Mälardalen University Press Licentiate Theses
No.58

PRODUCT DATA MANAGEMENT AND SOFTWARE CONFIGURATION MANAGEMENT INTEGRATION

Annita Persson Dahlqvist

2005

MÄLARDALEN UNIVERSITY

Department of Computer Science and Electronics
Mälardalen University

ISSN 1651-9256
ISBN 91-85485-02-0
Printed by Arkitektkopia, Västerås, Sweden
Distribution: Mälardalen University Press

To

Roger

ABSTRACT

Today, many products are built from both hardware and software components, this especially being true for technologically high-end and complex products such as cars, aircrafts, or mobile phones. Development of such products requires an integrated support that encompasses different domains such as electronics, mechanics, and software. Previously separated, the development of hardware and software components is becoming a common undivided process, which also requires integration of tools providing this support. One of the today's key factors for an integrated product information management is the possibility of integrated and uniform use of Product Data Management (PDM) and Software Configuration Management (SCM). These tools have similar purpose: providing an overall support for building and managing information infrastructure and collaboration between stakeholders. Yet, the integration of these tools and supporting processes has proven to be difficult and challenging for many companies.

The main objective of this thesis is to study the feasibility of achieving an integrated, consistent and efficient support to the complex products life cycles by PDM and SCM, either as separated or integrated tools. This objective has been achieved by several research activities: (i) analysis of the main characteristics of PDM and SCM, i.e. their key functionalities and relations between them, (ii) numerous industrial case studies of PDM and SCM usage and their interoperability, and (iii) a discussion of the hypothesis that the three factors, namely, technologies and architectures, processes, and stakeholders' cultural differences, are of a vital importance for a successful integration.

As a result the thesis provides a basis for thorough understanding of PDM and SCM and the prerequisites for their successful integration. The research is primarily based on literature studies, industrial case studies, and long experience from work in industry.

ACKNOWLEDGMENTS

The research presented in this thesis was carried out and financed by Swedish Foundation for Strategic Research and Ericsson AB. My interest in the area started when trying to set requirements for Ericsson on integration between one specific PDM and one specific SCM tool in close cooperation with one of the vendors. Without this struggling and meeting problems in finding a solution, I would never been interested in the topic and never understood the real problems for other companies. Without the possibility to be an appointed senior specialist in PDM, CM, and SCM at my Business Unit, I would not have the opportunity to perform this research work. I sincerely want to thank my business unit manager Sivert Bergman for appointing me to a senior specialist, my product development unit manager, Ola Gustafsson, for his support and interest in the research area. Further, I want to thank Annika Bergbom, Human Resources, for her encouragement and advices to manage my research and ordinary work concurrently. I also want to thank my former department manager, Anders Johansson, my department manager Jan Olson, and my closest manager Göran Humleby, in their effort to free time for my research.

I sincerely thank my supervisor Prof. Ivica Crnkovic for great support, long discussions, and important encouragement. I really appreciate Ivica’s effort in motivating me to prepare and do this thesis. My thanks to Mälardalens Högskola for letting me do this work.

Special thanks I want to give to my beloved husband Roger, for all support especially providing me with all the espressos, and help during long weekends work with my thesis. I also want to thank my son Fredrik, my stepson Kaspar and my stepdaughter Emma for their encouragement during this period.

Annita Persson Dahlqvist
Mölndal/Västerås 2005

LIST OF PUBLICATIONS

Peer reviewed papers and articles

The following peer-reviewed papers and articles have been published at international conferences.

Papers included in the thesis

The following are included in thie thesis.

A. ***Managing Complex Systems – Challenges for PDM and SCM in Software Configuration Management***
In Proceedings of the SCM10, 23rd ICSE, Toronto, Canada, May 2001.
Authors: **Annita Persson Dahlqvist**, Ivica Crnkovic, Magnus Larsson

B. ***Quality Improvements by Integrating Development Processes***
In 11th Asia-Pacific Software Engineering Conference, Busan, Korea, November 2004.
Authors: **Annita Persson Dahlqvist**, Ivica Crnkovic, Ulf Asklund

C. ***Important Factors for a Successful Integration of Product Data Management and Software Configuration Management Systems***
Technical Report, based on the book Implementing and Integrating Product Data Management and Software Configuration Management, [1]
Author: **Annita Persson Dahlqvist**

Other papers

The author has also co-authored the following articles, papers and one book:

D. ***The book: Implementing and Integrating Product Data Management and Software Configuration Managemen***t,
Artech House Published 2003 ISBN: 1-58053-498-8
Authors: Ivica Crnkovic, Ulf Asklund, **Annita Persson Dahlqvist**

E. ***Managing Complex Systems – Challenges for PDM and SCM***
Projects & Profits, IV-7 (Rs60):36-42, July 2004, ICFAI Press, Panjagutta, India,
Authors: **Annita Persson Dahlqvis**t, Ivica Crnkovic, Magnus Larsson

F. ***Complex Systems Development Requirements - PDM and SCM Integration***
IEEE Asia-Pacific Conference on Quality Software, IEEE, 2001.
Authors: Ivica Crnkovic, **Annita Persson Dahlqvist**, Daniel Svensson

G. ***Product Data Management and Software Configuration Management - Similarities and Differences***
The Association of Swedish Engineering Industries, V040096, 2001.
Authors: Ulf Asklund, Ivica Crnkovic, Allan Hedin, Magnus Larsson, **Annita Persson Dahlqvis**t, Johan Ranby, Daniel Svensson

H. ***Experiences of Customization and Introduction of a CM Tool***
Software Configuration Management SCM5, Seattle, USA, ICSE 1995.
Authors: **Annita Persson Dahlqvis**t

TABLE OF CONTENTS

1 INTRODUCTION

"Novice writers are big procrastinators. They find countless reasons not to get started. Even when they finally get themselves seated at their desks, they always seem to find diversions: make the coffee, sharpen the pencil, go to the bathroom, thumb through more literature, sometimes even get up and return to the field. Remember that you are never 'ready' to write; writing is something you must make a conscious decision to do and then discipline yourself to follow through."

Bogdan and Bilken, "Qualitative Research for Education:
An Introduction to Theory and Method",
Boston, MA: Allyn & Bacon, p 172

Many products consists of both hardware and software components. While this is today true for technologically high-end and complex products, e.g. cars, aircrafts, or mobile phones, this trend is expected in the near future to extend to almost all manufactured products, such as household appliances. Although, many of these products, can be seen as monolithic entities, they are complex since they consist of thousands of parts and components, both software and hardware. Mastering the interrelations between these parts is today a challenge, and is becoming increasingly for the success of any product on the market. This mastering is increasingly important as the number of parts in a product increases dramatically, and these parts are more and more developed independently of the product.

The more complex the product is, the more complex are the procedures for its development and support during its life. A new dimension of complexity comes with software inclusion in the products. The consequence for such products is that there is no pure hardware development; even the companies that develop hardware products must consider development of software. Consequently, for the product's life cycle, there is a need for a support for both hardware and software development and maintenance. The hardware and software domains have long been separated.

They have shared certain common concepts and techniques, but have been isolated from each other developing their own tools and using their own processes. Since the overlapping in functionality between software and hardware is increasing, the requirements on the tools support in respective domain increase. Consequently, the supporting tools are overlapping in functionality. The same is true for tools used for supporting development, production and maintenance: Product Data Management (PDM) and Software Configuration Management (SCM).

Such complex products, including hardware and software components, [2] Buur defines as mechatronic products. Buur [2] defines mechatronics as *"a technology, which combines mechanics with electronics and information technology to form both functional interaction and spatial integration in components, modules, products, and systems"*. We focus on the mechatronic products in our work.

PDM is used for managing product information, especially information used in the production phase. The computer tools managing this product data are called PDM systems.

A definition of PDM among a number of different definitions with the similar meaning is [3]:

> *PDM is the discipline of controlling the evolution of a product and providing other procedures and tools with the accurate product information at the right time in the right format during the entire product life cycle.*

PDM is an engineering discipline that includes different methods, standards, and tools. PDM (i) manages the data related to products, (ii) supports procedures during the product life cycle, and (iii) deals with the development and production infrastructure [1, 4, 5, 6]. Traditionally, PDM deals with hardware products and components only. PDM tools are integrated with computer-aided-design (CAD) and computer-aided-engineering (CAE) tools.

The development of complex and large software is characterized by collaboration and coordination of many developers. The objective of SCM is to ensure a systematic and traceable development process [7, 8]. Traditionally, SCM deals with software products and components only. SCM tools are integrated with the software environment and tools.

A definition of SCM is [9]:

> *Configuration management is the art of identifying, organizing, and controlling modifications to the software being built by a programming team.*

The scope of SCM is to (i) keep track of items and their versions, which are used in the product development and maintenance, (ii) manage all the changes made to these items during their entire life, and (iii) keep track of all documentation related to the product.

PDM and SCM have different histories and traditions [10, 11, 12]. Vendors drive the over-all development in the PDM domain even though there is research in several parts of the domain. Researchers drive the development on the SCM domain [7, 8, 11, 13]. PDM vendors and researchers have ignored software management in their development activities. Similarly, SCM vendors and researchers have until recently focused on supporting pure software development. Today, a trend in industry is to manage the entire product and not the hardware and the software parts separately. To obtain a better support for development and maintenance, companies have tried to integrate different supporting systems including PDM and SCM. However, there is a lack of knowledge in both disciplines to the adjacent domain, and an increased knowledge about the other discipline is required to achieve a thorough integrated support [7, 8, 12].

Consequently, we are faced with the need to understand and increase knowledge in both domains. Exhaustive research is needed to determine which integration and interaction methods are most suitable. This is the motivation for our research. The goal of the thesis is to:

- Analyze the similarities and differences between PDM and SCM.
- Investigate the abilities of the PDM and SCM tools to separately provide a full support for complex products development, and find the necessity of their integrations.
- Identify the most important requirements to be fulfilled for a successful integration of these tools.

1.1 Outline of the Thesis

The outline of the thesis is as follows. The first part of the thesis, section 1 to 6, contains the background and results of the research. Section 1 provides an introduction to the area. Section 2 gives a background and motivation for our research. Section 3 describes the selected research strategy used for the thesis, and the research questions. Section 4 lists the research results and contributions. Section 5 describes related work. Finally, section 6 formulates the conclusions and future work. The second part of the thesis consists of the included research papers.

2 RESEARCH MOTIVATION

"No problem can be solved from the same level of consciousness that created it."

Einstein

Traditionally, hardware development has been separated from software development. The development processes have been separated and different tools have been used to support these processes. Software products have been clearly separated from hardware products during development, and they have not been integrated before the start of system verification. Today, this border between hardware and software begins to vanish. The final product is a result of tight integration of hardware and software components and the decision whether a specific function should be implemented in hardware or software may come late in the development process and may even change during the product's life cycle. To achieve an efficient and complete support for the entire product life cycle, all included disciplines such as electronic, mechanical, and software domains have to be considered from the product, i.e. integrated, perspective. When the border between hardware and software becomes vague [14] for an efficient development support there is a need for an integrated support. Lack of integrated support can increase quality problems, cost increases and development lead-time overruns due to inefficient information flow and "manual" system integration. However, the requirements for such integration points out a number of challenges: process adjustments, information exchange, data access and seamless information flow, infrastructure support, tool integration, cultural differences, etc.

Several attempts to integrate tools from these domains exist, but they all show marginal visible success as discussed in [1, 4, 15, 16, 17]. The reason for this is that integration goes far beyond tool integration issues only.

In the process of the support integration many companies have chosen to use both PDM and SCM for managing the products and the components during the entire product life cycle. On the system level, where hardware and software components are integrated into the final product, the goal is to control the product for the entire product life cycle. To effectively manage a complex system on the system level, coordination of all included processes is needed [5, 16]. Different stakeholders, such as project manager, system engineers, integration and verification teams, and configuration managers, are all requested to follow up the development of the entire product of both hardware and software components. On component levels, the stakeholders want to have efficient and specialized tools that include specific support for development of components, and such support can be quite different, especially different in software development and hardware development.

PDM and SCM have similar support and similar purpose: providing an overall support for building and managing information infrastructure and collaboration between stakeholders. However, PDM and SCM also have fundamental differences in their visions, their assumptions, and their underlying technology. The PDM and SCM communities (including users, researchers and vendors) have developed not only different tools but also cultures. These differences can be traced back to significant differences in the way hardware and software products are developed, for example concerning version evolutions and product structuring. The real problem arises on the system level where hardware and software subsystems/components are integrated. At the system level, technology-independent product representations and working processes are essential for achieving a properly functioning product.

Since PDM and SCM systems have evolved in different development domains and have varying degrees of maturity, they have been developed on different technologies fulfilling the domains functionality demands and processes. The tools in respective area are different from a technical point of view. Consequently, a state of practice today in achieving an integrated support is the information integration, in a rather inefficient and cumbersome process: Product information in one system, either the PDM or the SCM, has to manually be introduced into the other system. The risk with this is either that the data is incidentally wrongly introduced or never performed [4].

Because of similarities in PDM and SCM functions, and because of overlap of information stored in them, there is a question if only one of these tools can be used for all development and maintenance processes, for both software and hardware parts? If not, would a seamless integration of these tools enable an integrated support for the development and maintenance process?

As main responsible for configuration management (CM) in several projects developing complex products, I received long (more than 20 years) work experiences from both domains. In discussions with practitioners and from their observations, I have observed on one side integration of hardware and software on the other side struggling with achieving this integration: (i) hardware and software processes were different and no integration points were defined, (ii) hardware related software, such as field programmable gate array (FPGA) or application-specific integrated circuit (ASIC) code, were managed by hardware developers as hardware components, (iii) hardware and software developers were seldom discussing common problems due to no common terminology and a lack of understanding of each other's domain and tools, (iv) integration and verification of products were time consuming due to inadequate support from tools, and (v) cumbersome and error-prone manual transfer of software product data to the PDM system were performed.

The goal of this thesis is to provide a better knowledge about PDM, SCM and relations between them and in this way provide a basis for their integration.

3 RESEARCH APPROACH

"Things should be made as simple as possible, but not any simpler."

Einstein

This section discusses the selected research design and strategy used with focus on investigations and case studies. In addition, the section contains a description of the case study set up and performance. The research questions and the approach taken to investigate each of them are described.

3.1 Selected Research Design and Strategy

In [18], Robson defines the research design as turning research questions into projects. Design concerns the various activities which should be planned for, when carrying out a research project. The used framework for our research design, based on [18], is shown in figure 1.

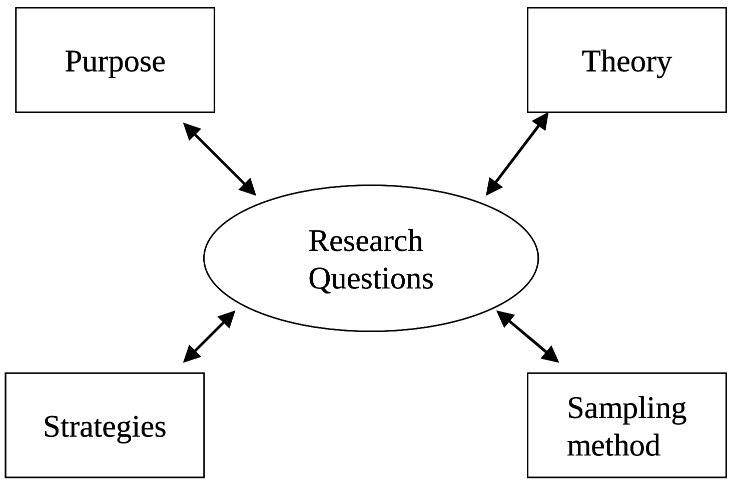

Figure 1 Framework for research design (Based on [18])

In *purpose*, the study's anticipated achievement is described. The purpose of our research was to produce *observations* and *findings* [19] responding to *what* the differences between PDM and SCM domains are, and *why* all of the three integration factors (which we have found), technology and tool, processes, and culture differences and people behavior are important when integrating PDM and SCM tools. The *rules-of-thumb* [19] will generalize the findings and observations in a larger domain than tested (the tested domain covers several companies, but there are indications to be valid for many companies developing and maintaining complex products).

The *theory* will guide our study. In our research project we have used the theory that all companies managing complex products will achieve a more efficient product development and maintenance when integrating PDM and SCM.

The purpose and the theory form the background and input in formulizing the *research questions*. Our research questions are to be found in section 3.2.

Strategy defines the specific technique used for collecting data. In our research we have divided the strategy into two parts; the first part has been performed by an investigation and description of similarities and differences between PDM and SCM, and the second part was achieved by case studies.

In *sampling method* the description on where and from whom data will be collected is described. For the investigation and description part, data was collected from literatures, discussions, and involvement in building an artifact. We have performed a number of case studies at different industries, both internationally and within Sweden, representing different business segments.

When using flexible or qualitative designs [18], there is a repeated revisiting of all the aspects as the research takes place, represented with double arrows in figure 1.

3.1.1 Investigation Strategy

The first part focuses on investigations of the two domains. The selected strategy for this first part of the thesis consists of three steps.

Firstly, the focus has been on understanding the domain in general, e.g. who is using the domain and for what purposes, what different hardware and software development processes are performed, which technological baseline is used, what different business models are used, what is the basic functionality, and which are the information models and system architectures. This part of the thesis has been performed by collecting information from literature such as tool manuals, journal papers, books, and research papers, describing the PDM, SCM and related domains and tools. We have performed investigations on key functionality in PDM and SCM, analyzed and categorized the similarities and differences in key functionality.

Secondly, discussions and open interviews with researchers, vendors, and tool users have been performed to get better understanding of the practitioners' needs, and researchers' questions and findings.

Thirdly, we have been involved in development of a physical artifact, i.e. interface between specific PDM and SCM tools. During the development, we had discussions with different stakeholders, such as developers, configuration managers, and line managers, which have increased our understanding on how the business processes, infrastructures and overall support and performance is related not only to the available tools but also to people's culture.

3.1.2 Case Study Strategy

For the case studies we have adopted strategies from [20, 21]. The first step is to describe the *problem*. The second step is to prepare the *case study design*, which is the link between the initial questions of the study to the data to be collected. Based on the design, *data collection* and *data analysis* are performed. Finally, the case study is *reported*. The different steps are shown in figure 2.

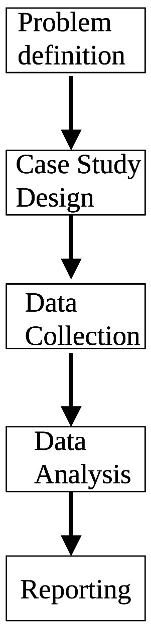

Figure 2 Case Study Research (Based on [21])

Since a large part of our research was performed in industrial settings we have decided to adopt principles from *explorative case studies* [18]. The performed case studies were focused on how practitioners use PDM and SCM, and why it is important to consider processes, tools and technologies and cultural differences when integrating PDM and SCM.

The overall research strategy selected for this thesis consists of five steps, as shown in figure 2. Firstly, the problem is defined. Secondly, in the case study design phase, a number of questions were formulated based on existing models, knowledge, and theories. The questions were mostly open-ended questions grouped into several areas. They were used to provide a common structure to all interviews and to form a basis for comparison and analysis of the results. The purposive sampling method [18] were used to find out typically and interesting companies. Several companies, which business segments were relevant for our research, have been selected for case studies. Thirdly, in the data collection phase, the questions were sent to the selected companies to inform about the interview questions. The semi-structured type of interviews [18] were used and some of the order of the questions were changed during the interviews altogether with explanations and additional questions included. The collected data and related documents were archived for further analysis. Fourthly, in the data analysis phase, the results from the interviews together with direct observations were analyzed against the key functionality found in the investigation and description phase. Fifthly, in the reporting phase, the cases were reported in several papers and widely discussed in one book. The steps 3 to 5 have been reused for each performed case study with use of improved questions.

3.1.3 Case Study Setup and Performance

A team of researchers designed the original case study questions, used at the first case study. No pilot case has been performed.

We have investigated PDM and SCM usage in different companies. In addition, we have studied two different initiatives in integration, both integrating commercial tools. In our case studies we have used the following types of analysis (i) embedded analysis [21], i.e. multiple units of analysis and (ii) holistic units of analysis [21], i.e. single-unit of analysis.

We have sent out the questions to the companies in advance, where the case studies were supposed to be performed. The layout and the clustering of questions can be found in the interview guide, which is shown below:

<u>Business issues and product descriptions</u>

1. Describe very short the company (the company size, the business goals, the organization).
2. What types of products does the company develop and sell? Describe them shortly.
3. Describe shortly the products and its content in form of included software and/or hardware components.
4. Do you develop products for many customers or few customers?
5. Do you develop the product on order basis or do you have standard products?
6. Do your products contain embedded software or are they stand-alone software systems?
7. Do you support many/some/one major release of the customer product?

<u>Processes</u>

1. Describe the product life cycle process and included stages, from the requirement specifications, design, implementation, test (verification and validation), production, delivery, and maintenance.
2. Describe the software development process, if applicable, and how it is related to the product life cycle process, if applicable. Do you use different software development process for embedded and stand-alone software products?
3. Describe the hardware development process, if applicable, and how it is related to the product life cycle process.
4. At which stage/point in the product life cycle process are the hardware and software processes separated?

5. At which stage/point in the product life cycle process are hardware and software components integrated? Are they integrated at different points? E.g. integration of embedded software with hardware at different test levels, no integration for stand-alone products except when manufacturing of a CD/DVD starts.
6. What kind of information is created during the product life cycle process?
7. At which stages in the product life cycle process is the different information created? E.g. Requirement specifications in the design phase, drawings during hardware development phase, source code during software development phase.
8. Do you use a specific document management process?
9. What kind of product data do you store for hardware products, if applicable?
10. What kind of product data do you store for software products, if applicable?
11. Describe the maintenance process. How does the developers, both hardware and software, find the right product to change?
12. Do you use any kind of decision process where the different information is reviewed, frozen, and change management is formalized? Describe this decision process and its main components. How is this decisions related to the product life cycle process?

Tools and Technology

1. Which particular Product Data Management (PDM) tool(s) do you use? What are the main advantages of using a PDM system?
2. Which particular Software Development (SCM) tool(s) do you use? What are the main advantages of using an SCM system?
3. Are you using a particular Document Management (DM) tool? If you do, what are the reasons for using a DM tool? What are the main advantages of using a DM tool?
4. What kind of tools (PDM and SCM) are you using for your product development, production, and maintenance? E.g. in-house built systems or purchased systems.
5. Which tool(s) (SCM) are you using during the software development?
6. Which tool(s) (PDM) are you using during the hardware development?
7. Do you use separate tools for managing product data? What kind of product data is managed in the tools, e.g. documents, software load modules?
8. In which tools is information created and in which tools is information archived?
9. How is product data shared with production and manufacturing? Is information manually or automatically stored in a separate tool? Is information stored in the development tools only?
10. Do you use a tool for all product data gathered for one specific customer product?
11. What kind of information is stored in PDM, and what kind of information is stored in SCM? Do you save the same information in both tools?

