When this is filled in the folder will be created on the server. Now it will be accessible in the left navigation field, see *Figure 10.8-3*. In step 2 the name of our folder is Test. (The folders and other document and features can later be added to the main page with the “Edit Page” action in the “Site Actions” menu).

2. Uploading files into the folder is done by navigating to the folder that you want to add files to. Inside the folder open the click the “Upload” tab and choose to either upload one file or several files, *Figure 10.8-3.*

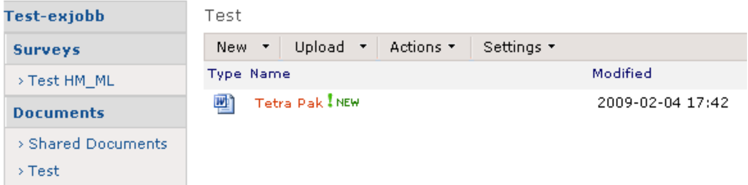


Upload Document
Upload a document from your computer to this library.

Upload Multiple Documents
Upload multiple documents from your computer to this library.

Figure 10.8-3: Navigation field to the left and uploading of files to the right.

After choosing “Upload” you browse to the location of the file/files that shall be added to the SharePoint server. The uploaded file will then appear in the folder, see *Figure 10.8-4.* The file (Tetra Pak.doc) is now accessible for check-out and shared.



Type	Name	Modified
	Tetra Pak ! NEW	2009-02-04 17:42

Figure 10.8-4: The file is now located in the "Test" folder.

3. The final step is to check-out and later check-in files. To check-out a file you use the Check Out command in the drop down window for the file that you wish check-out, see *Figure 10.8-5.*

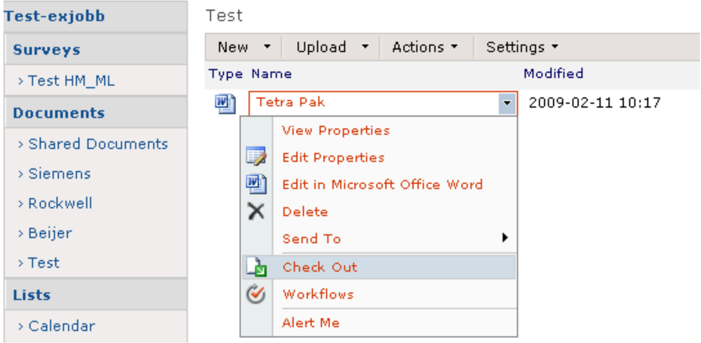


Figure 10.8-5: Check-out a file from the SharePoint server.

Doing a check-in of a file is done in the same way but instead using the Check In action, *Figure 10.8-6.* When working with for example a MS Word document the check-in can be made directly from within MS Word.

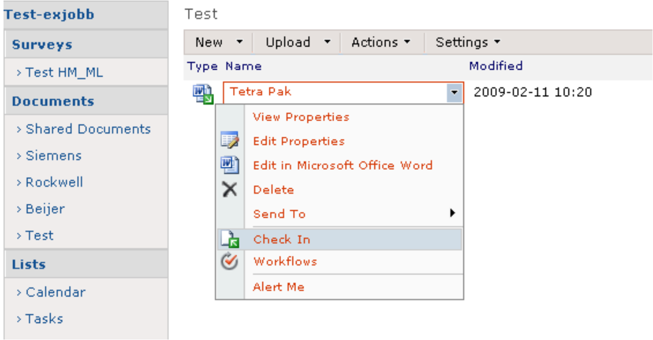
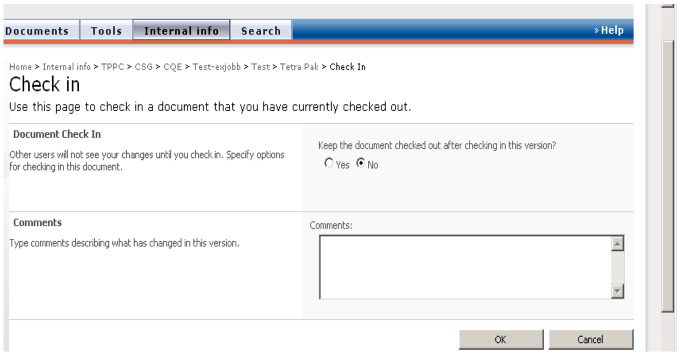


Figure 10.8-6: Check-in of a file.

In the next window you will be asked to do make a comment in regard to the check-in, see *Figure 10.8-7.*



Home > Internal info > TPPC > CSG > CQE > Test-exjobb > Test > Tetra Pak > Check In

Check in

Use this page to check in a document that you have currently checked out.

Document Check In

Other users will not see your changes until you check in. Specify options for checking in this document.