12. During the maintenance phase, how did the developers know where to find the information about the product, which has to be changed? Do they have to search in different tools or archives?

13. Do you follow any specific international standard for SCM, CM, DM and PDM?

People and Culture

1. How is the company organized regarding software and hardware development? Are they organized in different organizations or in the same organization on the lowest levels?

2. If applicable, how much and on which level does software and hardware development teams/developers cooperate?

3. Do you see any problems and difficult from hardware developers and software developers in not understanding each other? Describe these culture problems.

4. Do you have educations on your product life cycle process?

5. Do you have specific software process educations for software developers only?

6. Do you have specific hardware process educations for hardware developers only?

Integration – Processes

1. Do you have a product life cycle process managing both hardware and software development? What are the benefits on such process integration?

2. If you have a product life cycle process integrated with both hardware and software development, where does these processes differ, and how? On what level are they integrated?

3. Do you have education on such integrated process for both hardware and software developers?
4. If you do not have an integrated hardware and software development process, do you have any requirement or need for such integration?

Integration – Tools

1. Do you exchange information between PDM and SCM? If you do, in which process state are you exchanging information? What kind of information are you exchanging?
2. Do you have a need for better interoperability between a PDM and SCM system? What are the needs? Which benefits would you achieve by the integration?
3. Do you have an automatic integration between PDM and SCM? How is this integration implemented? On a lower level with no automatic transferring of information or with fully automatic transfer of information. Describe the integration. What are the drivers behind this integration? If not fully integrated, are there any demands for a full integration? What are the benefits for a full or not a full integration?
4. Describe the most serious problems you have had with PDM, SCM, and DM tools.
5. Do you miss some functionality in these tools (PDM, SCM, and DM) to fully support your product life cycle process?

Integration – People and Culture

1. Do you use the same terminology within the company to avoid misunderstanding between hardware and software developers? Who is maintaining this company terminology? Is this company terminology

enough in the sense of managing terminology on all levels? Is it commonly used?

2. If you are using a PDM and an SCM tool, how have you solved the terminology problem from the tools themselves? Have you customized the tools for managing the company terminology? Does this customization fulfill the demands on understanding the two domains without serious problems?
3. Do you provide any kind of education for both hardware, and software developers to reduce misunderstanding of each other's domain?
4. Do you have a need for integrating hardware and software developers, i.e. to educate and train them in both domains for a better understanding of each other?

We have performed case studies at seven different companies from three countries. The companies have represented different business segments such as telecom, IT and services provider, hardware development tools provider, power and automation, enterprise systems, defense, and mobile phones. They were:

- Sun Microsystems Inc., USA;
- Mentor Graphics Corporation, United Kingdom;
- Industrial and Financial Systems, Sweden;
- Ericsson Radio Systems AB, Sweden;
- Ericsson Mobile Communications AB, Sweden;
- ABB Automation Technology Products, Sweden, and;
- SaabTech Electronics AB, Sweden.

The case studies have shown a large diversity in requirements, in current states and future goals related to support of the product's life cycle for complex products.

However the case studies showed that there are some common characteristics, which might indicate general trends in this area:

Observation 1: Most of the companies have a solution for integrated support, but this solution is not completely satisfactory, and there are plans or ongoing activities for improvements.

Observation 2: The complexity of the development processes and information flow is vast.

Observation 3: There is a trend in understanding the benefits of a developed fully functional integration between PDM and SCM.

Observation 4: Large companies have a well-described process for the products life cycle and supporting tools on a company level.

	Case 1	Case 2	Case 3	Case 4	Case 5	Case 6	Case 7
SCM tool(s)	Com-mercial	Freeware	Com-mercial	Com-mercial	Com-mercial	Freeware	Freeware
PDM tool(s)	In-house built	N/A	In-house built	In-house built	Com-mercial	In-house built	In-house built
Information Integration reached	Partly	No	No	No	No	No	No
Common company terminology	Yes	N/A	Yes	Yes	Yes	Yes	N/A
Need for tools integration	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Separate HW & SW processes	Yes	N/A	Yes	Yes	Yes	Yes	N/A
HW & SW separately organized	Yes	N/A	Yes	Yes	Yes	Yes	N/A
PLC management improvement needed	Yes	N/A	Yes	Yes	Yes	Yes	N/A

Some observations:

1. No information integration is reached, but a need for tools integration is required in all companies;

2. A common company terminology was found in all companies. The common terminology was defined on a company level, but no common understanding between hardware and software terminology were used or defined. Technical integration is missing. Since software and hardware developers are organized in separate organizations, understanding for each others domain is not facilitated. From this, we can conclude that there are culture differences between hardware and software stakeholders.

3.2 Research Hypothesis and Research Questions

We have formulated the following two hypotheses:

(H1) For an efficient product life cycle management, a seamless integration of PDM and SCM is needed.

(H2) For a successful integration of PDM and SCM three integration factors are necessary: successful architectural tool integration, process integration, and removal of cultural differences between PDM and SCM stakeholders.

To justify these hypotheses, we have investigated if it is possible to use PDM for system development including software development, i.e. is PDM sufficient. In addition, we have investigated if it is possible to use SCM for system development including product management, i.e. is SCM sufficient.

To find answers on the first hypothesis (H1), we state the following research questions:

(Q1) What are the similarities and differences between PDM and SCM?

(Q2) Are the underlying mechanisms so different that they cannot be used in the other domain?

(Q3) Can integration between PDM and SCM lead to a more efficient support during development and maintenance?

To validate the second hypothesis (H2), we state following research question:

(Q4) *Why have the three integration factors, process integration, tool integration, and culture differences and people behaviors, to be considered for providing a successful integration of PDM and SCM?*

To answer these questions, we have analyzed technologies, information models, architectures, and processes used in these two domains and their key similarities and differences.

4 RESEARCH RESULTS AND CONTRIBUTION

"If the facts don't fit the theory, change the facts."

Einstein

4.1 Research Results

The four research questions have been stated and here follows the answers to them described in form of research results:

(Q1) What are the similarities and differences between PDM and SCM?

In answering (Q1) we have analyzed the domains and found the most important and significant similarities and differences. The results of the analysis and study are presented in papers A, B, C and [1], analyzing and discussing them from the different perspectives. Here we give a short summary of the results described in the papers. We have found following ***similarities*** in PDM and SCM.

Both PDM and SCM have **version control management**. Versions in SCM form a hierarchical structure, in which two versions of a file can be developed simultaneously in branches. These branches may be merged together again if needed. In PDM systems revisions are organized in a flat structure (only one main branch as compared with the hierarchical structure in SCM). PDM has a simpler version management model compared to SCM.

Both PDM and SCM support **change management**. In PDM there are add-on modules, which support change management. In SCM there exists specific change management tools integrated with the SCM system. Due to of the nature of software, the change process and the traceability are better integrated with change management. For hardware products the change process itself is usually outside the scope of the change management tools.

Both systems support **release management**. Simple support for release management is available in SCM for pure software products (e.g. packaging of the executables, related documents, and installation programs). In PDM the support for release management is strong. The production engineer uses the BOM (Bill-Of-Material), the list of all parts included in the final product, to assemble the final product.

Both PDM and SCM support **workflow and process management**. Workflows and processes are standard parts of PDM: they can be defined and executed in PDM systems. All processes can be changed and adapted to specific projects. Some SCM tools incorporate similar functionality or provide it using tools tightly integrated within the 'tool suite'. Most SCM tools, however, do not provide this support directly, but the tools have a means for adding new functions (like workflow management).

Both manage **large amount of data**. The formats of data are, however, different.

Both PDM and SCM are used during the total **product life cycle** of hardware or software.

Following ***differences*** are found in PDM and SCM:

The **information models** in PDM and SCM are fundamentally different. PDM systematically distinguishes metadata (called business items) from actual data (called data items). Only a few of SCM tools distinguish metadata from actual data and not in the way PDM does. Most PDM tools use an object-oriented data model in which a hierarchy of different types of business items can be created with references to related data items. Many PDM tools also use an object-oriented relational database to store the business items, while the data items are stored separately in file systems. In SCM data is stored in file systems. In general, all kinds of file types and objects represented as a file or directory may be managed and stored in the SCM tool.

Metadata and data in PDM can be **distributed** separately or in a combination. In most SCM tools, the replication functionality is implemented as an add-on feature.

PDM uses **revisions** for major changes. The revisions are versions of business items. If a data item is associated with a business item, the changes of the data item are denoted versions. These versions are not visible for the user. SCM uses **versions** for all changes. SCM has **branches** and supports **merge** functionality. PDM does not. This implies that, in SCM several people can work on the same file concurrently using the branch facility, which is not possible in PDM.

Product structure management is a basic functionality and fundamental in PDM. The PDM system describes a configuration by arranging the parts in a structure consisting of different products or parts connected by relationships. The product structure commonly follows the same pattern as the structure of the physical product. SCM tools do not explicitly address and support product structures. Only rudimentary support, in the form of directories in a file system, is available for use in building a hierarchical structure.

Build management is vital functionality for building executables from source code in an automatic fashion. The most known building tools are different versions of Make facility. Build is in no way supported by PDM. PDM tools do not support **configuration/selection management**. SCM is strong on this.

Some PDM tools use industrial **standards**. One example on such standard is the Standard for the Exchange of Product Model Data (STEP), ISO 10 303. There is no standard for SCM tools, although there are standards for configuration management.

In an SCM system the user checks out all the files to be changed and stores these in his/her **workspace**. The SCM system registers all files checked-out; the version checked out, by whom, and in which workspace the copy is stored. If many users check out the same file, these check outs are coordinated under control of the tool in accordance with the synchronization model used. Each user can set up and change the definition (selection) of which files (versions) to be checked out to the workspace. Workspace management in this form is not provided in PDM systems. PDM systems have work locations (which are private file locations), with one location defined per user. In PDM, the user checks out one file at a time and update it. Locking prevents more than one user from checking out the same version of the same file simultaneously (it is usually only the latest version which is checked out).

(Q2) Are the underlying mechanisms so different that they cannot be used in the other domain?

In paper A, B, C, and [1] we state how software and hardware engineers are working and noting their demands for support during the development and maintenance phases. PDM and SCM have different demands for their users, e.g. a software designers need to be able to work concurrently on a single file, need support for building the system, need to be able to work in isolation but still under the tool control. Such support is not provided by PDM. Hence, PDM is not suitable for system development including software components development. Hardware developers have other requirements on the tool to use compared to software designers. They need e.g. support for product structures, document management, visualization, and collaboration with manufacturing. Since none of these demands are fulfilled in SCM, SCM is not suitable for system development including hardware components development.

From the analysis of basic characteristics of PDM and SCM tools we found that there are similarities in them, but the underlying concepts and structures are quite different. Some of the functions are similar with a similar implementation, some of the functions have similar goal and purpose, but implementations are very different, while some of the functions are unique for PDM or SCM.

(Q3) Can integration between PDM and SCM lead to a more efficient support during development and maintenance?

In paper A, B, and [1] we state that there is a demand for interoperability between PDM and SCM for more efficient management of product data. The conclusion reached in comparing SCM and PDM; that SCM tools do not have the necessary functionality to support the development of a complex product during its entire life cycle, and that PDM tools do not have sufficient functionality to support software management, particularly during the development phase. Even though there is much functionality redundancy, however, SCM tools and PDM tools complement each other. The products life cycle require support during the development, production, and maintenance phases. All the different stakeholders need support for their daily work, and are using tools adjusted for their activities. This implies that for the entire product life cycle support different tools with different purposes are needed. For an efficient development and maintenance, an overall integration of information is needed during the entire development and maintenance processes at all structure levels. Since there is a need for information exchange between the different stakeholders, and between tools, a seamless interoperability between tools is required. To achieve seamless information flow on system and subsystem levels; integration of the PDM and SCM tools are needed. Such integration will increase the efficiency for support to the hardware and software developers and other stakeholders. We discuss integration possibilities in form of loose integration, full integration, and no integration.

(Q4) Why have the three integration factors, process integration, tool integration, and culture differences and people behaviors, to be considered for providing a successful integration of PDM and SCM?

For Q4 we have performed case studies at different companies, where we have found initiatives in integrating PDM and SCM. Together with the people in the companies, we have found that the problems have been in communication between people and their mutual misunderstandings, the tools did not properly support activities required by the established processes, and the processes were not compatible. This claims that these three factors are important. This is described in paper A and [1].

In addition to the social culture described in [22], we have found a social culture difference in PDM and SCM, which e.g. reflects different terminology, same terminology with different meaning, and different approaches in development processes (usually with strict processes related to PDM, and iterative processes related to SCM), as well as their physical separation have increased the gap of their understanding. Furthermore, since hardware development processes are different from software development processes, integration points of the processes need to be established to achieve a successful integration and test of the final product. This has been described in paper A, B, C, and [1].

4.2 Summary of Included Papers

The papers included in this thesis cover three different areas: similarities and differences between PDM and SCM, integration initiatives, and the three main factors to be considered when building integration.

Paper A: Managing Complex Systems – Challenges for PDM and SCM,
In proceedings Software Configuration Management, SCM10, 23rd ICSE, Toronto, Canada, May 2001.

Authors: **Annita Persson Dahlqvist**, Ivica Crnkovic, Magnus Larsson
Annita Persson Dahlqvist is the main author of this paper, and has written part of section 3 Integration Possibilities, all in section 4 Integration Experiences, and all in section 5 Investigation Initiative PDM/SCM.

ABSTRACT:

In this paper we discuss the industry's need of controlling the whole product development process including both hardware and software components. The integration of development processes meets many problems partially because of the different nature of the processes and partially because of the different approaches. A typical example of overlapping processes is Software Configuration Management (SCM) and Product Data Management (PDM). Both SCM and PDM try to solve similar problems but in different ways. To get a more efficient development process, the companies try to integrate PDM and SCM systems, which has not yet been very successful. This paper gives a brief overview of common characteristics of SCM and PDM and gives an analysis of a possible integration. An example of an early attempt of integration is depicted. Finally, the paper presents an initiative by the Swedish industry to provide better understanding of SCM and PDM integration problems and to give directions for the possible integrations.

Paper B: Quality Improvements by Integrating Development Processes,
In Proceedings *11th Asia-Pacific Software Engineering Conference*, Busan, Korea, November 2004, IEEE.

Authors: **Annita Persson Dahlqvist**, Ivica Crnkovic, Ulf Asklund

Annita Persson Dahlqvist is the main author of this paper, and has written all sections (1, 2, 3, and part of 6) except section 5.

ABSTRACT:

Software is an increasing and important part of many products and systems. Software, hardware, and system level components have been developed and produced following separate processes. However, in order to improve the quality of the final complex product, requirements and prospects for an automatic integrated process support are called for. Product Data Management (PDM) has focused on hardware products, while Software Configuration Management (SCM) has aimed to

support software development. Several attempts to integrate tools from these domains exist, but they all show small visible success. The reason for this is that integration goes far beyond tool issues only. According to our experiences, three main factors play a crucial role for a successful integration: tools and technologies, processes, and people. This paper analyses the main characteristics of PDM and SCM, describes the three integration factors, identifies a model for the integration process, and pin-points the main challenges to achieve a successful integration of hardware and software development. The complexity of the problems is shown through several case studies.

Paper C: Important Factors for a Successful Integration of Product Data Management and Software Configuration Management Systems,

Technical report.
Authors: **Annita Persson Dahlqvist**
Annita Persson Dahlqvist is the author of the paper. The report summarizes certain parts directly related to the thesis from the book [1], in which Annita Persson Dahlqvist actively contributed as a co-author.

ABSTRACT:

Since PDM and SCM have been developed in their respective domain solving the domain specific requirements using different technology; on a higher level they seam to be similar in functionality, support and infrastructure. The similarities and differences, however, are found on practical lower levels such as in the product, evolution, and process model. The main characteristics of PDM and SCM are described more in detail. We have found in our investigations, that three factors are important to achieve a successful integration: processes, tools and technology, and people and culture. These three factors are discussed more in detail.

In addition, the report presents two case studies done at Ericsson Radio Systems AB and Industrial and Financial systems. The case studies are focusing on how the

companies are using PDM and SCM, their processes, any need for integration between PDM and SCM, and conclusions. The second case study was preformed later and it is not included in the book [1]. This case is used for validation of the hypothesis: the new elements in this case study are the starting assumptions that are based on the experiences and findings from the previous case studies.

4.3 Validity of the Research

To establish the quality of an empirical social research, where a case study is a form of such research, commonly four different tests are being used: (i) construct validity, (ii) internal validity, (iii) external validity, and (iv) reliability [18]. As a part of the validity we first identify and limit the scope of the research objectives and the research objects. The scope of our research is usage of PDM and SCM tools that includes their similarities and differences, and integration possibilities, including supporting processes and culture differences.

Construct validity relates to the data collection phase where correct operational measures for the study are established. The construct validity is dealt with performing multiple case studies at different organizations. Multiple-case design has been performed based on interviews, mail exchanges, and workshops in various companies both internationally and within Sweden. The collection of data were performed by interviews of people having different roles, such as configuration manager, process manager, development manager, people knowledgeable in PDM and/or SCM. All interviews were reported in a draft case study report and reviewed by the interviewed participants and informants. The case study reports were updated in accordance with the comments from the reviewers. In addition, the researchers experience in PDM and SCM provided a base for relevant focus on the investigations and the interviews.

Internal validity concerns in establishing a casual relationship between the observed behavior and the proposed explanation for this behavior. This test makes only sense for explanatory case studies. Our case studies are not explanatory, so we do not consider this type of validity.

External validity deals with the problem of knowing whether a study's findings are generalizable beyond the immediate case study. External validity has been dealt with by replicating the findings at several organizations performing different practices. The organizations represent both large companies with design teams consisting of thousands of developers, and medium-sized organizations with design teams approximately not more than 200 developers. In addition, the investigations cover a larger geographical area, such as Sweden, United Kingdom, and USA. All organizations are parts of large worldwide companies developing multi-technology and complex products. The companies and organizations have represented different business segments such as telecom, IT and services provider, hardware development tools provider, power and automation, enterprise systems, defense, and mobile phones. They are all representative for their business segments. Either the organizations deliver complex products consisting of hardware and software components, or deliver software products consisting of software components only. Even though the companies have different business goals, and serving different domains, similar methods are used in managing complex products. All studied organizations are struggling with manual management of product data related to software components. The product data is separately managed in different product data management systems with humans as integrators. Despite the fact that seven case studies have been performed, the result would be more reliable if more case studies have been performed, especially at companies in USA and Asia. Since no small company or organization has been included in the case study, we cannot generalize the findings to be valid for all sizes of companies. Furthermore, we do not know if the interviewees did understand what was asked of them. Since all interviewees knew the case study result should be published, we do not know if they answered truthfully or used an answer more positive for the company. We have however analyzed their statements with the observed practice, and we have compared the answers of different stakeholders.

The goal for **reliability** is to minimize the errors and biases in a study. This is to insure that later investigators, following the same procedures as described by an earlier investigator and conducting the same case study strategy all over again, should arrive at the same findings and conclusions. The description of the data collection methods and case study preparation materials will enable similar investigations to be performed by other investigators and other researchers to review the analysis and make valid conclusions. We have performed the investigations during the case studies following the same procedures and rules in all seven case studies. In particular the last case study has been performed after the hypotheses (described in section 3.2) were refined and results achieved (described in section 4.2), so it can be utilized as validation of the findings. The studies have been managed by description of data collection methods and the creation of a research database including background material, case study preparation material, and data collected in the different case studies. The data collection methods and case study preparation material will facilitate similar investigations to be made, e.g. in small and midsized companies or organizations. The collected data includes notes from interviews, presentation materials provided by the interviewees, written answers from the interviewees on the case study questions, and several documents provided by the companies or organizations. This will enable other investigators to conduct the same case studies using the described methods. The collected data will enable other researchers to investigate the material to ensure proper analysis has been made and valid conclusions have been drawn. Our hypotheses have been tested in case studies. After analyses have been performed, the hypotheses have been refined. The latest case has served as a validation case using the latest refined hypothesis.

5 RELATED WORK

"Imagination is more important than knowledge."

Einstein

In [7, 8] J. Estublier et al. discuss next steps for research such as unifying SCM and PDM, managing component-based software development, and understanding the relationship between SCM system models and software architecture [23, 24]. They conclude that it is clear why these issues are currently being addressed: SCM is no longer a stand-alone discipline.

The seminal papers by Jacky Estublier et al. addressing PDM/SCM integration [10, 11] have inspired current research. Estublier et al. conclude in the papers:

(1) In product engineering, there is a clear distinction between the design, called product model or product data, and the corresponding real artifact. In software engineering, the source code is the model but a compiler transforms at almost no cost the design into the product, which is also (a set of) files. The software is both the model and the product.

(2) The structure of the product, the nature of each component, the way two components fit together is highly constrained by the reality in PDM. In Software Engineering (SE) software is an intellectual construction. Worse, in SE, the technologies and methods are evolving very fast; no one of today standards will survive more than a few years.

It will require fundamental research and major experimentation to acquire understanding and investigate the integration possibilities. Integration of today's tools is not feasible due to too many differences in the concepts. It will require fundamental research and major experimentation.

Conradi and Westfechtel [12] conclude that many concepts of PDM and SCM are similar, but there are some differences concerning the objects to be managed. As a consequence, some sophisticated features for modern SCM systems are not applicable in Engineering Data Management (EDM) systems, another name for PDM systems. They continue their conclusion with the necessity of cooperation between the domains to support the development of hybrid products consisting of both hardware and software components in a uniform way.

In [25] J. Estublier et al. discusses how to provide a high level view where the application can be described, independently from the real tools specificities, and where the application behavior, services and properties can be described at that level of abstraction (process control, paradigm control, security etc.). Further, they provide a meta-model for interoperability between systems [26], where they introduce a new approach to SCM in which the system is built from potentially heterogeneous, existing pieces, with assembly mechanisms that enforce high-level properties. This approach does not provide a simple SCM tool, but a family of tools. Their experiment system shows that very advanced and state of the art features easily could be included into a federated system.

In [27] Gomes et al. present an approach to integrate system engineering artifacts and methods with discipline-specific detailed design artifacts and processes. They address the life cycle management of complex products involving mechanical, electrical, electronics and software aspects and being designed following a formal product development methodology. System data management (SDM) aims to support interdisciplinary collaboration and is a collaboration and infrastructure pattern. Under the SDM data model, discipline-specific artifacts remain under the management of their respective teams and repositories. The interdisciplinary traceability information is explicitly captured and managed in an additional repository called the SDM repository. They have developed a prototype SDM infrastructure based on a service-oriented architecture and existing PDM and SCM technologies to validate the concept. Each development organization has its own prescribed product development methodology and environments; therefore their SDM must be customized accordingly. The assumed prescribed methodology is rational unified process [28]. They have set up discipline-specific environments, where each discipline has its own environments, based on Clearcase®, Requisite Pro®, and the PDM tool SmarTeam. The SDM data model is implemented and processes, sub processes, artifacts, versions, branches, merge, and repositories are described.

Work at Chalmers University of Technology [29, 30] has aimed at developing an integrated product life cycle model framework, connecting information models representing a product through its life cycle ranging from customer needs to product retirement. In order to achieve this, product model theories from different domains such as mechanical, electronic, and software engineering were compared. Similarities and differences were found between these models. In the Chromosome Model theory [31] was shown how to implement the information requirements posed by mechatronic products. To some extent, the results have been validated in proof of concept prototypes, but there is a need for more comprehensive implementations in order to verify the ideas.

Crnkovic and Svensson [32] analyze the similarities and differences in processes of software and hardware development and used processes as an integration framework to achieve efficient interoperability between PDM and SCM.

In [33] Chalmers University of Technology and Industrial Research and Development Corporation have studied how to manage and distribute product data for embedded systems. Their purpose was to implement an information model to manage mechanical, electronic, and software components included in complex products. The project goals were to build an information model for a conformed version management, support for communication of product data between product development and logistic development, and implementation of the information model into one PDM system. The project has built an interface between SmarTeam (the PDM tool) and Visual SourceSafe (the SCM tool). The interface was PDM centric due to the project had no interest in having a bi-directional communication. In the PDM system a specific object for software was instantiated and manually fed with the path to the document or software stored in the SCM tool. Only the versions connected to a specific product revision were managed through the interface. The conclusion from the project is that this information model could be possible to use for integrations between other PDM and SCM tools.

In [34] Zimmerman discusses the information management for mechatronic products with focus on information modeling aspects. He concludes that there is no fundamental design theory for mechatronic products. No one seam to be fully incorporated in all levels, and detailed knowledge is required. There are many conceptual similarities between the different engineering domains. A more effective integration in the future does not seem impossible. Considering the total information scope of mechatronic product development, there is a risk of overflow. Instead Zimmerman proposes to partition information and system functions into a network of information systems with the PDM system as the backbone in such information system architecture. However, PDM needs to be complemented by other systems such as SCM systems.

In [14] Carlsson et al. discuss the importance for the industry to start planning for coordination of hardware and software development instead of managing the processes in parallel. The authors claim that there is low knowledge about processes for software development compared to knowledge about hardware product development. Carlsson et al. conclude that research in software engineering has focused on methods, techniques, such as object oriented programming, and usage of technologies supporting the developing process, e.g. CASE tools, but there is lack of knowledge regarding software development. The software development and its technology environment evolve fast, which leads to changes in the hardware development late in the development process. Research on hardware development has been in areas such as interaction between the humans and the processes involved in different phases. The authors continue that software and hardware domains with similar problems need to improve knowledge about the other domain. The important part is the interface between hardware and software development with little research performed. Carlsson et al. conclude that the two domains use different terminology, organized in different organizations, and different values are used in respective domain.

5.1 Conclusions

There are still major challenges with respect to theory and industrial practice, including the achievement of

- A deep understanding of the industrial requirements for collaborative development in the area, and also of the shortcomings of current commercial solutions vs. these requirements;
- An established shared terminology for interdisciplinary product development enabling engineers from different domains to communicate and collaborate effectively;
- A coherent theoretical basis and concept, that can guide the development of generic product information models, and consequently of generic development process models, including how to maximize the generic part and how to minimize the business-specific parts of an integrated PDM and SCM solution. These will need to have a holistic view, considering aspects of tools and technologies, processes, and organizations including culture differences.

New technologies and trends in modeling (such as component-based development and model-based development) and standards such as Extensible markup language (XML), XML metadata interchange (XMI), Standard for the exchange of product model data (STEP), Resource Description Framework

(RDF, RDFS) and Web Ontology Language (OWL) provide great possibilities to obtain integrations in new innovative ways and significantly improve intelligent support for collaborative PDM and SCM. Many of the PDM and SCM tools [35-39] have started to use these technologies and by that significantly increased integration possibility. The integrations themselves still have to be done.

6 CONCLUSIONS AND FUTURE WORKS

"The whole of science is nothing more than a refinement of everyday thinking."

Einstein

6.1 Conclusions

The primary objects of the study in this thesis are to describe possibilities to achieve an integrated support for the life cycle of products consisting of both software and hardware; concretely describe (i) the basic functions and underlying principles in PDM and SCM, their similarities and differences, (ii) integration possibilities, (iii) development processes and integration points in the processes, (iv) and culture differences between stakeholders from hardware and software domains.

Industries experience frequent problems with increasing costs and lower quality as there is a lack of an efficient integrated support for product development and maintenance. Both PDM and SCM provide similar support and have similar purposes, yet their integrations prove to be very difficult processes with so far no dramatic improvements. What causes problems in the integration attempts is that PDM and SCM have fundamental differences on their visions, assumptions, and underlying technologies. These differences can be traced back to crucial differences in the way hardware and software products are developed; e.g. concerning version evolutions, system architectures, development processes, and information system modeling. The PDM and SCM domains are different, but have similar patterns such as being huge domains, using completely different tools which both think can manage all situations in the other domain, and lack of knowledge in the adjacent domain.

Integration between PDM and SCM will undoubtedly lead to a more efficient development and maintenance support of today's complex products.

In our study we have found that three factors play a crucial role for a successful integration: tools and technologies, processes, and culture and people's behavior. For integration purposes, terminology and cultural differences are one of the factors to highlight.

6.2 Future Work

The work presented in the thesis is much related to state of the art and state of the practice, with inclusion of some general findings useful for further work on integration of PDM and SCM. For the future work related on the PDM-SCM integration we identify that research directions should focus on:

- *Information models*; the ultimate prerequisite for an ideal integration is a common information model (this includes structural static information and principle, and dynamic information (exchange) model). Since PDM has a more advanced and more formalized information model than SCM, the first step in the further work can be definition of an SCM model, and the second step its integration with a PDM information model. A starting work can be providing an information model for PDM and SCM at particular development organizations;
- *Architecture integration*; a technological base for integration of PDM and SCM can be achieved by identification and development of integration on the architectural level. Which are the points and what is the nature of integration in PDM and SCM architectures, which are the interfaces for the information exchange, how to integrate distributed systems, etc. – these are the question of interest for an architectural analysis;
- *Process integration*; study in detail common processes, define information flow between PDM and SCM, combine the processes and information flow to find integration points, e.g. analysis of version and configuration management in PDM and SCM and define a synchronizing model;

- *Culture differences*; The main challenge is to achieve common understanding of the processes, development approaches, and overcome problems closely related to social anthropology. The solution space lies in organizational, cultural and technical aspects. Finding and analyzing the possible solutions is a challenge for further research.

Our concrete plans for a future work include a combination of these challenges, with emphasizes on information modeling and processes, using particular industrial settings.

7 REFERENCES

[1] Crnkovic I., Asklund U., and Persson Dahlqvist A., *"Implementing and Integrating Product Data Management and Software Configuration Management"*, ISBN 1-58053-498-8, Artech House, 2003.

[2] Buur, J. *"A Theoretical Approach to Mechatronics Designs"*. Doctorial Dissertation, IK publication 90.74 A, Technical University of Denmark, Institute for Engineering Design, Lyngby, Denmark 1990.

[3] CimData, *Product data Management and Computer-Aided Software Engineering*, 2000.

[4] U. Asklund, I. Crnkovic, A. Hedin, M. Larsson, A. Persson Dahlqvist, J. Ranby, and D. Svensson. *"Product Data Management and Software Configuration Management - Similarities and Differences"*, The Association of Swedish Engineering Industries, V040096, 2001.

[5] Svensson D. and Crnkovic I., "*Information Management for Multi-Technology Products*", International Design Conference - Design 2002, IEEE, 2002.

[6] D. Svensson, *"On Product Structure Management throughout the Product Life Cycle"*, Thesis for the Degree of Licentiate of Engineering, Machine and Vehicle Design, Chalmers University of Technology, 2000.

[7] J. Estublier, D. Leblang, G. Clemm, R. Conradi, A. VanDerHoek, and W. Tichy, *"The Impact of the research community in the field of Software Configuration Management"*, SEN, September 2002.

[8] J. Estublier, D. Leblang, G. Clemm, R. Conradi, A. VanDerHoek, and W. Tichy, *"The Impact of the research community in the field of Software Configuration Management"*, International Computer Software Engineering, May 2002, Orlando Florida.

[9] W. A. Babich, *"Software Configuration Management: Coordination for the Productivity"*, Reading, MA: Addison Wesley, 1995.

[10] Estublier J., "*Software Configuration Management: A Roadmap*", In Proceedings of 22nd International Conference on Software Engineering, The Future of Software Engineering, pp. 279-289, ACM Press, 2000.

[11] Estublier J., Favre J-M., and Morat P., "*Toward SCM/PDM Integration?*", In Proceedings of Software Configuration Management SCM-8, Lecture Notes in Computer Science, nr 1439, pp. 75-94, Springer, 1998.

[12] B. Westfechtel, and R. Conradi, *"Software Configuration Management and Engineering Data Management: Differences and Similarities"*, Proceedings of 8th International Symposium on System Configuration Management (SCM-8), Lecture Notes in Computer Science, No. 1439, Berlin Heidelberg, Germany: Springer-Verlag, 1998, pp. 96-106.

[13] J. Estublier, D. Leblang, G. Clemm, R. Conradi, A. VanDerHoek, W. Tichy, D. Wiborg-Weber, *"Impact of the Research Community for the field of Software Configuration Management"*, Proceeding International Computer Software Engineering, ICSE2002, pp. 643-644.

[14] C. Karlsson, E. Lovén, *"Utveckling av komplexa produkter – integrerad mjukvara i traditionellt mekaniska produkter"* Institute for Management of Innovation and Technology, IMIT report IMIT 2003:127.

[15] Persson Dahlqvist A., Crnkovic I., and Larsson M., "*Managing Complex Systems - Challenges for PDM and SCM*", In Proceedings of International Symposium on Software Configuration Management, SCM 10, 2001.

[16] Persson Dahlqvist A. Crnkovic I., and Asklund U. *"Quality Improvements by Integrating Development Processes"*, In Proceedings of Asia-Pacific Software Engineering Conference, APSEC2004, 2004.

[17] A. Persson Dahlqvist, I. Crnkovic, M. Larsson, *"Managing Complex Systems – Challenges for PDM and SCM"* has been published in the Journal Projects & Profits, IV-7 (Rs 60): 36-42, July 2004, ICFAI Press, Panjaquatta, India.

[18] Colin Robson, *"Real world research"*, ISBN 0-631-21305-8. Blackwell Publishing, Malden USA, Oxford UK, Victoria Australia.

[19] Brooks, Jr., F.P. 1988: "*Grasping Reality through Illusion—Interactive Graphics Serving Science*" Invited keynote address at Conference on Human Factors in Computing Systems, Washington, D.C., May 17. Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, May 1-11. E. Solloway, D. Frye, and S. Sheppard, Eds. Reading, MA: Addison Wesley, 1988.

[20] Judith Bell, *"Doing Your Research Project: A guide for first-time researchers"* in education and social science (third edition), ISBN 0-335-20388-4, Open University Press, Maidenhead, Philadelphia. England, USA, 1999.

[21] Robert K. Yin, *"Case Study Research: Design and Methods"* (third edition), ISBN 0-7619-2553-8, Sage Publications, 2003.

[22] G. Hofstede, *"Cultures and Organizations Software of the Mind Intercultural Cooperation and its Importance for Survival"*, ISBN 0-07-029307-4, McGraw-Hill, 1997.

[23] André Van der Hoek, Dennis Heimbigner, Alexander L. Wolf, *"Software Architecture, Configuration Management, and Configurable Distributed Systems: A Ménage a Trois"*. Tech Report CU-CS-849-98. U. Colorado.

[24] André van der Hoek, Dennis Heimbigner, Alexander L. Wolf. *"System Modeling Resurrected"*. System Configuration Management (SCM-8), Brussels, Belgium 1998, Springer-Verlag LNCS 1439.

[25] J. Estublier, H. Verjus and P.Y. Cunin, *"Building Software Federations"*, The 2001 International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA) Las Vegas, Nevada (USA). June 25-28, 2001.

[26] J. Estublier, S. Garcia and German Vega, *"Using Federations for Flexible SCM Systems"*, SCM-11 May 2003, Portland, Oregon, USA.

[27] Gomes, M-M. Singh, M. Keren, S. Zeng, J. Rubin, L. Balmelli, I. Boier-Martin, *"System data management: An Inter-Disciplinary collaboration Architecture for Systems Engineering"*, ASME IDECT/CIE 2005.

[28] Kroll P. and Kruchten P., *"The Rational Unified Process Made Easy"*, ISBN 0-321-166009-4, 2004.

[29] Hallin, K., Zimmerman, T., Svensson, D., Malmqvist, J., (2003) *"Modeling Information for Mechatronic Products"*, Proceedings of ICED 03, Stockholm, Sweden, 2003.

[30] Zimmerman T., Hallin K., Malmqvist J. (2004) *"Information Model for the Mechatronic Product Focusing the Functional Abstraction"*, Proceedings of DESIGN 2004, Vol. 1, Dubrovnik, Croatia, 2004, pp. 571-581.

[31] Andreasen, M.M. (1992) *"Designing on a "Designers Workbench" (DWB)"*, Proceedings 9th WDK Workshop, Rigi, Switzerland.

[32] Svensson, D., Crnkovic, I. (2002) *"Information Management for Multi-Technology Products"*, Proceedings of DESIGN 2002 – Vol.1, Dubrovnik, Croatia, pp 551-559.

[33] Vinnova dnr 2002-01491, *"Slutrapport av projektet Hantering och spridning av produktdata för inbyggda system"* (Translated by author;*"How to Manage and Distribute Product Data for Embedded Systems"*), IVF-rapport 040003.

[34] T. Zimmerman, *"Information Management for Mechatronic Products Focusing on Information Modeling Aspects"*, Thesis of he degree of Licentiate of Engineering, Chalmers University of Technology, Gothenburg, Sweden 2005.

[35] ClearCase, www.ibm.com, October 2005.

[36] Telelogic CM Synergy, www.telelogic.com, October 2005.

[37] PVCS, www.serena.com, October 2005.

[38] Teamcenter, www.ugs.com, October 2005.

[39] E-Matrix, www.matrixone.com, October 2005.

APPENDIX 1
ACRONYMS

BOM	Bill of materials
CAD	Computer-Aided design
CAE	Computer-Aided-Engineering
CAM	Computer-Aided-Manufacturing
CASE	Computer-Aided software engineering
CM	Configuration Management
DM	Document Management
EDM	Engineering document management
OWL	Web Ontology Language, based on RDFS (see below); formerly DAML+OIL. OWL and RDFS enable richer semantics to be attached to models. See: http://www.w3.org
PDM	Product data management
PLC	Product Life Cycle
RDF, RDFS	Resource Description Framework, an XML-based markup language for describing web resources. RDFS is RDF Schema, the RDF Vocabulary Description Language
SCM	Software configuration management
SDM	System data management
STEP	Standard for the exchange of product model data
UML	Unified modeling language
XMI	XML metadata interchange
XML	Extensible markup language

PAPER A
MANAGING COMPLEX SYSTEMS
CHALLENGES FOR PDM AND SCM

Annita Persson Dahlqvist
Ericsson Microwave Systems AB
Transmission Mobile Systems
431 84 Mölndal, Sweden
Annita.Persson@emw.ericsson.se

Ivica Crnkovic
Department of Computer Science
Mälardalen University
721 23 Västerås, Sweden
Ivica.Crnkovic@mdh.se

Magnus Larsson
Development and Research
ABB Automation Products AB
721 59 Västerås, Sweden
Magnus.Larsson@mdh.se

In Proceedings of the SCM 10, workshop on Software Configuration Management, 23rd ICSE, Toronto, Canada, May 2001

Abstract

Within the industry there is a need of controlling the whole product development process including both hardware and software components. The integration of development processes meets many problems partially because of the different nature of the processes and partially because of the different approaches. A typical example of overlapping processes is Software Configuration Management (SCM) and Product Data Management (PDM). Both SCM and PDM try to solve similar problems but in different ways. To get a more efficient development process, the companies try to integrate PDM and SCM systems, which has not yet been very successful.

This paper gives a brief overview of common characteristics of SCM and PDM and gives an analysis of a possible integration. An example of an early attempt of integration is depicted. Finally the paper presents an initiative by the Swedish industry to provide better understanding of SCM and PDM integration problems and to give directions for the possible integrations.

Keywords: Software Configuration Management, Product Data Management, Development process.

1 Introduction

Product Data Management (PDM) is the discipline of designing and controlling the evolution of a product [1, 3, 5]. Software Configuration Management (SCM) is the discipline of controlling the evolution of a software product. Historically PDM has been focused on hardware development and SCM has been focused on software development. A trend in both domains is the understanding of the needs for co-operation, especially on the tool side by natural causes. In industry trend today is to manage the entire product and not the hardware and the software part separately. To get a user-friendly and efficient development environment, the companies try to integrate different systems. PDM and SCM systems are part of this integration.

PDM vendors have ignored software management in their development activities. Similarly, SCM vendors were, up to very recently, concentrated on supporting pure software development. In general there is a lack of knowledge in both disciplines, and exhaustive research is needed to find out which way of integration and interaction is the most suitable. For vendors and users, the payoffs are likely to be tremendous for a relatively low-cost and minimal investment of resources in software management. However, the vendors have been too occupied with increasing challenges within their domains and did not have considered possibilities of unifying processes. It is likely that PDM and SCM users must determine what they expect to accomplish from such an integrated system and then put pressure on vendors to deliver those capabilities. This was one reason for the Swedish industry and academia to start an initiative to analyze the similarities and differences between SCM and PDM. The initiative will indicate the need for using a common development support, with integrated functions from these two domains. The paper gives an overview of the overlapping disciplines and shortly describes the aim and the goals of this initiative.

2 SCM and PDM Domains

The characteristic of SCM and PDM originates from the nature of the artifacts developed. In the life-cycle models, PDM is focused the hardware design phase and later at the production and maintenance/support phase. The software development phase support is significantly smaller. On the opposite, in the software product life cycle, the development phase is usually marked as the most intensive part. According to these efforts, the tools bring into focus the support for the corresponding processes.

Figure 1 schematically shows support provided by these tools during the product life cycle. SCM and PDM together support the entire product life cycle. A possibility of integration is even more attractive as the trends in both systems are enlarging the area of control that is already covered by the other system. For example, the configuration management of imported components is getting more important than pure version management of source code [4]. SCM becomes more similar to PDM due to structuring and configuration of complex products. On the other hand the development phase, due to extensive use of CAD and simulation tools, becomes more important for PDM users.

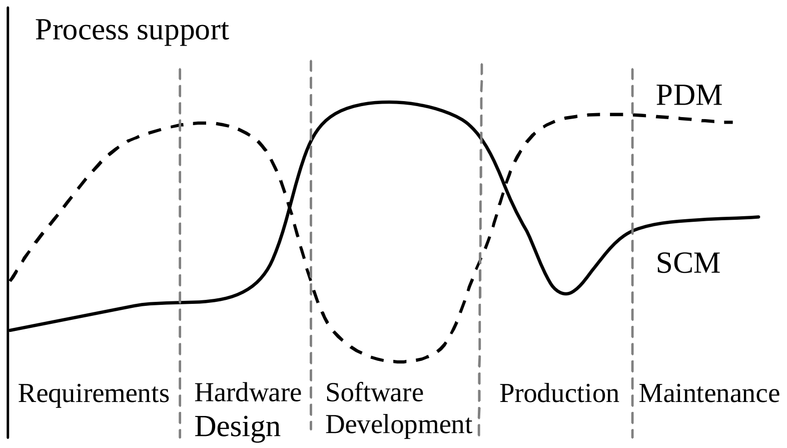

Figure 1. PDM and SCM Process support

In practice, there exist many problems. First, neither SCM nor PDM systems have yet completely solved the problem of sharing or exchanging data between different tools from the same domain but from different phases. A more serious problem occurs when data must be shared or exchanged between the tools from these two domains. The second problem is to choose the tools and methods to cover the overlapping areas. Even if a particular tool gives an excellent support within one domain, it does not mean that it is suitable or well integrated within the second domain.

The overlapping functions are numerous. Figure 2 depicts the most important functions from both domains. The figure shows that there are numerous functions supporting the same or similar process.

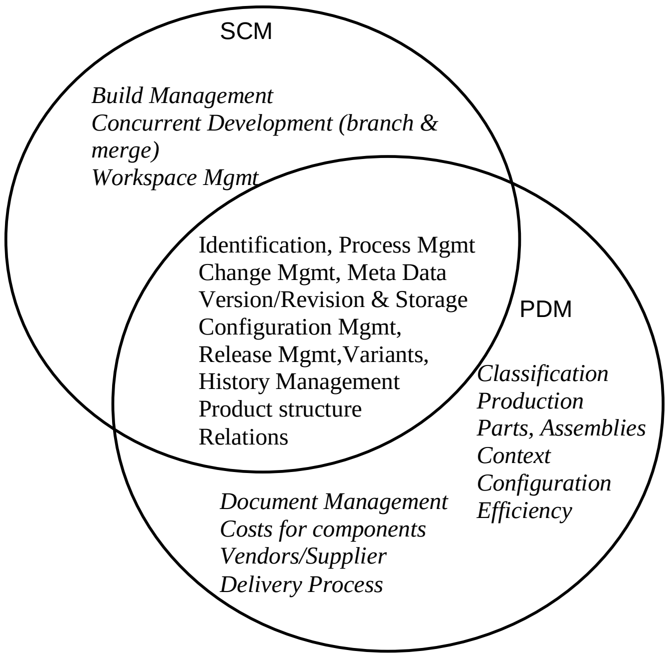

Figure 2. The main functionality of SCM and PDM

There is an urgent need of a common terminology and semantics to understand the two disciplines.

We can conclude that there are many similarities on the conceptual level between PDM and SCM, but the emphasis of different moments is quite different. The implementations are also different. It is not possible just to take the systems, package them together and use them as a single product. What is needed is a careful integration of specific parts of these systems, which can even require redesign of these parts, or a complete new approach must be taken.

3 Integration Possibilities

The question is which kind of integration or cooperation can be achieved with these two systems? To find out the real possibility for integrating the tools, analyses beyond the functional level must be done. Estublier [2] analyses similarities and differences looking at the following categories:

- The product model (data model, configuration);
- The evolution model (versioning);
- The process model.