Keep the document checked out after checking in this version?

○ Yes ● No

Comments

Type comments describing what has changed in this version.

Comments:

Figure 10.8-7: Check-in comment that will be displayed in the audit log.

11 Appendix 2: Survey

AR and TR survey about Configuration Management (CM)

This question sheet is handed out to AR’s and TR’s within Tetra Pak. The survey will take about **15 minutes** to fill out, it’s anonymous and the result of your answers will be used as input to our report. This survey is very important to us and will hopefully make it possible to ratiocinate about the overall CM within the company. The goal is to come up with suggestions for improving today’s CM routines. Feel free to answer in Swedish and make adjustments in the flowcharts.

If you have any questions you can contact Henrik, Henrik.Israelsson@tetrapak.com or Leonardo, Leonardo.Bello@tetrapak.com.

Short description of Configuration Management
The way of handling files, naming files, tracking changes to files, reporting and recording the status of all versions of objects, quality assurance, keeping an audit trail of all actions, events and notifications that happen to all objects under CM control etc.

1. What is your title? How long have you worked here? Any previous experience from this kind of work?

   ____

   ____

   ____

2. Do you think the flowcharts a representative of how you work?

   Yes □   No □   Uncertain / Don’t know □

   Is anything wrong? Is anything missing, feedback or improvements for instance?

   ____

   ____

   ____

   ____

3. I am familiar with CM (except from the short description on the first page).

Disagree	Partially Agree	Agree
□	□	□

Elaborate:

4. I think that CM in general is important for our company.

Disagree	Partially Agree	Agree	Uncertain / Don’t know
□	□	□	□

5. I use CM in my work.

Yes	No
□	□

If yes, do you think CM plays an important role in your work?

Yes	No	Uncertain / Don’t know
□	□	□

6. Do you have any official CM-routines?

Yes	No	Uncertain / Don’t know
□	□	□

If yes, do you work accordingly to it?

Yes	No
□	□

If no, why not?

____

____

____

7. I feel that today’s CM-routines are working properly within BU DBF.

Disagree	Partially Agree	Agree	Uncertain / Don’t know
□	□	□	□

8. I would like to see changes in the CM-routines within BU DBF.

Yes	No	Uncertain / Don’t know
□	□	□

Elaborate:

____

____

____

9. Do you have anyone responsible for CM?

Yes	No	Uncertain / Don’t know
□	□	□

If yes, please define the name of the person:

____

10. Do you keep track of all variant releases?

Yes	No	Uncertain / Don’t know
□	□	□

Do you know which of the delivered machines that share the same code?

Yes	No	Uncertain / Don’t know
□	□	□

If no, can you easily find out?

Yes	No	Uncertain / Don’t know
□	□	□

11. I can handle the speed and volume with which RFC, feedback and improvements to my programs occurs.

Yes	No
□	□

12. I think that having specified release deadlines would be of help in my day to day work.

Yes	No	Uncertain / Don’t know
□	□	□

Why?

13. I document all changes to my programs.

Never	Rarely	Most of the times	Always
□	□	□	□

Is it easy for other people to read and understand the changes just by using this documentation?

Yes	No	Uncertain / Don’t know
□	□	□

14. I store my programs in an organized way.

Disagree	Partially Agree	Agree	Uncertain / Don’t know
□	□	□	□

It’s easy for other people to find these programs.

Yes	No	Uncertain / Don’t know
□	□	□

How do you name your files/folders? *(for example: filename_ver_XXX.XXX_XX or similar)*

15. I know who to contact or which documents to look at to find out how long it typically takes to make certain type of changes in my programs.

Disagree	Partially Agree	Agree	Uncertain / Don’t know
□	□	□	□

Elaborate:

16. I am reusing bug fixes across releases rather than wasting time fixing the same bug that was fixed in a previous release.

Never	Rarely	Most of the times	Always
□	□	□	□

If not always, how come?

17. I know for sure that everything that is released has gone through the necessary testing.

Disagree	Partially Agree	Agree	Uncertain / Don’t know
□	□	□	□

18. I can roll back a program to an earlier version to recover from a bug in a short amount of time.

Yes	No	Uncertain / Don’t know
□	□	□

19. I know which developer is editing code or content and why they are doing it.

Yes	No	Uncertain / Don’t know
□	□	□

20. I think that I work in a structured and organized way.

Disagree	Partially Agree	Agree	Uncertain / Don’t know
□	□	□	□

21. Which is the most common problem you encounter in your work with regard to CM? Is it in the process, the CM tool, the program, trouble finding files, etc?

22. Are you missing anything in your daily work when it comes to keeping track of versions and changes, looking in between products and categories?

Yes	No	Uncertain / Don’t know
□	□	□

If yes, what?