A full integration can be achieved by using common infrastructure, common interfaces and common data. This and other analysis shows that the support within these categories is very different, except for the process model where a general development process can be supported in both systems [2, 8].

Another possibility of integration is week integration with separated infrastructures and data, but well-defined and efficient interface between them.

The simplest way of integration is building a common application user interface that will manage both SCM and PDM functions and use them as a common interface to the users.

This model unfortunately cannot work well for tools available on the market today. Most of them have a poor API, which provide a partial (if any) functionality. However, one main challenge for both tools, independently of each other, is interoperability with other engineering tools. The interoperability requirements will require of better and clearer APIs. The new, component-based technology also encourages use of APIs, so we can expect better possibility of integration in the future.

As APIs of SCM and PDM tools do not provide full functionality, a solution in practice can be as shown in Figure 3.

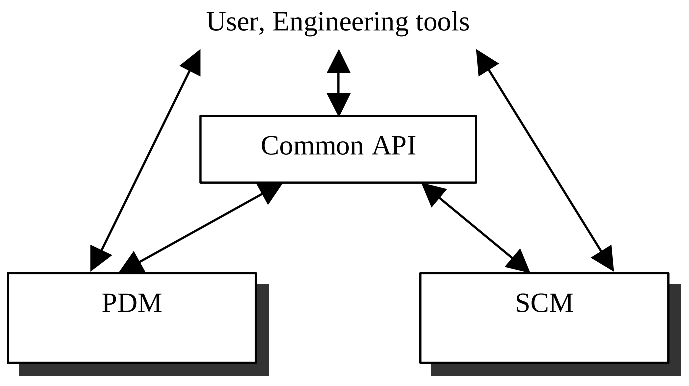

Figure 3. Direct and indirect use of tools

Such solution may generate problems as some manual actions may introduce inconsistent states for a SCM/PDM combination.

To make integration more robust and efficient, the SCM and PDM vendors should provide the integration. As PDM covers larger part of the total product life cycle and as PDM deals with meta-data (i.e. description and structuring of data), it is natural that the communication to the user goes through PDM, as shown in Figure 4. PDM tools use the API from SCM. The users communicate only via PDM, which tool is responsible for updating the information from both PDM and SCM data. This model provides better control for the consistency of duplicated data. However a similar problem remains as in the previous model as per figure 3 explained.

The integration between different development tools and SCM tools already exist and it is unrealistic that they will not be used independently of PDM integration. This means that there will always be a possibility to modify data in one database and introduce an inconsistent status. To avoid possible inconsistencies, a database synchronization process must be included between the databases on periodical or interrupt/trigger base.

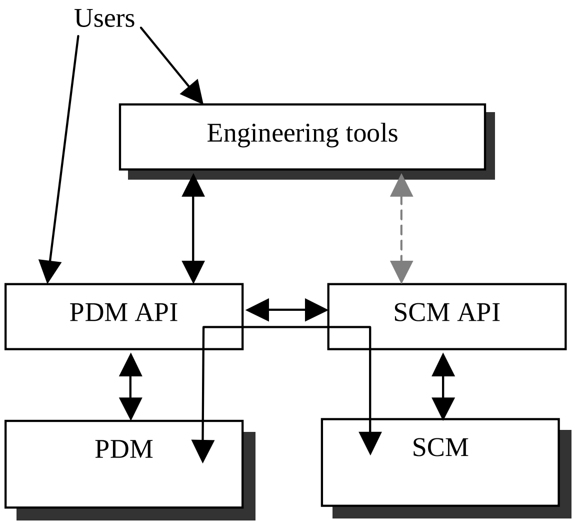

Figure 4. Partial direct SCM/PDM integration

Another problem, which already exists in both systems, becomes more acute in the integration process. PDM and SCM tools are complex and have complex and often unfriendly user interface. When integrated, the system will impose an even more complex user interface.

What parts of the tools can be integrated depends on the specific tools. The minimal integration required is the one on the version and configuration level. As PDM does not have flexible mechanisms for version management it is suitable to have file versioning under SCM control. From the PDM perspective, it is more interesting to keep information about specific versions of files collected in a configuration or in a baseline.

Workspace management is very important in SCM and in more advanced tools tightly integrated in the entire process. This part must remain under SCM control as well, which implies direct interaction between users and a SCM tool. This approach deviates from the general intention to have control of the product from one tool.

Change management and general process management can be kept under PDM control. This implies that change management parts in SCM tools should be hidden from users in form of process and action initiation, but kept as triggers to actions and information status inside SCM. The SCM change management mechanisms must be used if we want to have traceability of changes down to source code.

4 Integration Experience

Today there is one first known attempt for integration between an SCM system and a PDM system. The integration is between the SCM system ClearCase [6] and the PDM system Metaphase [7]. Technically ClearCase is more or less a file manager, and Metaphase is an object-oriented tool. The integration has to deal with a mapping between an object and a file.

The first releases of this integration attempted to get hold of data in ClearCase from the Metaphase environment. The interface to **this** integration is designed to manage software files from ClearCase into Metaphase. The main functionality is to find files within ClearCase, to register the file, and manage metadata about the file in Metaphase. When you have registered a file/object in Metaphase, you are allowed to build relationships in your product structure to the registered file. Metaphase is managing the product structure for the whole product. Later releases of the interface will cover the aspect of managing components from Metaphase into ClearCase. The interface is developed by SDRC, the vendor of Metaphase.

In Metaphase the software products will be managed together with all hardware products within the same product structure. In Figure 5 shows an example of a product structure including both hardware and software components, which is managed by Metaphase. The structure is a part of the product MINI-LINKTM, used in mobile networks developed and manufactured within Ericsson. The product contains several parts, e.g. a radio and a modem, which both contain of PCB's, Printed Circuit Boards, rack, and embedded software parts for control functions. The software part is the executable module and will be treated as embedded software that is represented as a box with relationship to other boxes.

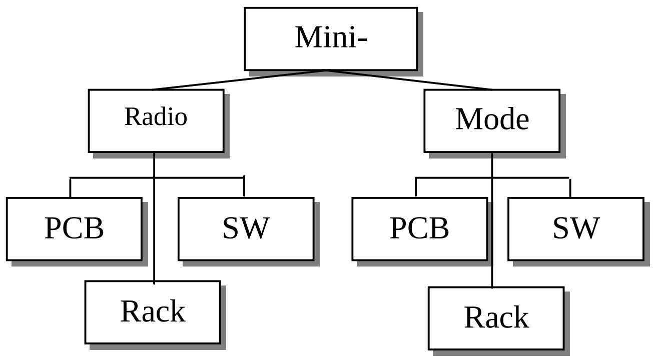

Figure 5. Example of a Product Structure

In the shown example the PDM system contains the result of the software development. The aim of the integration is to manage development cycles of both hardware and software parts. The integration is based on a common interface. The interface is built on a data exchange facility, where Metaphase is running ClearCase commands with arguments through perl scripts and the results from ClearCase will be stored within an XML-file. How the exchange is performed is shown in Figure 6. The design of the interface started with the ClearCase 3.2 and Metaphase 3.1, but had to be extended to later versions of Metaphase.

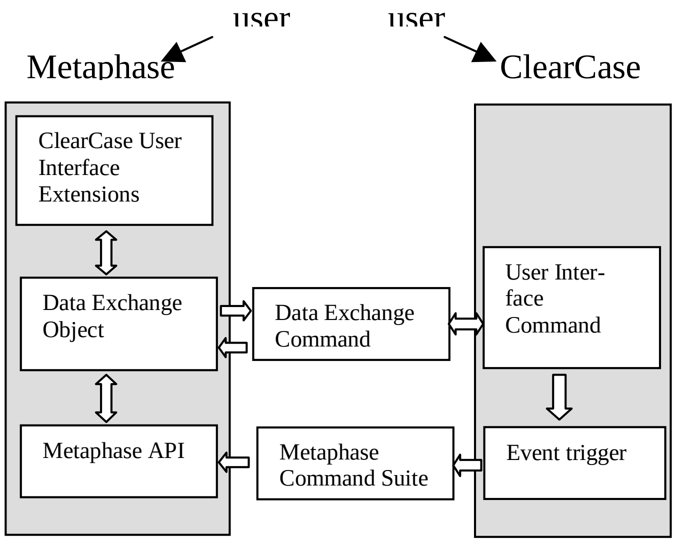

Figure 6. Data Exchange Architecture

So far there are no plans for using the ClearCase API instead of this data exchange facility.

ClearCase is still the SCM tool owning all files stored in the tool. In Metaphase you have to manually set up the path to a specific file in ClearCase to be able to see it. Metaphase is not able to create new versions of a file stored and owned in ClearCase. This has to be done within ClearCase.

We have tested the very first release of the interface and found it is not easy for the end-user to understand and use the interface. First of all the end-user has to have a full understanding of both systems on a technical and terminology level. This requires more training of the end-users. Secondly, the user has to determine if the actual data he/she wants to manage in Metaphase should have a static version in ClearCase or if it should always have the latest version in ClearCase. Today there are two different ways of getting the data from ClearCase; through the ClearCase way of describing the actual version, or through a static view defined in Metaphase. This view has nothing to do with a ClearCase view. A third way of finding data in

ClearCase by using the configuration specification rule files will not be available until later releases.

The following conditions are assumed to get the integration to work properly:

- The end-user has to have an in-depth technical knowledge of both systems to understand how to use the interface, and to understand the mixed terminology within the manuals and the interface;
- A ClearCase view must exist before registering in Metaphase;
- The view must be started in ClearCase before being used in Metaphase;
- The file system has to be defined in Metaphase first;
- The owner of the Metaphase installation software has to be the owner of the ClearCase vob mount point;
- A view in ClearCase cannot be updated from Metaphase;
- Only metadata of one file at a time is possible to get hold of in ClearCase, no transferring of meta data of a number of files concurrently;
- A software product, built in ClearCase, managed in Metaphase has to be registered in the PDM system. This means that the product will be put under version control in both systems;
- A software product managed in Metaphase is stored within ClearCase, but metadata are placed in both systems.

These required conditions show how complicated the integration is developed. There exists a high risk that data will not be synchronized. In addition to these implementation problems, there exist problems of a more general nature. SCM users do not understand how PDM systems work and vice versa. All PDM systems, including Metaphase, are designed to meet the needs of the hardware people including their terminology and not the SCM people.

5 Investigation Initiative for PDM/SCM

Since many companies struggle with problems using SCM/PDM systems, and more is to be expected, The Association of The Swedish Engineering Industries is sponsoring a project where similarities and differences between SCM and PDM are studied. The investigation team consists of a mixture of industry and academia people with in-depth knowledge of SCM and PDM. The team utilizes their own knowledge in the two areas, but is also interviewing different companies to get a deeper knowledge of the problems related to SCM and PDM integration. Literature, research results, vendor information and other related information is also used to get a better understanding.

The purposes for the project are to:

- Give large companies in-depth knowledge about PDM and SCM including the most common tools and a general theoretical description of SCM and PDM;
- Give smaller companies knowledge of how far they can use the tools they have today;
- Find out how SCM and PDM systems work together, what do they have in common;
- Describe the differences, similarities, and overlapping parts of SCM and PDM;
- Gather experiences – status within Swedish Industries through interviews;
- Investigate trends from other countries, companies and researchers;
- Set up a fictional scenario where one hardware company is merged together with a software company; how to treat the different ways of managing products and development data, misunderstanding of the two different groups of developers; suggest a process for this merging of companies or organizations.

The work is performed with the following activities:

- Meeting within the team;
- Study different literatures – research or industrial papers;
- Discussion with experienced industry people;
- Discussion with experts within the PDM and SCM area, researchers, industry people, and vendors.

This project will deliver a technical report covering what SCM and PDM have in common and how to get a better understanding of both areas. All interviews made during the research will be included in the report. A conference will also be set up to share the knowledge collected.

6 Conclusion

Although the trends in system development take an integrated approach, where products are built from both software and hardware, these processes are still separated. One of the reasons is inadequate integration between tools managing hardware and tools managing software. Current SCM and PDM systems differ too much to be easily integrated. The integration can be achieved by exchanging data using import/export functions triggered by change of state in databases or invoked through API from users of other engineering tools.

We expect the outcome of the work, performed by the Swedish Engineering Industries group, to give a deeper understanding, guidelines and more efficient usage of PDM and SCM.

7 References

[1] CIMdata: "*Product Data Management: The Definition*", CIMdata Inc., Ann Arbor, MI, USA, 1998.

[2] J. Estublier, J-M Favre and P. Morat: *"Toward SCM/PDM Integration?"*, System Configuration Management, SCM-8, Lecture Notes in Computer Science 1439, Springer, pp. 75-94.

[3] S. B. Harris, "*Business strategy and the role of engineering data management: a literature review and summary of the emerging research questions*", Proceedings of the Institution of Mechanical Engineers: Part B: Journal of Engineering Manufacturing, 210: pp. 207-220, 1996.

[4] M. Larsson, I. Crnkovic, *"New Challenges for Configuration Management"*, System Configuration Management, SCM-9, Lecture Notes in Computer Science 1675, Springer, 1999.

[5] P. Pikosz: "*Product Data Management in the Product Development Process*". Licentiate thesis, Machine and Vehicle Design, Chalmers University of Technology, Göteborg, Sweden, 1997.

[6] Rational ClearCase, http://www.rational.com/products/clearcase/index.jsp

[7] SDRC Metaphase, http://www.sdrc.com/metaphase

[8] B. Westfechtel, R. Conradi: *"Software Configuration Management and Engineering Data Management: Differences and Similarities"*, System Configuration Management, SCM-8, Lecture Notes in Computer Science 1439, Springer, pp. 96-106.

PAPER B
QUALITY IMPROVEMENTS BY INTEGRATING DEVELOPMENT PROCESSES

Annita Persson Dahlqvist[1], Ivica Crnkovic[2], Ulf Asklund[3]

[1]Ericsson AB, Mölndal, Sweden,
Annita.Persson.Dahlqvist@ericsson.com
[2]Mälardalen University, Department of Computer Science and Engineering, Västerås, Sweden, ivica.crnkovic@mdh.se
[3]Department of Computer Science, Lund University, Lund, Sweden,
ulf.asklund@cs.lth.se

In Proceedings for the 11th Asia-Pacific Software Engineering Conference, Busan, South Korea, November 2004.

Abstract

Software is an increasing and important part of many products and systems. Software, hardware, and system level components have been developed and produced following separate processes. However, in order to improve the quality of the final complex product, requirements and prospects for an automatic integrated process support are called for. Product Data Management (PDM) has focused on hardware products, while Software Configuration Management (SCM) has aimed to support software development. Several attempts to integrate tools from these domains exist, but they all show small visible success. The reason for this is that integration goes far beyond tool issues only. According to our experiences, three main factors play a crucial role for a successful integration: tools and technologies, processes, and people. This paper analyses the main characteristics of PDM and SCM, describes the three integration factors, identifies a model for the integration process, and pin-points the main challenges to achieve a successful integration of hardware and software development. The complexity of the problems is shown through several case studies.

1. Introduction

Traditionally, hardware development has been separated from software development. The development processes have been separated and different tools have been used to support these processes. In fact, software products have been clearly separated from hardware products during development, and they have not been integrated before the start of system verification. Today this border between hardware and software begins to vanish. The final product is a result of tight integration of hardware and software components and the decision whether a specific function should be implemented in hardware or software may come late in the project and may even change during the products life cycle. When the border become vague it is no longer possible to keep the development organizations separated and to use different life cycle processes, but they should be integrated. However, the requirements for such integration points out a number of problems: process adjustments, information exchange, access and flow, infrastructure support, tool integration, cultural differences, etc. To integrate the processes and the tools have been difficult problems and challenges for many companies [2].

Product Data Management, PDM, is an engineering discipline including different methods, standards, and tools. It (i) manages the data related to products, (ii) supports procedures during the product lifecycle, and (iii) deals with the development and production infrastructure [1],[2],[14]. Traditionally PDM deals with hardware components only.

The software development phase is characterized by collaboration and coordination of many developers. Software Configuration Management, SCM, manages this type of complexity. The scope of SCM is to (i) keep track of all the files and modules constituting the product, (ii) manage all the changes made to these items during their entire life, and (iii) manages all documentation related to the product [1],[2],[14].

On the system level, where hardware and software components are integrated, the goal is to control the product development process for the entire product [1],[2]. To effectively manage a complex system on the system level, adjustments of all included processes are needed [4],[14]. To bridge the gap between PDM and SCM, three main factors are crucial; (i) processes, (ii), tools and technology and (iii) people and cultural behaviors.

During 2001 the Association of Swedish Engineering Industries sponsored a project about PDM and SCM. As part of this project several case studies were performed, e.g. ABB and Ericsson AB, in order to analyze concrete current requirements and solutions. The project resulted in a report [1]. The work went further, more case studies were performed, Sun Microsystems, and Mentor Graphics, which resulted in a book published by Artech House [2].

In this paper we use our experiences from these earlier studies to analyze the gains of integrating PDM and SCM. We identify the main challenges to achieve a successful integration of hardware and software development processes, mainly on the development phase. We have focused on the two domains PDM and SCM, and our analysis is based on studies of different PDM and SCM tools and several case studies from large companies using PDM and SCM with different levels of integration. The case studies were made during several years and some of the findings have been published [2],[1],[3],[11]. This paper gives an overview of our conclusions, illustrated by some of the cases.

The remainder of this paper is organized as follows. The important integration issue, life-cycle processes, is discussed in section 2 in which we point out some major similarities and differences between hardware and software development processes. The second factor, tools and technology, is discussed in section 3. Major differences and similarities in a technology aspect are discussed. The third factor, people and cultural behaviors, together with terminology are discussed in section 4. In section 5 we discuss different integration aspects. Finally, section 6 concludes the paper.

2. Development Processes and Infra-structure Support

The development of hardware and software products seams on a high level to be very similar. Similar processes are used and the infrastructure and data flow used to manage all information are also similar. The question is if this similarity is deep enough to make it possible to either integrate them seamlessly or to let one of them acquire the other. Can software development acquire a hardware development process, and vice versa? To answer these questions we analyze both processes and the underlying support from PDM and SCM: data flow, information management, and standards used or supported within these domains.

2.1. Processes and Underlying Principles of PDM and SCM

Often the result of the development phase for a hardware product is the set of many different documents describing both the product itself and the included components, e.g. drawings, manufacturing specifications, bill of materials, etc. Everything included in a hardware product has to be described and documented, before the pre-production phase can start to produce a prototype, which often is done once or twice before ramping-up the production to full scale. In the pre-production phase the documents are used by the manufacturing people often located in another organization within or outside the company. The manufacturing phase is usually long and costly, e.g. a new production line has to be purchased and set up, new tools have to be designed and produced. Furthermore, changes to a hardware product have to be done first in the documents and then in the production phase.

The most commonly used process for hardware development is the waterfall model, as shown in Figure 1. The main characteristics are the sequential flow of information, and the presentation of data and structures following the physical structure of the product.

The most important PDM-related requirements for hardware development are document management, product structure management, and process support. The objects managed in PDM are not the products themselves but different data about the products designated as metadata. This data is usually collected from different tools and spread out through different organizations.

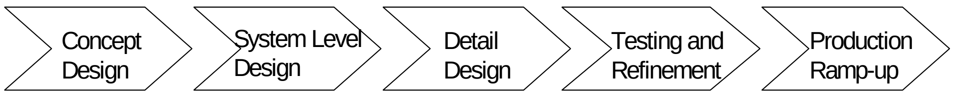

Figure 1. A generic Waterfall model commonly used in hardware development

During software product development the product is often designed incrementally, i.e. planned parts of the software are designed, integrated, and tested before next increment starts. Figure 2 shows an example of three increments and their activities [9]. The developers build the executables often, sometimes on a daily basis. All necessary documents are written in an incremental way too. When all increments are finalized, the software is built and released. The build, the production phase, is very short and cheap compared to hardware production.

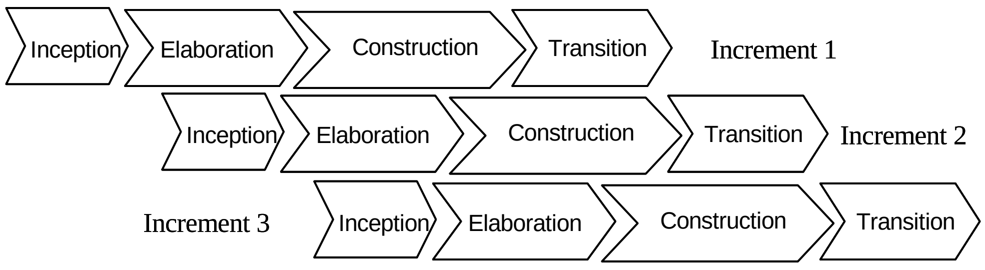

Figure 2. A generic incremental model commonly used for software development

The most fundamental differences in the development processes are the following: hardware development, supported by PDM, follows a sequential process with a clear separation between the phases. The software development process, supported by SCM, is flexible, with unclear borders between the phases. While outcomes from different phases of hardware development differ significantly in form and even physical shape (a technical drawing of a product is very different from the product itself), the outcomes from software development phases are very similar and often only transformations of each other (for example, from a UML design code can be partially generated, and the final production is a transformation of source code into binary code). Such facts make these processes incompatible.

2.2 Information Management and Data Flow – A Case Study

A discussion of the hardware and software design process and information usage in [14] concludes that every company has its own customized development process, usually a variant of the generic model. Therefore, in this section we discuss a case study from the telecommunication company Ericsson [15] where we look into (i) where in the processes the information is generated, (ii) in which tools the information is stored and when, (iii) how the information is interchanged between these tools, and (iv) how information is managed on a system level.

During the development of hardware products, information is created in all phases: during concept and system level design mostly requirements documents, in detail design phase the documents specifying the product, in the testing and refinement phase the changes of the product the change requests and documentation updates, and during production ramp-up phase a few changes in the documentation (see Figure 3). Drawing documents are created in CAD/CAM tools. They are stored in the PDM system for manufacturing accessibility. Some documents will remain in the development tools due to internal database structures not possible for extraction.

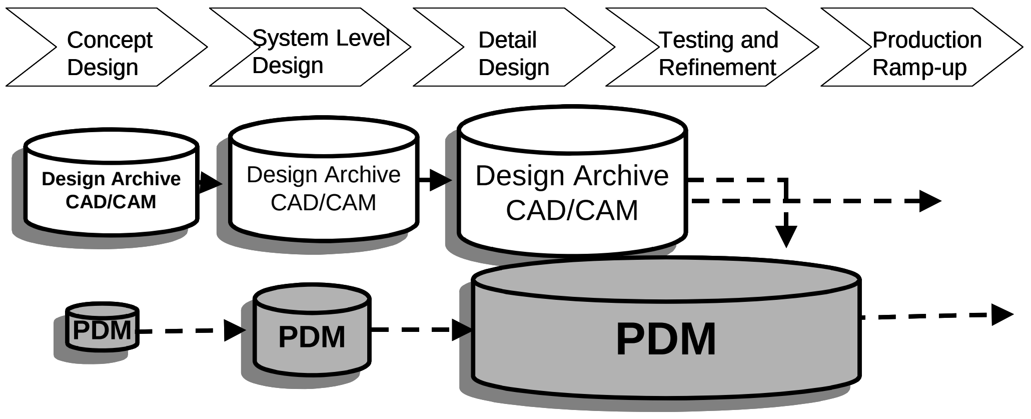

Figure 3. Processes and information storage for a hardware product

Similarly in software development, information is created in all phases. During the inspection phase, documents describing different requirements on the product are written. During the elaboration and construction phases use cases, source codes, detailed design descriptions, test cases, and user documentation are written, executable files generated, and test cases performed. In the phase transition, the final product is tested for deployment. The software product is ready and transferred to the PDM system for manufacturing accessibility (see Figure 4).

In the case of hardware development we see that the tendency is to save most of the information in a PDM system, while in the case of software development it is the SCM system that comprises most of the information, although the final product information might be stored in a PDM system. In both cases, PDM and SCM have a similar integration role. The question is if in an integrated environment, one of these systems can overtake the role from the other (can PDM or SCM be exclusive information integrator)? To answer this question we must look at the differences and similarities between the tools and underlying technologies described in section 3.

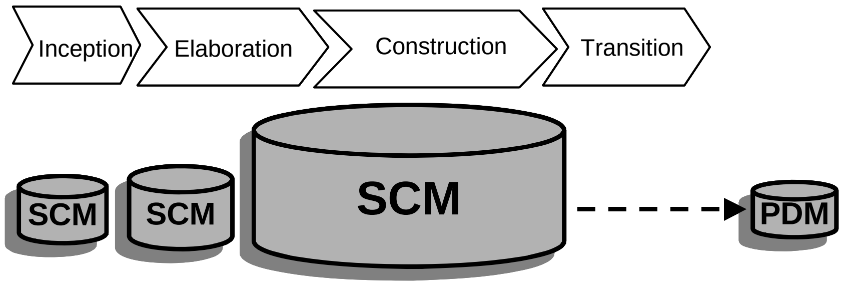

Figure 4. Processes and information storage for a software product in one increment

2.3 Standards

Standards and de facto standards vary considerably, in their scope, in their purpose, in the formality of their acceptance, their use, etc. With respect to PDM and SCM systems we can classify standards as those used for information exchange in its broadest meaning, or standards, which specify processes in particular, domains. Further, there are standards, which are applicable to SCM only or to PDM only, or standards, which are valid for both PDM and SCM and, in many cases, for other domains. Several CM standards were acquired by SCM. Finally, there are standards which can be directly implemented by software (typically the implementation of particular protocols or the management of particular data formats), and standards which involve human activities and can possibly be supported, but not automated, by tools (usually process-related standards).

PDM and SCM systems usually consist of several tools that exchange data. As these tools have neither common data nor a common information model and exchange of information is one of the major problems in their use.

For PDM there exist standards as ISO 10300 STEP [13], and relating standards as ANSI/EIA-649 [10] Non-consensus Standards for CM. Although PDM uses many standards, there are no standards that are exclusively intended for PDM systems.

Many standards are closely related to PDM and originate from PDM-related requirements.

No explicit standards exist for SCM except related standards for CM such as ISO 10007 Guide Line for Configuration Management [12], IEEE STD 1042-1987 Guide to Software Configuration Management [6] and IEEE STD 828-1998 Standard for Software Configuration Management Plans [7].

There are different standards and models for different Product Life Cycle Management (PLCs). Some standards addresses the life cycles of systems closely related to PDM and SCM, e.g. ISO/IEC FDIS 15288 Systems Engineering – System Life Cycle Processes [8].
For integration purposes no standards exist today.

2.4 Conclusion

From a system level, there are requirements on managing the whole product irrespective of its contents of hardware and software components, i.e. interoperability in the information flow. The development processes for hardware and software development, although similar, distinguish on a detailed, practical level. SCM and PDM have different production phases; PDM with high cost, long lead-time, and another organization involved, and SCM short and cost effective with no other than the developer team performing the product manufacturing involved in the production phase. PDM-related and SCM-related standards in CM exist, but they are too vague and too little integrated in PDM and SCM to be used as a common integration factor between PDM and SCM.

3. Tools and Technology View

In a well-integrated development process we need tools that cover all development cases of both software and hardware development. The question arising is: Is it possible to use one of the tools or must we use both PDM and SCM tools? To be able to answer this question we discuss some basic functionality in the tools: data representation, version management, management of distributed data, product structure management, process support, and document management.

3.1 Data Representation

The information in a PDM system is structured to follow an object-oriented product information model. Objects are of two different kinds: business items and data items. Objects used to represent parts, assemblies, documents etc. are designated business items. A business item contains attributes and metadata. A PDM system also manages files. A file is represented in the database as a data item. The metadata that provides additional information about data (file) is separated from the content or actual data (file). Separating business items from data items makes it easier to manage heterogeneous data. Several business items can reuse a data item, which is not possible in a standard file system. Figure 5 illustrates the data representation of documents. The Cylinder consists of two different documents, the CAD model and the specification, represented by a business item each with different metadata. The actual document or file is represented by the data item and is related to the business item, e.g. the Specification Large can have the file Spec_can.doc related to it.

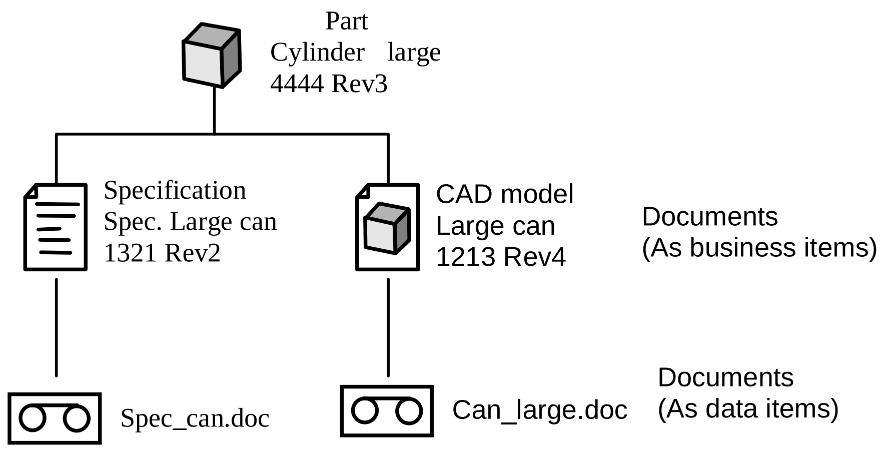

Figure 5. Data Representation of documents

The basic elements SCM deals with are files and directories in a file system. Metadata for a file is stored within the file and not in a separate database. Certain SCM systems use a similar paradigm as the PDM systems with a database containing metadata and files placed outside the database, but they do no have defined product structures.

Since PDM and SCM have different data representations, their usage in the other domain is limited.

3.2 Version Management

In PDM systems, the versions of business items are called revisions and are organized in sequential series. The business item contains metadata, denoted attributes. PDM supports customized attributes. Major changes of business items are tracked by revisions manually transformed by the user. Different revisions of a business item are connected by a relationship, the revision-of relationship. A PDM system may contain many other relationships, which may have one or more attributes. If a data item is changed, it may be checked in and out several times without creating a new revision. Versions are used to manage the sequence of data items but are usually not visible to the users. Only one user at a time can update a file, i.e. there is no support for concurrent engineering.

Versions in SCM form a graphical structure (see Figure 6).

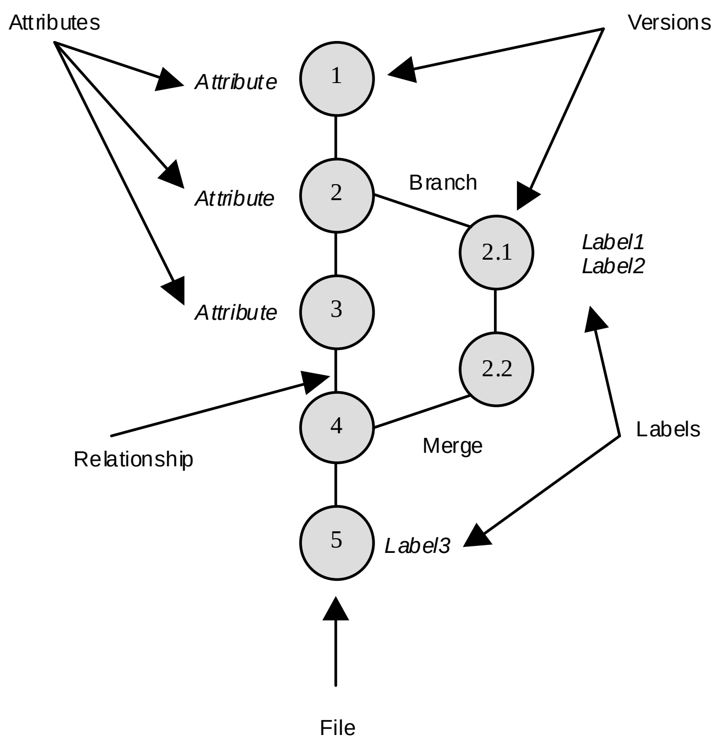

Figure 6. Version management in SCM

SCM provides support for concurrent engineering: several versions of a file can be developed simultaneously in branches, which may be merged together again if needed. Each time a file is checked out and in, a new version is created. This corresponds to a version in PDM. In SCM, however, versions are visible to the users and are used frequently. A version of a file can be marked with attributes. Versions are often marked using a special attribute called tag or label. Labels almost correspond to revisions in PDM. In SCM there is no support for relationships. Because software developers usually work on the same file at the same time, the branch and merge mechanism is very important.

In spite of in principle similar mechanisms, the version management in PDM and SCM is quite different and would require significant changes in order to support the other domain: SCM is missing advanced management of attributes and relationships, PDM is missing advanced version management.

3.3 Management of Distributed Data

Both PDM and SCM systems support distributed development by enabling replication of data. There are however differences. In the PDM system only metadata or metadata and the files are replicated to other sites as illustrated in Figure 7.

A typical PDM tool has a master server, often denoted corporate server. This server contains common information such as access rights for other servers, and locations of them in the network. Irrespective of where in the network the file is located, it is locked when it is updated. A distributed lock mechanism controlled by the master server prevents the checkout of a file by two users at the same time. Such solution does not permit full parallel development, a strategy commonly used in software development.

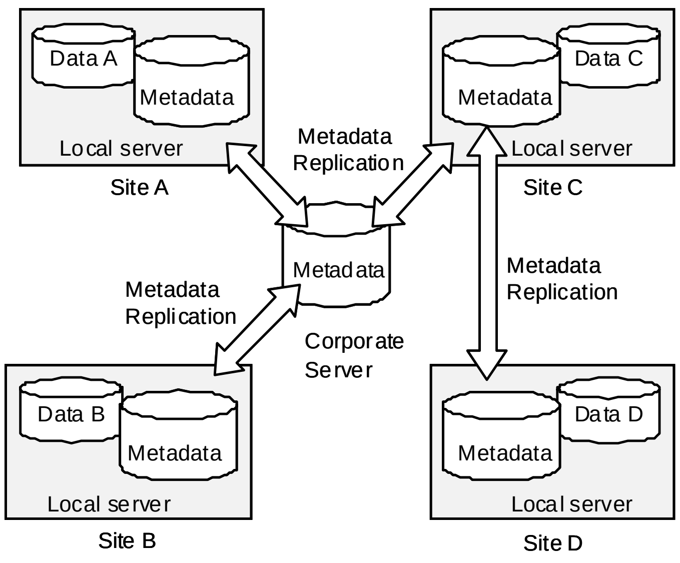

Figure 7. Server replication in a typical PDM environment

The SCM environment replicates the total file including the metadata. SCM tools, the servers replicate data between two nodes, using a peer-to-peer protocol. Any structures of servers can be built by connecting servers to each other. An example with four servers is depicted in Figure 8. These examples show that the PDM mechanism is not appropriate for distributed software development. Similar is valid for SCM tools: in cases in which metadata is more often manipulated the SCM solution is not the most appropriate.

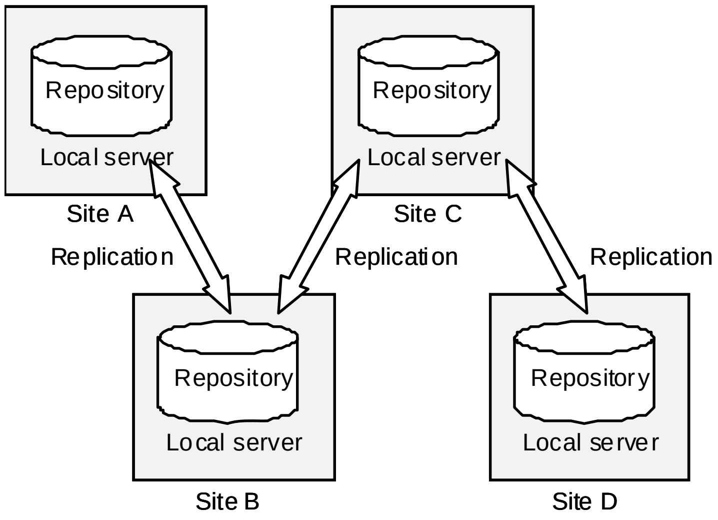

Figure 8. Server replication in a typical SCM environment

3.4 Product Structure Management

Product structure management is a basic and fundamental functionality in PDM systems [5]. The product structure is a configuration of parts connected by relationships. A database model supports the building of a product structure. Figure 9 shows an example of a product structure of a bicycle. The structure is a so-called quantified Bill-Of-Material (BOM) used in production for collecting all objects and information.

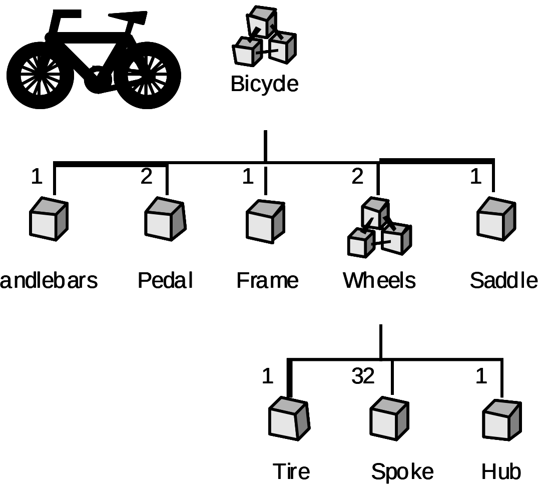

Figure 9. Example of a product structure in a PDM tool

Software uses a similar approach in object-oriented design and programming. SCM tools however do not explicitly address and support product structures. Only rudimentary support in form of directories in a file system is available for use in building a hierarchical structure. SCM tools provide support for managing these structures.

3.5 Process support

Workflow management is a critical part in the product definition life cycle to ensure that the right information is available to the correct users at a proper time. It includes defining the steps in the process, the rules and activities associated with the steps, the rules for approval of each step, and the assignment of users to provide approval support. Workflows in PDM systems provide the mechanism for modeling and managing defined processes automatically. Data can be submitted to the appropriate workflow for processing. Appropriate information is routed automatically.

Some SCM tools incorporate similar functionality or provide it using tools tightly integrated. However, in most SCM tools the support consists of triggers only, which can execute scripts written by the users.

From a system level perspective, the process support is essential. Processes as change management, baseline management, and document approval are examples on processes useful for not only PDM and system level, but for SCM too. In principle the support provided either by a SCM tool or PDM tool can be used in both domains. The problem that should be solved is the integration of the tools, which are supposed to be triggered by events from the workflow management tool.

3.6 Document Management

PDM has built-in functionality for managing documents such as queries, viewing, and access control. Document management is an important function in the PDM systems. This function is not available in SCM. However, developers prefer to work in their integrated development environment; software developers prefer to keep documentation in SCM although SCM does not provide efficient support.

3.7 Conclusion

From the analysis of basic characteristics of PDM an SCM tools we find that there are similarities in them, but that the underlying concepts are quite different. PDM tools support, document management, product structure management, distributed development and awareness of changes of documents. Of these features an SCM tool does only support awareness of changed documents and an effective replication between sites. On the other hand SCM tools support concurrent engineering on file level, and replication without locking on file level. A PDM tool does not support these features. Using PDM tools for development of software would be very difficult and inefficient. Using SCM for hardware products would be practically impossible.

4. People and Cultural View

The cultural differences between hardware and software development groups play a much more important role than visible when building integration between PDM and SCM. First of all, both domains are huge using completely different tools. Secondly, users from the different domains do not have knowledge about the other domain.

Low communication between the domains causes poor understanding of each other's problems and requirements. Thirdly, users from both domains believe that the system they use can manage all situations from the other domain [2], [11]. Fourthly, PDM and SCM users are often located at different departments within the company. Their geographical separation can increase the gap in their understanding of the other group. Fifthly, the hardware designer uses a lot of documents to describe the product. These documents are transferred to the production and manufacturing part used of another person to produce the actual product. Hence, the hardware designer focuses on documents. The software designer writes a lot of source code. The designer then generates the actual product, the load modules, with no other person involved. Hence, the software designers focus on source code more than documents and have small understanding of the importance of writing documents.

4.1 Terminology

Since both PDM and SCM are domains evolved independently from each other and no common standard occur some of their terminology differ. Different terminology is used for the same concepts or different terms for similar concepts; For example, in PDM configuration control is the definition and management of product configuration, while in SCM it means the control of changes to a Configuration Item (CI) after formal establishment of its configuration documents. SCM uses versions for all changes, but PDM distinguish between minor changes, designated versions, and major changes, designated revisions. Another example is the term efficiency used in PDM, which is a concept similar to change management in SCM.

4.2 Conclusion

Since hardware and software designers are focusing on different activities, they have both low knowledge and understanding for each other's requirements due to organizational, cultural, and domain specific behavior. On top of this, the terminology is almost the same but with different meanings. For integration purposes, terminology and cultural differences are key factors to highlight. A common understanding for both domains and terminology is essential to provide when integrating these domains.

5. Integration

From the analysis we have seen that PDM and SCM tools cannot replace each other. We have also seen that the software and hardware development processes differ and cannot be directly replaced. PDM and SCM are complex tools themselves and often very difficult to successfully deploy and utilize even for development of pure hardware or pure software products. The things are getting more complicated for development of systems that include both hardware and software components. Due to their differences many integration attempts have succeeded only partially [2].

Usually the development of such systems is divided into development of components, in particular separated in development of hardware components from development of software components. This separation can however not be complete; there exists common system requirements and the components must in the end be integrated into the final system.

To be able to provide full support for the entire development process, the tools should support the development of hardware and software components, and in addition to this a seamless integration of information should be provided.

Full integration can be achieved through integration of processes, tools, and by achieving a common understanding between developers of the software components, hardware components and integrators of the final system.

5.1 Process integration

To successfully integrate software and hardware development processes into a unique process we must: (i) identify the possible integration points in which the information can be exchanged, (ii) identify which information will be exchanged and in which form, (iii) provide the tools that automatically can exchange the information, (iv) find out which information is common and which system should be the primary repository of that information. For example in a total process, initial phases (requirements specification, overall system specification and design) can belong to a common process, the detailed design and implementation of components can be separated processes managed separately by PDM and SCM, and the final integration can again be a part of a common process, as illustrated in Figure 10. This integrated process is described in detail in [3].

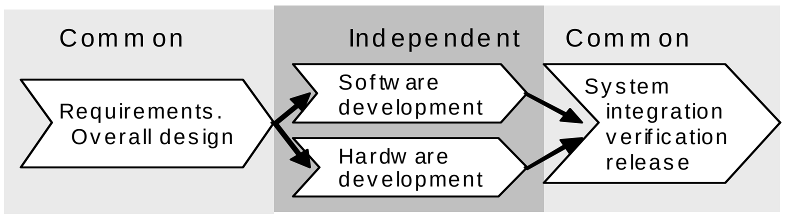
Common
Independent
Common
Requirements. Overall design
Software development
Hardware development
System integration verification release

Figure 10. Integrated process

5.2 Tool integration

The identification of the integrated process will lead to decisions, which tools can be used and which are the integration requirements. Further a policy for the integration of the tools should be decided: integration can be achieved through a common information model (tight integration), or in a loose way in which the tools preserve their internal structure, but interpolate through Application Program Interfaces (APIs), integration languages and commands, or web-based services and components [3]. A tight integration is based on a consistent information model, which makes simple interoperation between the tools. However, a tight integration requires a lot of efforts to achieve agreement about a common information model. Since different tool providers want to keep their advantages on the market, they usually are not willing to change their internal representation to standard formats and models. Instead of that they focus on enabling integration with other tools. In a loose integration there is not one common information repository, but the same data may be saved in several, different, repositories. For this reason a policy for information management must be decided. For example: (i) which system should be the main archive for documents (drawings, source code, etc.), (ii) which system should manage the product structure and the revisions of all products included, and (iii) which system will manage metadata of delivered products. In particular the problem of version and configuration synchronization might be problematical.

Figure 11 shows an example from a case study of loose integration of two tools aimed to make it possible for the system managers to continue to work in their PDM tool (in this case eMatrix) and the software developers to continue in their SCM tool (in this case Clear Case). Both tools store and manage their "standard" information, but they also retrieve some (pre defined) information from the other tool and present it to "its" users.