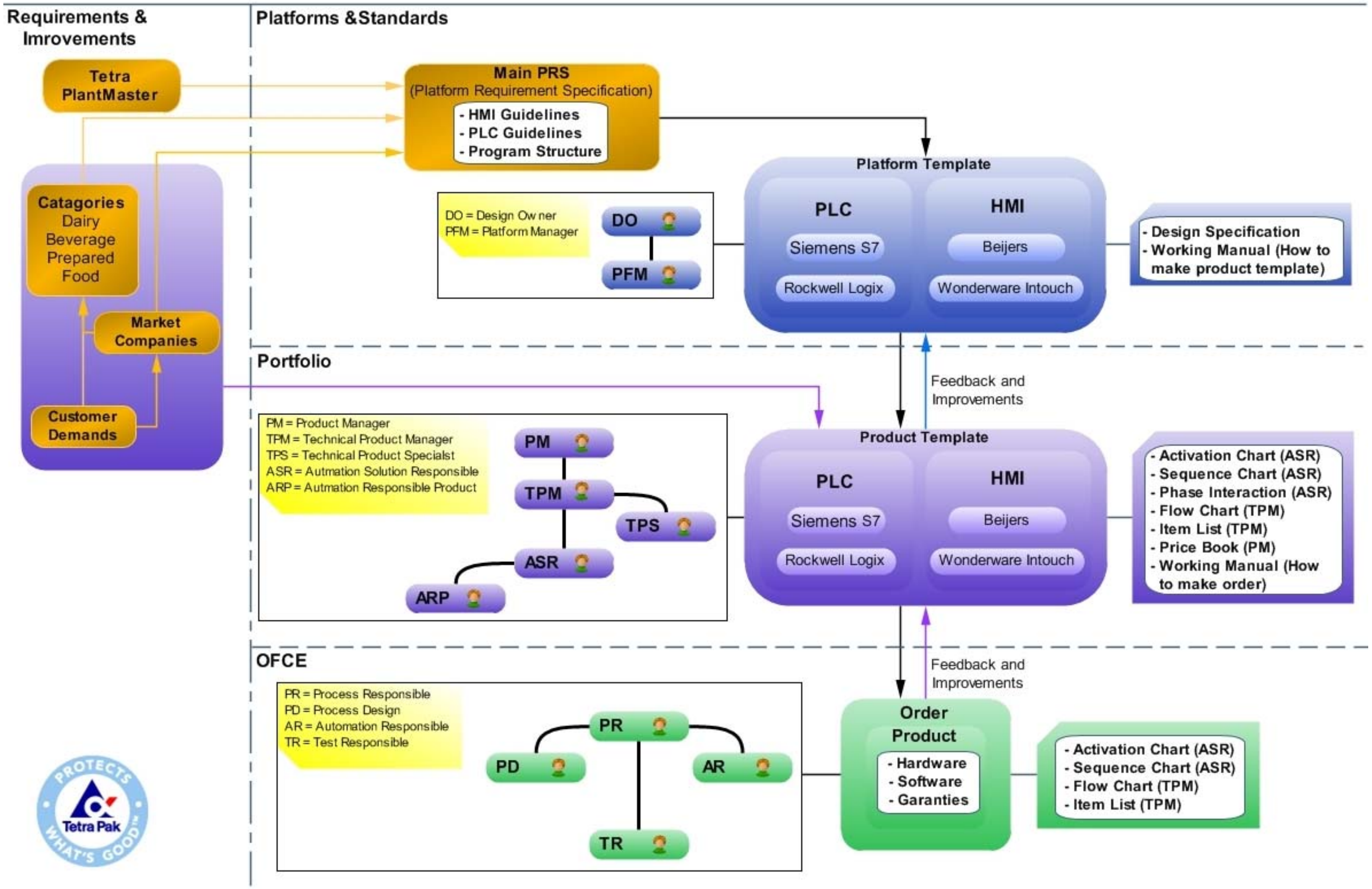
23. Are there any particular features you would like to have in a CM tool?

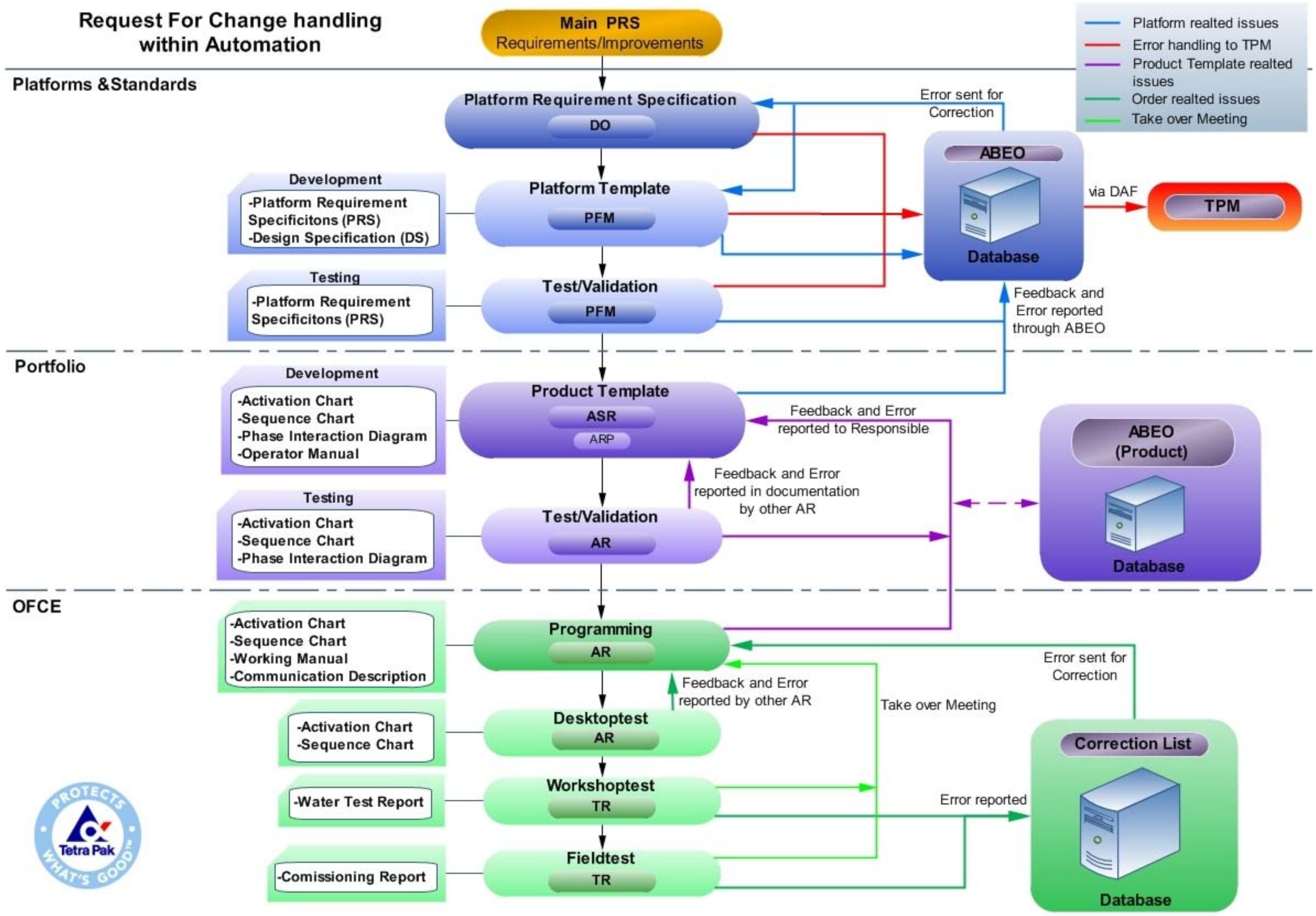
Yes	No	Uncertain / Don’t know
□	□	□

If yes, what?

Thank you for your time!

Ownership and availabilty of Documents
and Templates within Automation




Request For Change handling
within Automation
Main PRS
Requirements/Improvements
Platform realted issues
Error handling to TPM
Product Template realted issues
Order realted issues
Take over Meeting
Platforms &Standards
Platform Requirement Specification
DO
Error sent for Correction
ABEO
Database
via DAF
TPM
Development
-Platform Requirement Specificitons (PRS)
-Design Specification (DS)
Platform Template
PFM
Testing
-Platform Requirement Specificitons (PRS)
Test/Validation
PFM
Feedback and Error reported through ABEO
Portfolio
Development
-Activation Chart
-Sequence Chart
-Phase Interaction Diagram
-Operator Manual
Product Template
ASR
ARP
Feedback and Error reported to Responsible
ABEO (Product)
Database
Feedback and Error reported in documentation by other AR
Testing
-Activation Chart
-Sequence Chart
-Phase Interaction Diagram
Test/Validation
AR
OFCE
-Activation Chart
-Sequence Chart
-Working Manual
-Communication Description
Programming
AR
Error sent for Correction
Feedback and Error reported by other AR
Take over Meeting
-Activation Chart
-Sequence Chart
Desktoptest
AR
Correction List
-Water Test Report
Workshoptest
TR
Error reported
-Comissioning Report
Fieldtest
TR
Database
PROTECTS WHAT'S GOOD
Tetra Pak