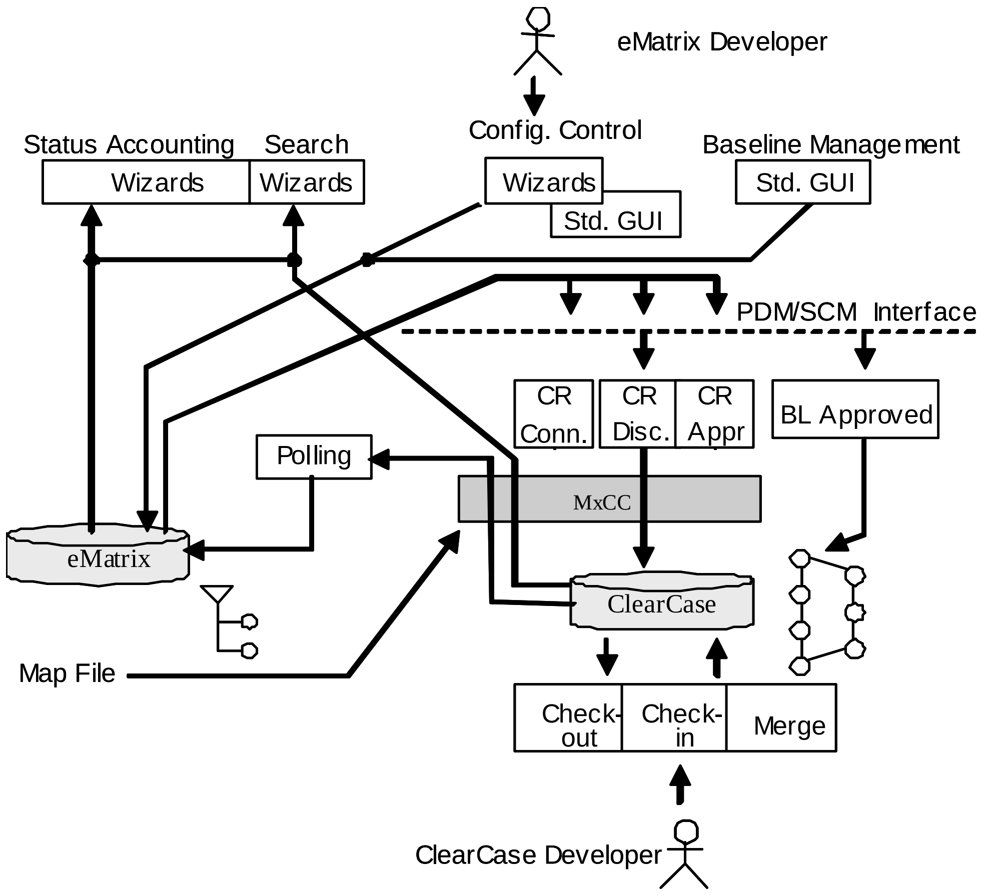

Figure 11. PDM and SCM integration example

Another case from a Swedish company with a complex integration is shown in Figure 12. Information exchange between different tools from SCM and PDM follows a complex pattern, which makes it difficult to understand where the original information is placed, which data are read-only, which can be modified. It is also quite unclear which repositories should be updated when particular data is changed. The process is in particular complicated as the information transfer is performed half automatic.

This case is also interesting as it clearly showed the results of cultural differences of the developers. Earlier, the company used SCM tools for all development activities but decided to introduce also a PDM tool. However, due to bad knowledge of what PDM actually is, a document management tool was bought instead (Documentum). The need for PDM functionality remained and new tools had to be bought (PVCS Tracker, SAP R/3) resulting in a complicated structure of different tools.

Independently of which integration strategy is chosen, the integration process is very complex and it often requires considerable knowledge of both systems and technologies. For this reason, many end-users are not capable to perform the integration alone and need the assistance of the vendors or consultant companies providing such service.

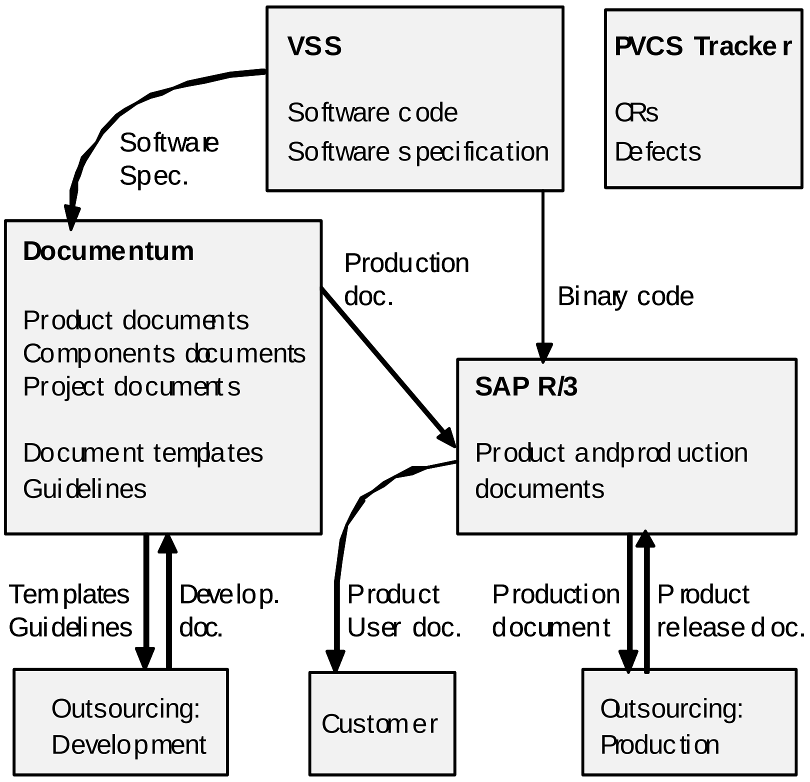

Figure 12. Example of a complex integration of PDM and SCM tools

5.3 Common Understanding

Depending on the process and integration of the tools, the developers will have a need to learn about the other domain. In a tight integration with a common information model, the developers must get familiar with the entire process; in a loose integration (like the case showed in Figure 11) most of the developers will work in their environment using their normal tools. In any case, since the final product is a result of integrated hardware and software components, it is important that the developers from both domains build up understanding of the entire process. This means that it is not enough to integrate the tools and the processes, the people involved should also pass through an “integration process”.

In [3] a case is discussed, which did not succeed to integrate a SCM and PDM tool. The integration did not succeed because the tool vendors focused only on technical interoperability issues building automatic import/export tools, but did forget the two other important factors. First, they neglected the process issues – which actions and which tools are performed in which phases. Second their decision was that the user interface, the terms, and in general the overall philosophy should follow PDM standards. This caused large problems for software developers, which did not, understood the PDM concepts, and were not willing to accept them.

6. Conclusions and Future Work

In a rapid expansion of computer-based systems developers from different engineering domains are enforced to work together. This collaboration enables significant improvements when complex products are developed and manufactured, i.e. when the development process has high demands on efficiency and quality. However, the challenges to achieve this quality are many, not only in the technologies of the particular domains but in the coordination, interoperability and integration of these domains. A characteristic example of such challenges is the integration of PDM and SCM tools, which provide information and management support for the development and maintenance of hardware and software assets, respectively. Many companies developing and manufacturing products that include both software and hardware components face this problem of building up an integrated support of these products. The initial steps towards an integrated development and production environment and an integrated process are painful; there are a number of unsuccessful or only partially successful attempts to integrate functionality available from these tools. In this paper we have shown why such integration is so difficult. First, the functions that the tools from these domains provide are in general similar but in principle very different. Second, the pure technical solutions for integration are not sufficient; a total coherent and integrated process is as important as the technical ability of integration of the tools. Finally we have experienced that the cultural differences between domain engineers play an important role. A lot of efforts must be put in removing cultural barriers, in education and in building common understanding to make it possible to introduce a new integrated support for the entire development process. Our findings are also that loose types of integrations in which developers can keep their old tools and local processes are more feasible than tight integrations requiring a new information model and entirely new processes. Again, the reasons are not only of technical nature, but very much of cultural.

We will continue our work on how to integrate commercial tools in practice. Within Ericsson a project recently started with the aim to integrate commercial PDM and SCM tools. We will be part of this work.

Another work is to see how product data and tools for both production and design can be integrated. One overall goal is to develop enabling technologies to support smooth integration of different tools, and to support concurrent updating of the product data in order to allow people to work in parallel. In this work we will investigate the possibility to introduce techniques from the software development field into the product data field, which may give rise to new, more flexible, ways thinking about the tools in that area.

7. References

[1] U. Asklund, I. Crnkovic, A. Hedin, M. Larsson, A. Persson Dahlqvist, J. Ranby, and D. Svensson. *"Product Data Management and Software Configuration Management - Similarities and Differences"*, The Association of Swedish Engineering Industries, 2001.

[2] Crnkovic I., Asklund U., and Persson Dahlqvist A., *"Implementing and Integrating Product Data Management and Software Configuration Management"*, ISBN 1-58053-498-8, Artech House, 2003.

[3] Crnkovic I., Persson-Dahlqvist A., and Svensson D., "*Complex Systems Development Requirements - PDM and SCM Integration*", IEEE Asia-Pacific Conference on Quality Software, IEEE, 2001.

[4] Estublier J., "*Software Configuration Management: A Roadmap*", In Proceedings of 22nd International Conference on Software Engineering, The Future of Software Engineering, pp. 279-289, ACM Press, 2000.

[5] Estublier J., Favre J-M., and Morat P., "*Toward SCM/PDM Integration?*", In Proceedings of Software Configuration Management SCM-8, Lecture Notes in Computer Science, nr 1439, pp. 75-94, Springer, 1998.

[6] IEEE STD 1042 - 1987, Guide for Software Configuration Management, 1987.

[7] IEEE STD 828 - 1998, Standard for Software Configuration Management Plans, 1998.

[8] ISO TCI194/ SC4/WG5, S. P. 1., Overview and fundamental principles, 1991.

[9] Kroll P. and Kruchten P., *"The Rational Unified Process Made Easy"*, ISBN 0-321-166009-4, 2004.

[10] National Consensus Standard for Configuration Management, A. N. S. I., ANSI/EIA-649-1998, 1998.

[11] Persson-Dahlqvist A., Crnkovic I., and Larsson M., "*Managing Complex Systems - Challenges for PDM and SCM*", In Proceedings of International Symposium on Software Configuration Management, SCM 10, 2001.

[12] SIS, S. S. I., Quality management Systems - Guidelines for configuration management, ISO 10 007, 2003.

[13] STEP Part 1, Overview and fundamental principles," ISO TCI194/ SC4/WG5, 1991.

[14] Svensson D. and Crnkovic I., "*Information Management for Multi-Technology Products*", International Design Conference - Design 2002, IEEE, 2002.

[15] Telefonaktiebolaget LM Ericsson, www.ericsson.com, 2004.

PAPER C
IMPORTANT FACTORS FOR A SUCCESSFUL INTEGRATION OF PRODUCT DATA MANAGEMENT AND SOFTWARE CONFIGURATION MANAGEMENT SYSTEMS

Annita Persson Dahlqvist
Ericsson AB, Mölndal, Sweden
Annita.Persson.Dahlqvist@ericsson.com

Technical report: MRTC report ISSN 1404-3041 ISRN MDH-MRTC-194/2005-1-SE

Abstract

Since PDM and SCM have been developed in their respective domain solving the domain specific requirements using different technology; on a higher level they seam to be similar in functionality, support and infrastructure. The similarities and differences, however, are found on practical lower levels such as in the product, evolution, and process model. The main characteristics of PDM and SCM are described more in detail. We have found in our investigations, that three factors are important to achieve a successful integration: processes, tools and technology, and people and culture. These three factors are discussed more in detail.

In addition, the report presents two case studies done at Ericsson Radio Systems AB and Industrial and Financial systems. The case studies are focusing on how the companies are using PDM and SCM, their processes, any need for integration between PDM and SCM, and conclusions. The second case study was preformed later and it is not included in the book. The case is used for validation of hypothesis: the new elements in this case study are the starting assumptions that are based on the experiences and findings from the pervious case studies.

1 Introduction

Many high-end and complex products are developed by means of different technologies based on both hardware and software components. Examples on complex products are such as mobile phones, cars, and aircrafts. The consequence for these products is that there is no pure hardware development; even the companies that develop hardware products must consider development of software. The final product is a result of tight integration of hardware and software components. In order to achieve an efficient integration, the entire development process including both development of hardware and software must be synchronized and coherent [1, 2], and adjustments of all included processes are needed [3, 4]. Thus, the hardware and software development processes demand integration points to support the system level. The decision whether a specific function should be implemented in hardware or software may come late in the project and may even change during the product's life cycle. When the border becomes vague [5] it is no longer possible to keep the development organizations separated and to use different life cycle processes, but they should be integrated. However, the requirements for such integration point out a number of problems: process adjustments (including information exchange, data access and information flow), infrastructure support, tool integration, culture differences between the stakeholders, etc.

Since the hardware and software development processes have evolved in parallel, also their respective supporting tools have evolved in parallel [3, 6, 7]. Product Data Management (PDM) systems is used for managing hardware product information [8, 9]. Software Configuration Management (SCM) systems aim to manage software product information [10, 11, 12, 13, 14].

This report gives a summary of selected topics from the book *Implementing and Integrating Product Data management and Software Configuration Management*, [2]. Furthermore to this, it describes one additional case study. The purpose of this study is the validation of our assumptions based on the findings from the pervious studies.

The remainder of this report is organized as follows. The technical principles and key functionality of PDM and SCM are discussed in section 2. Further, we summarize the weight for the pros and cons of using PDM and SCM supporting complex product development and maintenance. We continue to discuss the structure of complex products and the complex product lifecycle management process in section 3. In section 4 we discuss culture differences. Section 5 provides scenarios in an integrated environment. Section 6 summarizes the different case studies we have performed. Further, we report from two case studies, one case study performed at Ericsson AB (former Ericsson Radio Systems AB), and Industrial and Financial Systems. Finally, section 7 concludes the report.

1.1 Research methods

The first step is to understand these domains, and to do this we have analyzed the domain specific processes, and tool functionality. In this paper, we have analyzed the main technical characteristics of PDM and SCM, i.e. the key functionalities and relations between them, and we have identified similarities and differences. This has been performed by literature study, use of PDM and SCM products, and discussions with researchers and practitioners, including tools' providers and tools' users.

In addition, we have performed several industrial case studies of PDM and SCM usage [1, 2]. We have focused on the tools and technologies exploited their interoperability, and culture differences, which cause problems when integrating PDM and SCM [15, 16].

The case studies have been performed in form of interviews. A number of questions were formulated based on existing models, knowledge, and theories. Several companies, which business segments were relevant for our study, i.e. those that develop, produce and maintain complex products, have been selected for case studies. The questions were sent to the companies to inform about the interview questions. The interviews were performed either by visiting the companies and having discussions with different stakeholders knowledgeable in PDM and/or SCM or by telephone interviews. If further questions were to be asked to clarify specific answers or find more information, the questions were sent by mail to the contact person in the company or a telephone meeting was set up to have further discussion. The results from the interviews were archived and analyzed. All case studies were reported in draft reports and reviewed by the interviewees. In addition, several researchers knowledgeable in the PDM and SCM area, were reviewing the reports. Since all interviewees did know the cases should be published, we cannot assure the truthfully of the answers. The interviewees could use an answer more positive for the company. However, we have analyzed their statements with the observed practice. In addition, we have compared answers from different stakeholders.

During the analyses of the case studies, it becomes more and more visible that the three parts; tools integration and interoperability, development processes, and cultural differences, are the vital factors for a successful integrated infrastructural support. The last interview started from this hypothesis, and was used for the hypothesis validation. This iterative approach in reaching the hypothesis is shown in Figure 1.

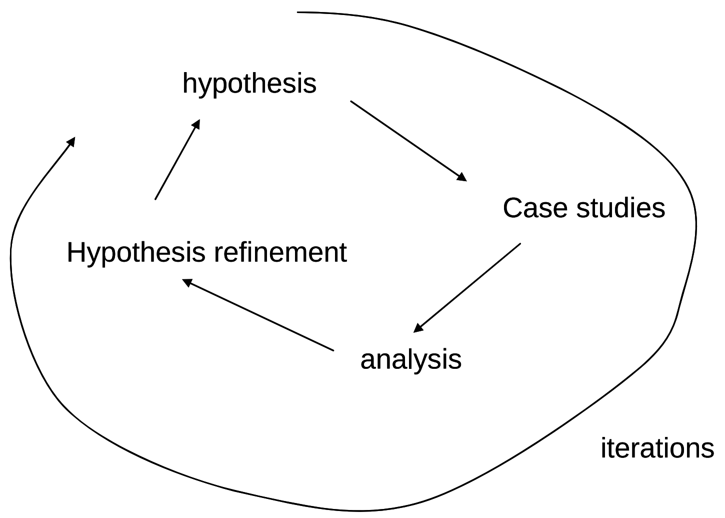

Figure 1. Data representation in PDM systems

2 Technical Principles and Key Functionality

In this section we compare the functions of tools within the two domains, both on underlying principles and with respect to most important functions of the tools.

2.1 Comparison of Technical Principles

We discuss four fundamental areas of each in the PDM and SCM domains respectively, are compared. The four areas are:

- *System architecture* describes the architecture of respective PDM and SCM, their infrastructure, and the abilities of integration with other tools.
- A *product model* is an information model used to describe the structure and behavior of a product managed by the system.
- The *evolution model* manages changes during the product's life cycle and is related to version management.
- The *process model* is described by a set of states ad rules for passing from one state to another.

2.1.1 System Architecture

Most PDM and SCM tools use a client-server architecture, where the server contains the database in which all data is stored. The data is stored following a certain data representation implementing a storage data model. Many servers are used to provide effective support for distributed development. The architecture includes the strategy for server use (which data is stored in which server), the client-server, and server-server communication, and synchronization schemas.

To show the important architectural elements we look at data representation, data replication, and application integration. These architectural elements are less described in literature. We will end with a short discussion of some models, product, evolution, and process, due to already described in literature [6]. All topics are described in more details in [1, 2].

Data Representation

The information in a PDM system is structured to follow an object-oriented product information model [17, 18]. Objects are of two different kinds: *business items* and *data items* see figure 2. Business items are objects used for representing parts, assemblies, documents etc. A business item contains metadata and attributes. *Metadata* describes properties of the product data. An *attribute* consists of a value and a name, and may be customized. The actual data is stored in files and represents in the database as data items. Separating business items and data items provides support for managing heterogeneous data and enables replication of metadata separately. One business item can be related to several data items. *Relationships* are used between business items and data items. Business items can build a tree structure including several levels of business items, see figure 2. A data item is always related to a business item, and represents a leaf in the tree structure. Attributes can be defined either on objects or relationships.

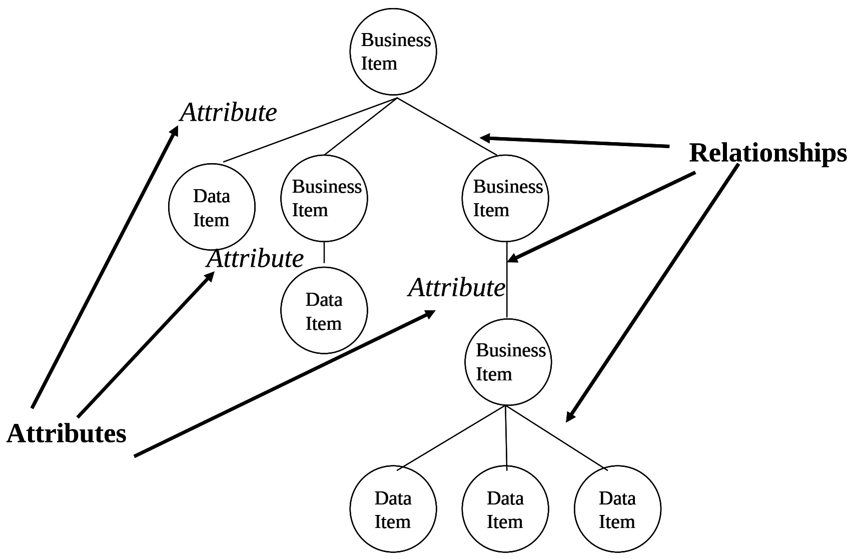

Figure 2. Data representation in PDM systems

In the SCM tools all kinds of file types and objects represented as a file or directory may be managed and stored. In most of the tools, the two types of files, source or binaries, are managed differently. For source files SCM provides additional support such as showing differences between different file versions or enabling interactive merging of two file versions. Metadata for a file is stored within the file and not in a separate database. Certain SCM systems use a similar paradigm as the

PDM systems with a database containing metadata and files placed outside the database, but they do no have defined product structures.

Since PDM and SCM have different data representations, their usage in the other domain is limited.

Data replication

Both PDM and SCM systems support replication of data, but replication is implemented differently. In the PDM system the replication can be set up when installing the system by either replicating the metadata only or metadata and the files. In a typical PDM tool the master server contains common information. When a user checks out a file, a locking mechanism is distributed to prevent concurrent check out of the same file.

In most SCM tools, the replication functionality was implemented as an add-on feature long after the standard systems were developed. In such tools, it is impossible to manage metadata and files separately. These tools replicate the total file including the metadata using a peer-to-peer protocol. For concurrency control, SCM systems use locking on branch level still possible to create a new branch from one of the owned branches, if needed.

This shows that the PDM mechanism is not sufficient for distributed software development, and in cases in which metadata is more often manipulated the SCM solution is not the most appropriate.

Application Integration

A PDM system is usually integrated with various applications. Data is gathered from the applications and exchanged. PDM has standards defining transfer protocols to enable exchange of data with different formats. Integrations range from the simpler; where the application is launched, to the tighter, where the PDM system retrieves information from the applications. PDM tools are often the central process that initiates other activities in other tools.

An SCM tool can be used either as a stand-alone tool, or as a set of tools. The SCM tools are often designed to provide information with other information and data. Plain files are used for exchange of data. Many SCM tools are integrated in other tools, such as IDE, and for this reason include APIs with basic SCM functions. SCM tools are more passive and initiate not other activities in other tools.

2.1.1 Evolution Model

The evolution model provides a framework for managing changes during the product life cycle and is related to version management.

PDM distinguishes three different concepts of versioning: *historical versionin*g, *logical versioning*, and *domain versioning*. Historical versioning is conceptual similar to SCM versioning, managing revisions/versions of a product, without branch and merge features. Logical versioning manages versions of parts such as alternatives, possible substitutes, or options. Domain versioning is a presentation of further views of the product structures (e.g. as-planned, as-designed, and as-manufactured) used by different stakeholders during the product life cycle. These views are fundamental in PDM tools.

Historical versioning in SCM originate from differences in the natures of the products: software may be changed more easily than hardware. Thus, SCM must manage versioning in a more sophisticated way than in PDM. Versioning in an SCM tool must always include functions for creating and merging branches. There is no logical versioning in SCM, since variants are managed by using branches or conditional compilation, which are not clearly visualized using the product structure. SCM tools do not support domain versioning. Although views are used in SCM tools, they are related to create configurations by selection of consistent versions of the files included in a specific configuration. This is used to create private workspaces and to build the product.

In section 2.2.1 we describe version management in PDM and SCM more in detail.

2.1.2 Product Model

A product model is an information model used to describe the structure and behavior of a product managed by the system. A PDM system provides a support for building product models. The basic principle of product modeling in PDM is the composition relationship, used to form tree structures, referred to as product structures. The product structure is visible and edited by the user. A hardware product has a physical existence and consists of physical parts, and thus represented by a part structure.

In SCM product modeling is week, and the tools do not manage a product model. This originates from the nature of software products. During the software life cycle, the software is transformed through different structures, such as software architecture developed during design phase, development structure used during implementation phase (source code and related documentation), and the software delivery package. These structures are not physical, but virtual, and can be easily changed. As SCM tools are focused on the development phase, they usually have certain support for managing developing structures. Only a few SCM tools include a customizable data model. Most SCM systems structure information by using the file and directory structure used in the operating system.

The extensive support for product model management in PDM and its absence in SCM is one of the largest technical differences between PDM and SCM.

In section 2.2.2 we describe product structure management in PDM and SCM more in detail.

2.1.3 Process Model

PDM systems have two process-related concepts: object states and workflows. The object-state defines the life cycle of an object. Workflows are based on description of the process, its activities, their sequence, and relationships between them.

In many SCM tools the process models are based on state transition diagrams (STDs). Some SCM tools provide process support similar to the workflows in PDM. Most SCM tools provide triggers to implement a process, which can activate scripts at certain occasions.

The process models for PDM and SCM are conceptually similar.

In section 2.2.4 and 2.2.6 we describe change management and workflow and process management in PDM and SCM more in detail.

2.2 Comparison of Key Functionality

Since both PDM and SCM provide infrastructural support for products (either hardware or software) they include a number of different functions needed for that support. The functions we refer to here are either overlapping in both PDM and SCM, or vital for one of the domains.

We discuss the following functions:

- Version management - support for managing different versions of an object;
- Product structure management – support for describing the product in a hierarchy structure;
- Build management – mechanisms for building software (compiling and linking) and keeping generated software up to date, preferably without unnecessary rebuilding;
- Change management – keeping track of changes introduced in the product and providing support for implementing changes in the product;
- Release management – packaging the product in a form suitable for distribution and generating documentation to inform users and developers of changes included in the release;
- Workflow and process management – support made available for the developers in following a certain process with specific activities;

- Document management –support for managing documents allowing users to store, retrieve, and share them with security and version control;
- Concurrent development – support for control simultaneous access by several users (either by preventing or by providing support);
- Configuration management and selection management – providing support for creation or selection of associated versions of different objects;
- Workspace management – providing each user with a private location in which the user can work in isolation under the control of the tool.

2.2.1 Version Management

In PDM systems, versions of business items are called *revisions* and are organized in sequential series. Different revisions of a business item are connected by a relationship. *Versions* are used to manage the sequence of data items usually not visible to the users. A changed data item may be checked in and out several times without creating a new revision of the business item. Figure 3 shows the connection between business items, data items, revisions, and versions. Only one user at a time can update a file, i.e. there is no support for concurrent engineering on a single object. When an item is checkout by a user, it is locked for other users from checking out the same version. When the item is checked in again, the new version is stored and the lock is released.

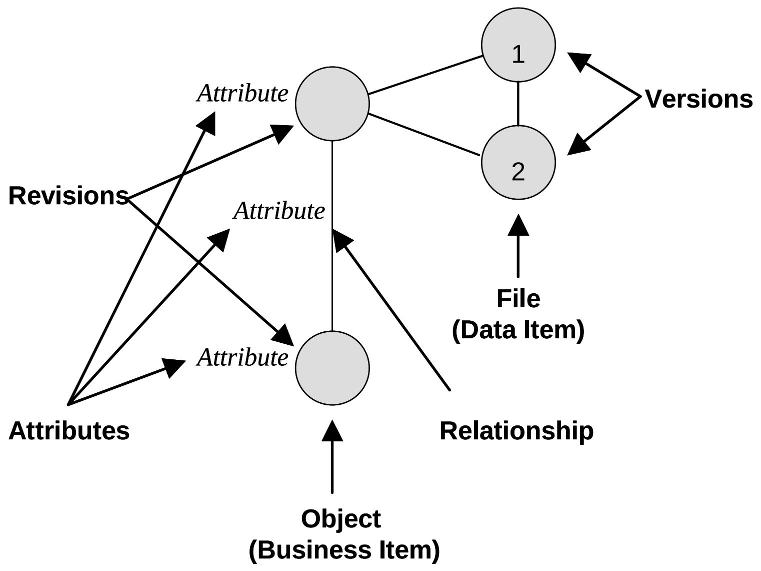

Figure 3. Version management in PDM

Figure 4 shows how business and data item are managed in a PDM system when they are changed. To be able to change a business item or the related data item, the business item has to be revised. The data item may be checked out, changed and then checked in again several times. When the update is ready, the business item is submitted into a new frozen revision.

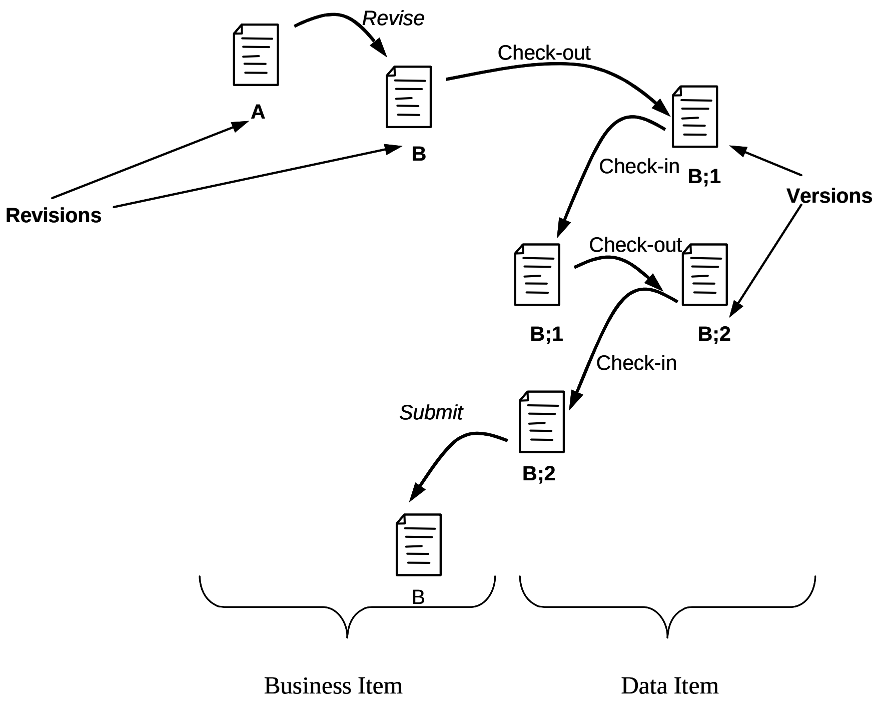

Figure 4. Check-in and Check-out of Data Items and Business Items

PDM supports concurrent development on a business item, but no support for merge, see figure 5. When one user revises the business item (revised to revision B in the figure), another user may revise the same business item but to next available revision (revision C in the figure). Both users may update the business item several times, and the business item is frozen when it is submitted. PDM does not support merge when the two business items are submitted. Users not aware of this basic functionality may loose their updates.

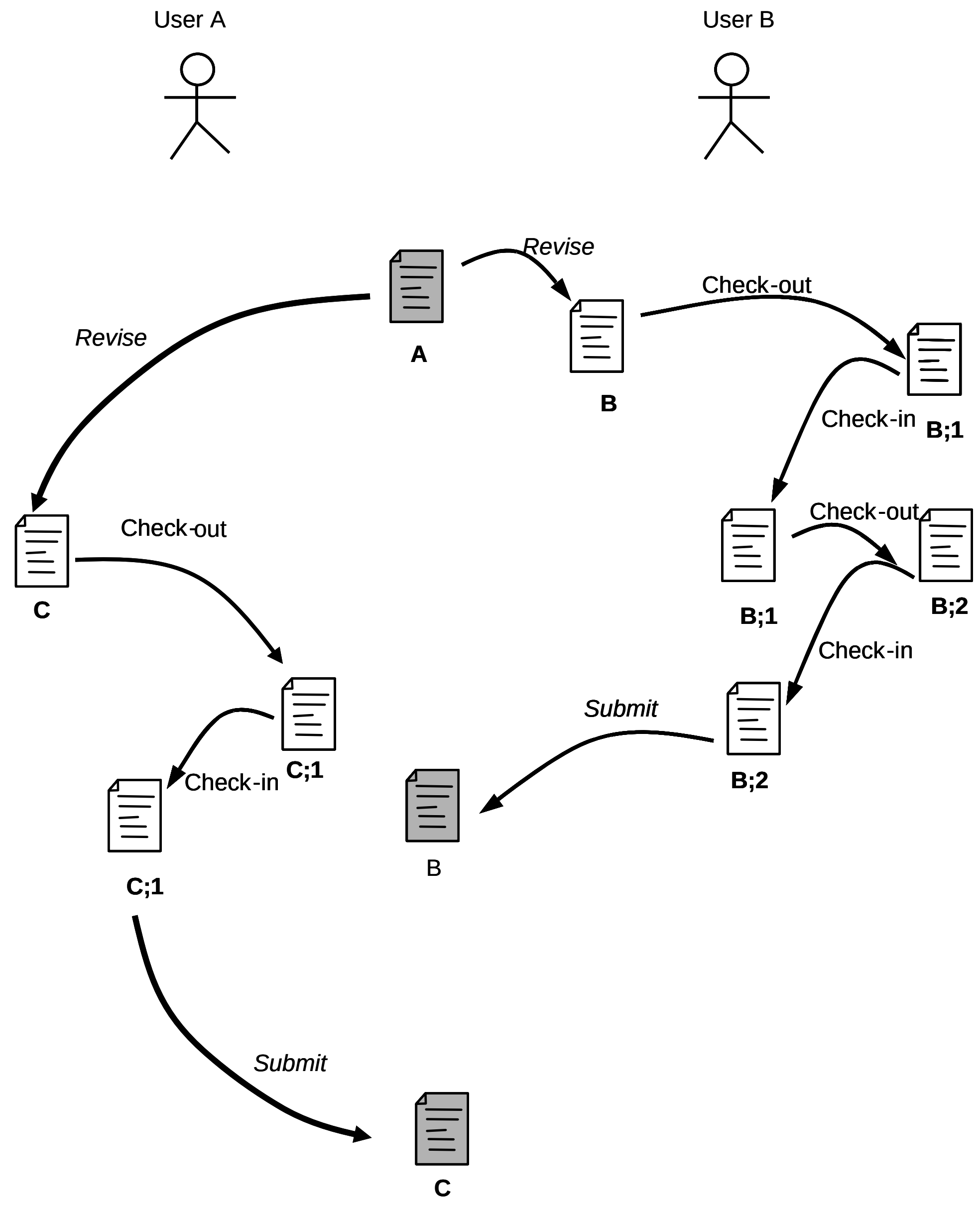

Figure 5. Concurrent development in PDM

Versions in SCM form a graphical structure (see figure 6).

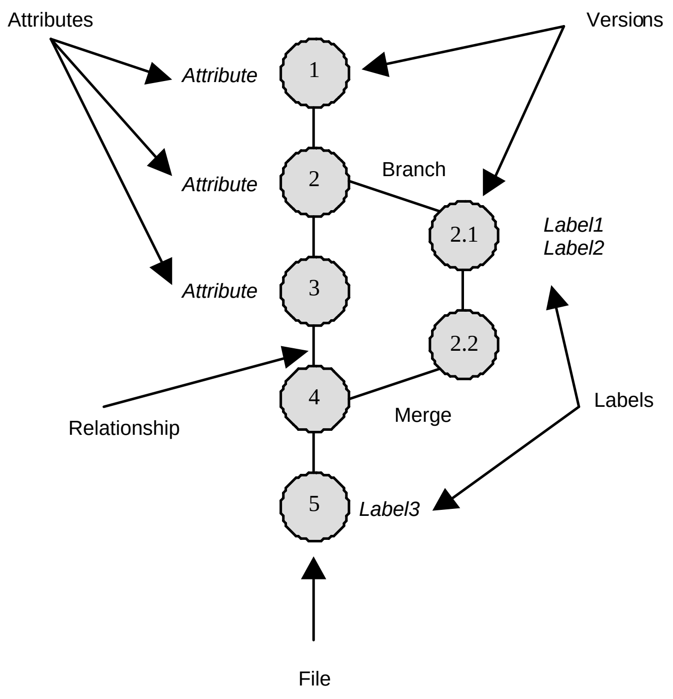

Figure 6. Version management in SCM

In SCM branches are used for several reasons; (i) adjustments to the file according to diverging requirements on the file e.g. different operating or window systems, or (ii) permitting concurrent development by supporting several versions of one file. Branches plays different roles depending on the reason for its creation, e.g. as the main line in the development process or the implementation of a change, bug-fix [19].

A *development strategy* [1, 2, 20] has to be chosen when and how often modifications of a system are to be made; either to bring about early integration of changes such that potential problems are discovered on an early stage, *optimistic strategy*, or to provide the developers with a stable working environment to avoid disturbance in their development work, *conservative strategy*. In addition, an *update strategy* [1, 2, 20] has to be decided, i.e. when a change affects other developers and who ensures the changes are used. Figure 7 shows how changes are promoted when using the optimistic vs. conservative update strategy. When using optimistic strategy, all changes are used immediately, and the opposite for conservative strategy.

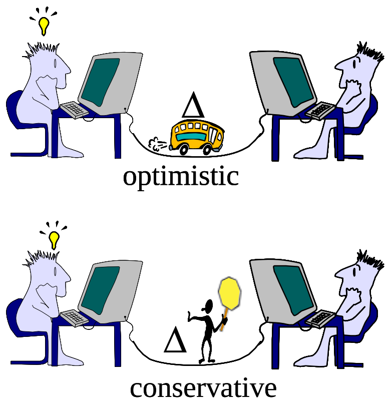

Figure 7. Update strategy based on [1, 2, 20]

In figure 8 one file is updated concurrently by two users. In the reserved checkout model, the optimistic development strategy is supported by using branch and merges without locking on versions when checking out. From the branch 1.1, user A is checking out the file and updates it. At the same time user B is checking out the same version. User A updates the file, and is ready with the changes before user B. User A decides to check in the file as 1.2. Then user B decides to check in the file, but is not allowed to check in before a merge with the changes made by user A is done. The user B can check in the file as version 1.3.

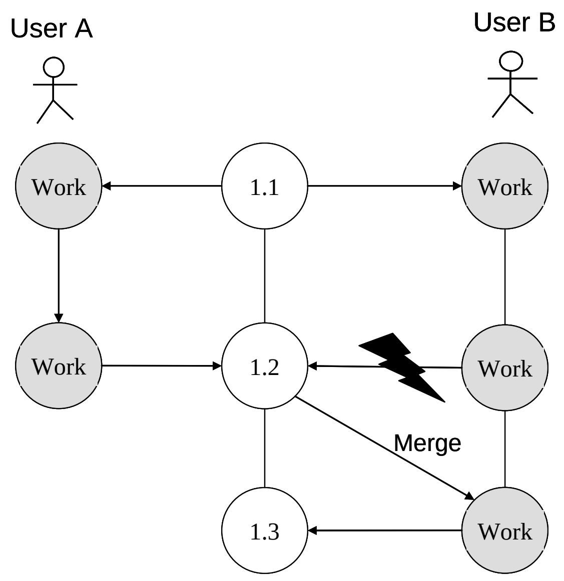

Figure 8. Concurrent development in SCM, an optimistic update strategy

A conservative development strategy is supported by using branch and merges with locking on versions when checking out, shown in figure 9. From the branch 1.1.1, user B is checking out the file and updates it. During the check-out, the version 1.1.1 is locked for other users to update, depicted in the figure as a padlock. The user A checks simultaneously out the version 1.2, and that specific object is locked (padlock in the figure 9). User A updates the file and checks it into version 1.2. The user B decides to merge his/hers changes into the changed version made by user A, version 1.2. This new versions including all changes is checked in to version 1.3. Concurrent development is possible to perform although the specific versions are locked in the repository.

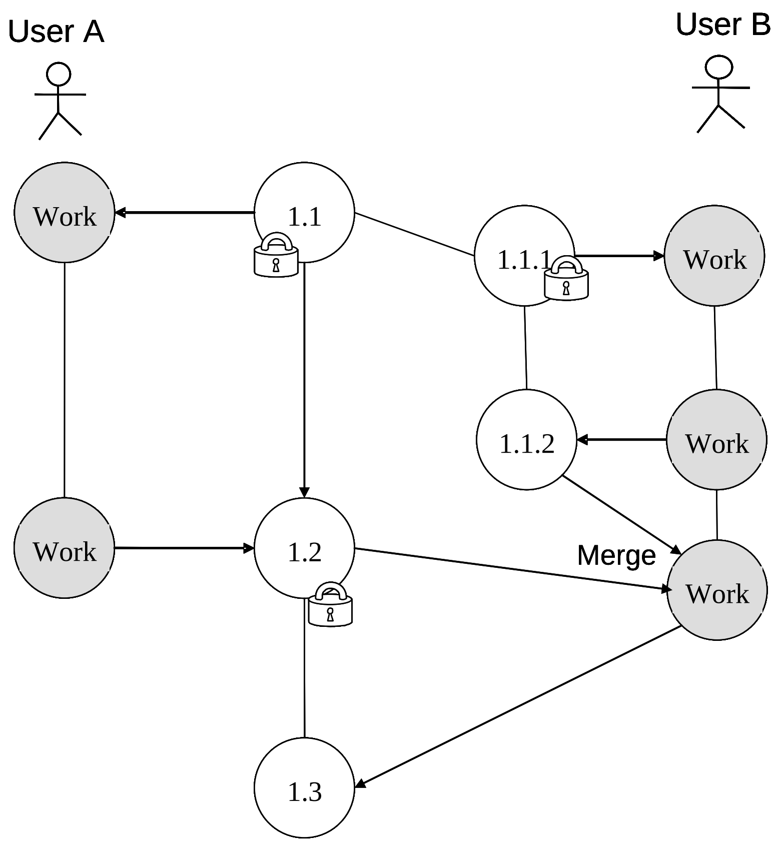

Figure 9. Concurrent development in SCM, a conservative update strategy with locking

The following list summarizes the similarities of version management in PDM and SCM [1, 2, 16] and the differences between them. The concepts in figure 3 and 6 are shown in italic type in the list.

- PDM manages *objects*. SCM manages *files* and *directories*;
- PDM uses *revisions* for major changes. SCM uses *versions* for all changes;
- SCM has *branches* and supports *merge* functionality. PDM does not;
- In SCM concurrent development on file level is supported. In PDM it is not;

- Both PDM and SCM tools have *attributes*. PDM support customized attributes. SCM has a special attribute called *label*, which is frequently used. General, user-defined, attributes are rarely used due to restricted visibility;
- PDM has *relationships*. SCM does not except the revision-of relationship implementing historical versioning;
- A relationship in PDM may have attributes in PDM but not in SCM.

2.2.2 Product Structure Management

A product structure most often forms a hierarchical structure. The product structure comprises components, the externally visible properties of those components, and the relationships between them.

In PDM product structure management is a basic and fundamental functionality. The PDM tool describes a configuration by arranging the parts in a structure consisting of different products or parts connected by relationships. Figure 10 shows a product structure where a specific product revision (in black) is related to several specific parts/sub-products (all showed in black).

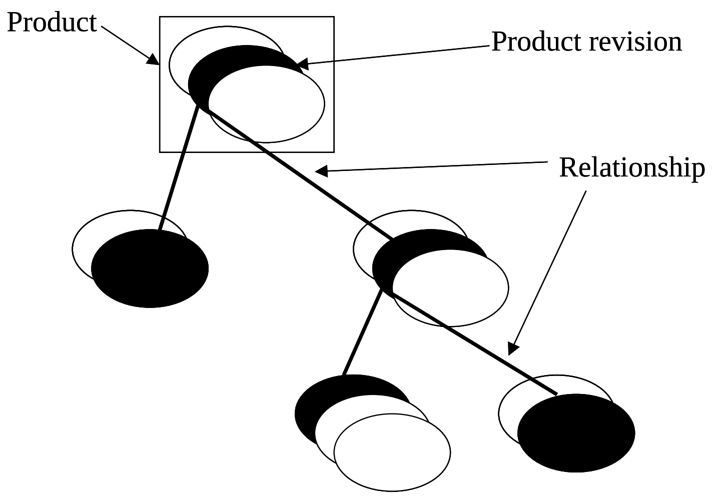

Figure 10. A configuration in a product structure in PDM

In PDM tools a specific product structure, Bill-of-material (BOM), is used to describe the objects and information the final product is built of. This structure is built by using a specific relationship. The manufacturing uses the BOM to collect the included parts when assembling the product [1, 2, 16].

A business item in PDM can represent any kind of object describing the product. Various kinds of relationships can be used to connect the business items, e.g. described-by, requirement-for, designed-as, built-as, and planned-as. During the product's life cycle, the product structure is used differently depending on the stakeholders' requirements. E.g. a designer and a manufacturing engineer need to see a product from different perspectives, which result in multiple product structures. The variants of product structures are referred to as views. These views are built by using the various kinds of relationships (designed-as, described-by, built-as etc.).

PDM systems also identify variants of parts. The relationships between parts may contain rules used for the selection of alternative parts. These kinds of rules define what to include in a product. Configuration effectivity is used to define when a part is valid in a product configuration and to select the correct revision of the part.

Software uses a similar approach, as PDM, in object-oriented design and programming. SCM tools, however, do not explicitly address and support product structures. Only rudimentary support for a software structure in form of files and directories in a file system is available for use in building a hierarchical structure. SCM tools provide support for managing these structures. One of the goals for SCM systems is to make the software structure explicit, defining the relationships between components. The backbone of such models is the dependency relationship. These product models form graphs with nodes as components. One difficulty for SCM systems comes from the fact that the dependency structure (a graph) does not replace but coexists with the file system structure (a tree), and they usually do not match. The management of both structures is not easy. This is why many SCM systems simply ignore the dependency structure.

Since PDM and SCM have different focus and support on product structuring, and different demands on their use replacing a PDM system with SCM only would be impossible. Replacing SCM with a PDM system would gain benefits for the developers, product managers, configuration managers, system engineers and other stakeholders in form of describing the system to-be delivered by help of a product structure.

2.2.3 Build Management

Build management in SCM supports the user in automatically building the software product and includes two central types of transformations. Source code is transformed to binary or executable form [14], and the product structure itself is changed. Typically, a new directory structure, which includes the newly created executable files, is created. Compilers perform the transformation of source code to executable code. Transformation of the structure is part of the build management and supported by various Make tools [21]. Make is a tool, which controls the generation of executables and other non-source files of a program from the program's source file. The Make tool gets its knowledge of how to build the program from the makefile, which lists each of the non-source files, how to compute it from other files, detects automatically which files needs to be updated based on source files which have changed, and the proper order of updating files. The Make tool is not limited to any particular language.

In PDM a specific business item is created, a configuration, which is a set of product revisions ordered in the product structure, see figure 10. The configuration has a revision and once a configuration is frozen, included products cannot be deleted from the PDM system. Build management is essential in SCM, but is in no way supported in PDM for software.

2.2.4 Change Management

The basic principles of change management PDM and SCM are similar.

In PDM add-on modules support change management. For hardware products the changes are either performed outside the computer system (if a physical change) or with tools such as CAD/CAM in which the drawings are managed. Figure 11, [22], shows an example of a process for change approval. In the work-in-progress vault (WIP) all documents which work is in progress are stored. When a review is to be performed, a work order is sent to the designer. The designer sends the document to designated users for reviewing. The change review board will manage the comments. When the document is approved, it will be stored in the release vault.

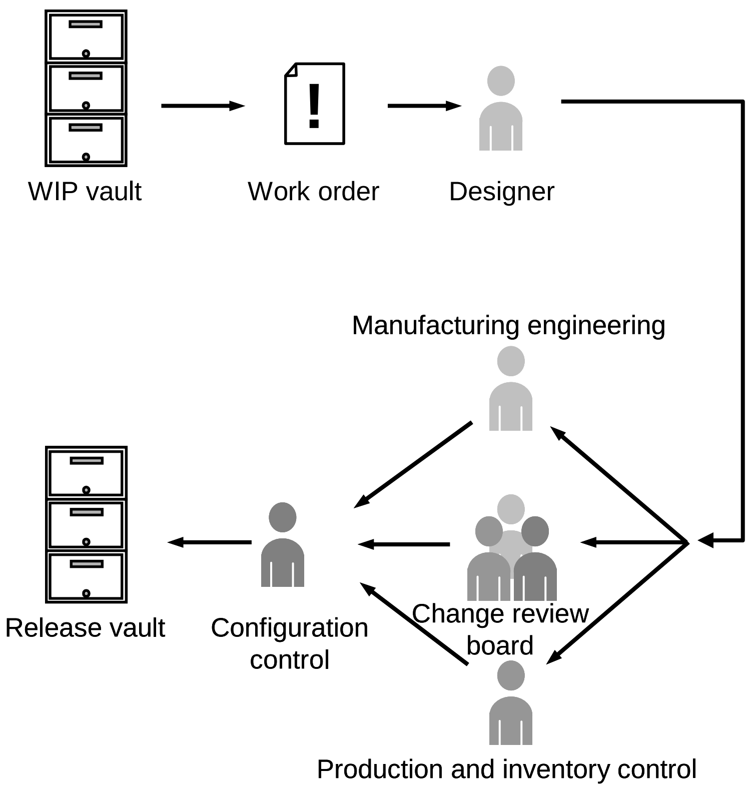

Figure 11. Example of a process for change approval in PDM

In SCM specific change management tools are integrated with the SCM system, e.g. ClearQuest® [23], and PVCS [24]. For software products, which is stored in a computer, a tight integration between the change process itself and change management is easier to achieve. For example, files can be accessed directly from the change management tool to be modified, or a new product version can be built automatically from a particular product version with added changes. This support is often available in SCM tools. Figure 12, [25], shows an example of a change process, which describes the steps a change is performing. Any team member of the project or other initiated stakeholders can submit a change proposal. The change proposal must be documented before the change control board (CCB) is deciding if the proposal should be approved or not. An approved change request is forwarded to the developer for implementation and testing. If a change proposal is rejected, the change will not be implemented, the proposer of the change is informed about the decision, and the proposed change is filed. The change control board decides which changes to be implemented in what release of the product.

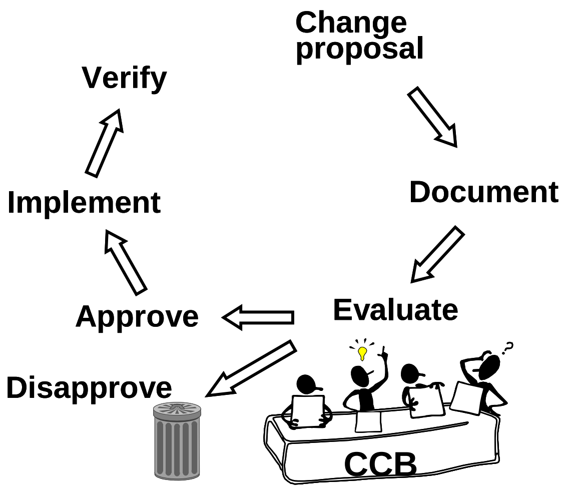

Figure 12. Example of a process for change approval in SCM

2.2.5 Release Management

The identification and organization of all deliverables incorporated in a product release is designated release management. Release management has a double role; (i) to prepare deliverables and all documentation for the users, and (ii) to provide information used internally for test purposes, maintenance or further development.

In PDM, the support for release management is strong. The product structure is sent to the manufacturing resource planning system and the BOM is sent to the production team to assemble the product. Configuration effectivity is used to define when a part is valid in a product configuration, and will inform the production team if a certain part should be used or not. The package sent to the customer is a component in the product structure with relationships to the constituent artifacts.

In SCM the support for release management of software products is simple. It is possible to create installation kits automatically to ease the task of the build manager. The build manager is responsible for providing the packed product with correct configuration and features. Products such as Windows installer [26] and Install shield [27] can be used to create installation kits.

2.2.6 Workflow and Process Management

Workflow management is a critical part in the product definition life cycle to ensure that the right information is available to the correct users at a proper time. It includes defining the steps in the process, the rules and activities associated with the steps, the rules for approval of each step, and the assignment of users to provide approval support. Workflows in PDM systems provide the mechanism for modeling and managing defined processes automatically. Data can be submitted to the appropriate workflow for processing. Appropriate information is routed automatically.

Some SCM tools incorporate similar functionality or provide it using tools tightly integrated. However, in most SCM tools the support consists of triggers only, which can execute scripts written by the users.

From a system level perspective, the process support is essential. Processes as change management, baseline management, and document approval are examples on processes useful for not only PDM and system level, but for SCM too. In principle the support provided either by an SCM tool or a PDM tool can be used in both domains. The problem that should be solved is the integration of the tools, which are supposed to be triggered by events from the workflow management tool.

2.2.7 Document Management

According to [28]: *"Document management is functionality for managing documents that allows users to store, retrieve, and share them with security and version control."* A document management system consists of several functions to support the document life cycle such as (i) document creation and import of documents, (ii) data storage, (iii) document editing, (iv) publishing, (v) viewing, (vi) archiving (long-term storage), and (vii) document disposal.

In many PDM tools, document management is its integral part. Several PDM vendors have integrated document management in their tools. A common PDM system has several built-in document management functions. Like document management systems [2], PDM systems use a relational database to store metadata about the document, provide similar version management, and workflow management. Further, similar to document management systems, PDM systems support distributed development by managing distributed databases using different replication mechanisms for updating the local database.

Documentation is an important part of software development. In many of the SCM procedures, documents are managed as any other item, i.e. under version control, change management, and release management. However, there are parts in document management not present in SCM tools. In SCM tools, there is no advanced search capability – comparison and merge functions usually not work for documents because of their internal format-, and web management is not part of SCM. Support for importing documents is not available in SCM. However, since developers prefer to work in an integrated environment, there is a trend to store documents in SCM, despite its lack of document management functions.

2.2.8 Concurrent Development

Both PDM and SCM provide shared databases and locking functions to prevent simultaneous updates. When a user checks out a data item (the actual file/object) in PDM, this version is locked to prevent other users from checking out the same version. Thus, there is no possibility of creating several versions of a data item to exist in parallel. When the data item is checked in again, the new version is stored and the lock is released. Several business items (only representing metadata) can be checked out simultaneously, but no user awareness or synchronization is supported. PDM does not support concurrent development on a single file.

Most SCM tools enable teams to work concurrently on a single object by supporting a specific synchronization model [20, 29]. Depending on selected model for synchronization of concurrent engineering (check out and check in, long transactions, and change sets), the usage is different. For example, if a user uses an SCM tool supporting the check out and check in synchronization model, concurrent development is supported by branch and merge. If the user needs to check out an item, which has been checked out by another user, a temporary branch is created where the item can be updated in parallel and then merged when concurrent work is no longer required. When several developers are working concurrently in their private workspaces, control is needed between the different copies of the same item. The workspace management provides this support.

2.2.9 Configuration Management and Selection Management

Configuration management manages both hardware and software and originally focused on manager support. From a management perspective, configuration management directs and controls the development of a product by the identification of the product components and the control of their successive changes. The objective is to document the composition and status of a defined product and its components, ensure the correct working basis is being used, and the product is composed correctly. Examples of standards supporting this discipline are ISO 10 007 [25], and

ANSI/EIA-649 [30]. Configuration management is both management discipline and a process.

CM from a PDM point of view provides the tools needed to more effectively communicate with dispersed workgroups and business partners that comprise to:

- Communicate and control engineering changes and determine which changes have been implemented;
- Plan and control product configurations supported by the product structure;
- Synchronize collaborative product development at geographical dispersed sites, and provide awareness of product progress;
- Synchronize multisource procurement and multisite manufacturing through centrally controlled and distributed BOM and related specifications to yield a product consistent with a single set of specifications;
- Configuration effectivity to meet different stakeholders need particularly used for manufacturing purposes.

Parts of configuration management, as described in standards [24, 30], are implemented in many PDM tools.

In PDM, selection is understood as dynamic filtering of information, similar to views in databases. This is named *configuration context* of the product. Views are hierarchically structured and built from relationships such as as-designed. PDM tools implement this concept of views by a label on composition relationships indicating in which views this decomposition is to be visible.

SCM is closely related to configuration management [24, 30], but more focused on specific software support such as change management and version management.

A software system consists of a large number of files and each file can include a number of versions. The possible numbers of combinations of files is enormous. Different stakeholders need different versions of items, e.g. developers need the latest versions, and the test team needs the tested versions. Several SCM tools support a rule-based selection mechanism, where rules such as the latest version in my own branch and the released version will be selected. This rule selects selected baselines.

2.2.10 Workspace Management

As described in [14, 31, 32] a workspace is a working environment or context providing safety from other developer's work on the same or different version of files. All work performed in a workspace is under the SCM control.

In an SCM tool the user checks out all the files to be changed. The files are stored in the user's workspace. The SCM system registers all files checked out, the version checked out, by whom, and in which workspace. If several users check out the same file, the tool in accordance with used synchronization model coordinates the checkouts. Each user can set up and change the selection of file versions that are to be checked out to the workspace.

PDM systems have work locations, with one location per user. In PDM the user checks out one file at a time and updates it. Locking prevent other users from checking out the same file. The file will be saved in the private work location, when it is checked out. The user has no authority to change the location.

2.3 Conclusion

The results of the discussion in this section are summarized in table 1 and table 2. In table 1 PDM and SCM are compared, with respect to availability of different functionalities. In table 2 we are summarizing the pros and cons of functionality needed for supporting complex product development in PDM and SCM. The pros are marked with grey.

Type of Functionality	*PDM*	*SCM*
Version Management	Yes, simple sequential versioning	Yes, with branch and merge
Product Structure Management	Yes	No
Build Management	No	Yes
Change Management	Yes, but not well integrated with other functions	Yes, well integrated with other functions
Release Management	Yes	Yes, but weak
Workflow and Process management	Yes	Yes, but weak
Document Management	Yes	Partly
Concurrent Development	No	Yes
Configuration/ selection Management	Yes	Yes
Workspace Management	No	Yes

Table 1 Summary of functionality of PDM and SCM.

PDM tools are strong in product modeling.	SCM tools are weak in product modeling.
PDM tools have a long tradition and standardized product evolution control know-how.	SCM do not have a long tradition in product development.
PDM tools are strong in workflow and process management.	Many SCM tools have a good support in workflow and process management
PDM tools are strong in document management.	SCM tools are weak in document management.
PDM is strong in data representation where metadata and data are separated.	In general, SCM tools are weak in management of metadata.
PDM is strong in the data modeling where an object-oriented data model is used**.**	There is no data modeling in SCM. SCM tools manage files and directories effectively.
PDM tools are good in release management and provide additional functions for production and selling.	SCM tools are good in software product release management.
PDM has a weak support for concurrent engineering.	SCM tools are strong in concurrent engineering.
PDM does not support workspace management.	SCM tools are strong in workspace management.
PDM tools do not support build management.	SCM tools are strong in build management.
PDM tools support configuration management.	SCM tools are strong in configuration/selection management.
PDM tools have simpler version management model.	SCM tools are strong in version management.

Table 2 Pros and Cons of support in PDM and SCM tools for supporting complex product development

We can conclude, in comparing PDM and SCM, that PDM tools do not have sufficient functionality to support software management, particularly during the development phase. SCM tools do not have the necessary functionality to support the development of a complex product during its entire life cycle. Thus, PDM cannot replace SCM tools and the opposite.

3 Interoperability in Common Processes

Most high-techs products consist of many components and are developed as complex products. A complex system or product consists, per definition, of many parts (called subsystems or components), and to manage this complexity its development is performed by different teams using their specific development processes. In order to manage the complex system, the system is divided into several subsystems, such as hardware and software subsystems. During the development phase, information is generated in the different subsystems, which is used in the subsystem, between subsystems, and on the system level by different stakeholders. The information flow is important to enable support to the different stakeholders. We discuss the importance of the structure of complex products and the information flow in this section.

3.1 Structures of Complex Products

Several development teams are involved in the development of a complex product. The teams use different technologies, different development processes, and different tools during the development and maintenance phases. The result from each team is assembled on the system level to provide a final product ready for production or delivery. Common to all product development activities is the necessity to manage data on the system level, between the teams, and within the teams. Figure 13 shows an example on system and subsystem levels of a complex product, where the system contains of mechanical, software and ASIC components. Product developed from the different subsystems will be integrated and tested on a system level before the product will be manufactured and shipped to the customer. The required support on subsystem levels is different due to various used procedures and technologies, but common at the system level. At the system level, the differences between the subsystems are disregarded and each subsystem is treated similarly irrespective of whether it is a hardware or software component.

On the system level, information about components, information about the contents of the products, customers, vendors, suppliers, baselines, releases, prices, and markets are needed. Different stakeholders have different demands on information on the system level. The project managers must know if the project is following the time schedule. The configuration manager needs to know the current product configuration and its status as well as related documents for the entire system and for all included components in the system. The designer must have access to requirements documents, project specification, and all information related to the product to be developed. The production engineer needs to find all documents related to a product ready for manufacturing. The sales person needs to find information about the product to present to the customer. All this data, created in the different subsystems, must be available on a system level.

During the development of a complex product several tenths of tools are used. Hardware developers use their hardware development tools for designing hardware components and information describing the components, usually managed in a PDM system. Software developers use specific development tools for building software components and related component information managed by an SCM tool. Other stakeholders use their specific tools for support of finding or refining product information. PDM tools are managing metadata and have advanced functions for data retrieval, data classification, and a product structure management. This implies the PDM systems are suitable for managing the system level. For hardware development, PDM includes or is well integrated with many hardware development tools. In this way, PDM supports procedures for hardware development on subsystem levels. However, there is inadequate support for software development environments. Consequently, a PDM tool cannot be used for the software development part when developing complex products as can be seen in figure 13.

SCM tools are not integrated with hardware development tools and do not support product configurations containing hardware components, or hardware and software components. As a product becomes complex, many activities at the system level are not strictly related to pure software domain, and the efficiency of SCM support is much less than at the subsystem level.

From all this we can conclude that a) different teams/stakeholders must have means by which support their activities in a most efficient way and b) information must be integrated in that sense that it is accessible for all stakeholders.

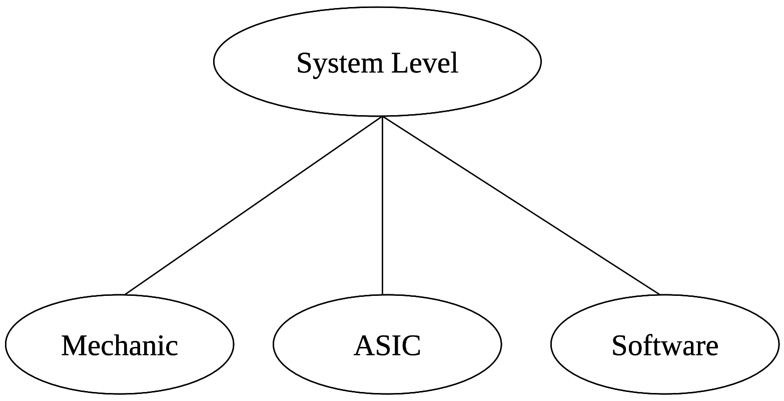

Figure 13. Example of a complex product with hardware and software components

3.2 Complex Product Lifecycle Management

The entire product life cycle management process includes a number of activities, which can be divided into three parts (see figure 14):

- A *common* part in which activities related to the system are performed and information required later in all subprocesses is obtained;
- An *independent* part in which activities related to obtaining solutions of particular product parts (hardware and software components). The subprocesses are progressed in parallel. The information in each subsystem is generated independently of other subprocesses;
- An *integrated* part performed during the integration process in which all processes must be accessible and integrated in common information.

Typically, a process (see figure 14) starts with an overall activity; collection, identification and specification of requirements. The functionality is identified; functions are identified, defined, and documented. The system architecture is defined and documented. The requirements and functions are allocated to the different subsystems. Then the independent activities start, where all these system-related information must be easily accessible for all stakeholders and possible to import the information into the tools used in activities in the subsystems. Figure 14 shows three different independent subprocesses; the sequential hardware development process, commonly known as the waterfall model [33, 34], the system life cycle process, and the unified process [35]. The parallel hardware and software activities need requirements and the system architecture from the common activities, where the requirements are refined. Functions are detailed described, and development of software or hardware components starts. The hardware components are described in their documents. Software components are developed and documented. During the development phase, information such as Change Requests, the product structure, and information related to the project manager and other stakeholders are needed. The integration phase need information from the independent phase such as the refined requirements, final detailed design, and final deliverables (executable code for software, prototype specifications for hardware, and documentation for both). During the integrated part of the process, the system integrates, verifies, and tests according to fulfillment of the defined requirements. Then the system is released.

From this we can conclude that hardware and software development processes should follow common procedures for product structure management, requirement management, and document mangement.

Today, there are problems in integrating processes since in hardware and software development different information models and different information flows are used. Further, they use different structures, have different naming conventions, and have different production process (hardware components are manufactured by a plant and software components are usually built by the developers).

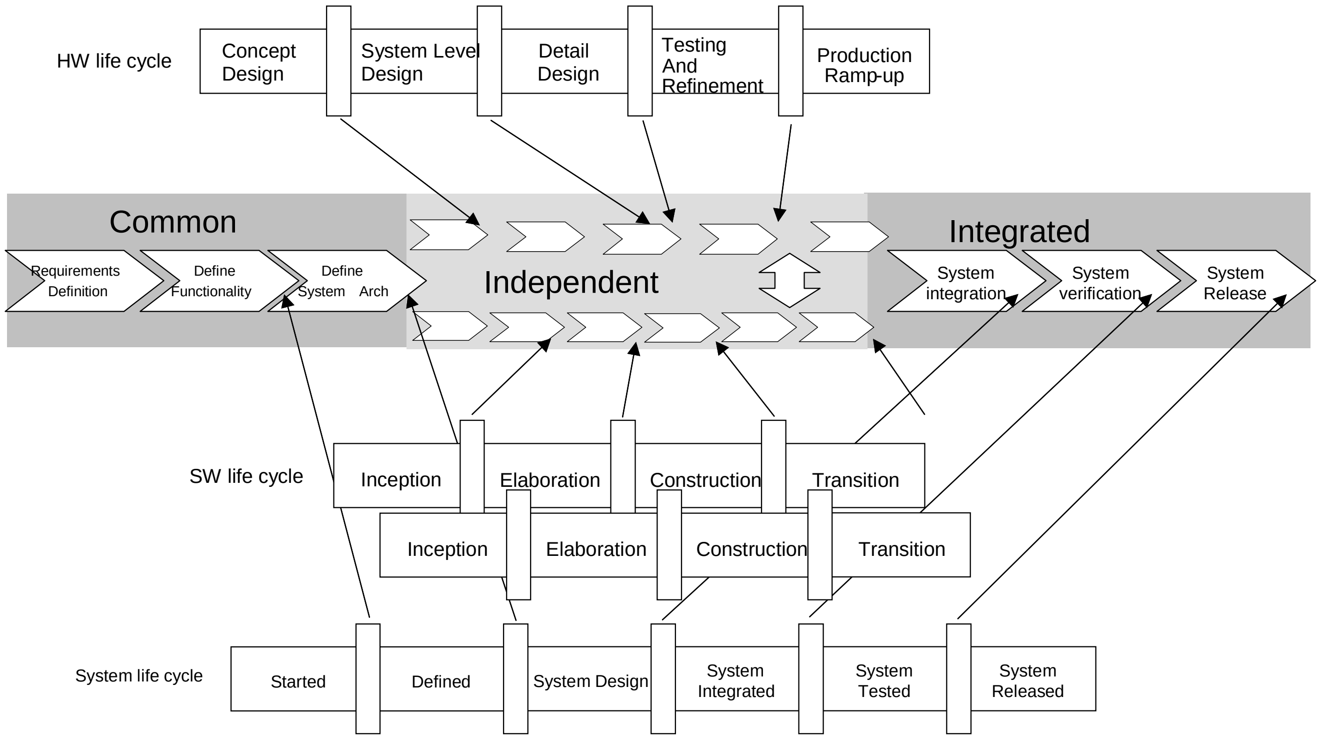

Figure 14. Complex Product Lifecycle

3.3 Conclusion

All development groups need support for their daily routines. This implies that these groups will use tools that are adjusted for their activities, and this implies that for the entire support different tools with different purposes are needed. Since there is a need for information change between the stakeholders, and consequently between the tools, a seamless interoperability between the tools is required. PDM and SCM tools do not provide the integrated support for the entire life cycle of the products separately. For an efficient development, an overall integration of information is needed during the entire development process at all structure levels. To achieve seamless information flow on system and subsystem levels, integration of PDM and SCM tools is needed.

4 People and Culture

In most Western language ‘culture’ commonly means civilization, but social anthropologists use the term in a broader sense and see the culture as mental programs of human beings. All humans carry within him or her patterns of thinking, feeling and potential acting [36], called mental programs. The sources of one’s mental programs lie within the social environments, e.g. family, neighbors, the nation we are living in, religion, gender, education, and profession. There are several layers of cultures, and one of them is the social class level, associated with education opportunities and with a person’s occupation or profession. Uncertainty in a culture level can create anxiety. Every society has developed ways to alleviate this anxiety. These ways belongs to the domains of technology, law, and religion. The essence of uncertainty is that it is a subjective experience, a feeling. Feelings of uncertainties are not only personal, but may be shared with other members in one’s society. Those feelings of uncertainties are acquired and learned. The feelings and the ways of coping with them belong to the cultural heritage of the society and are transferred to others in the same society leading to collective patterns of behavior. According to [34], hardware is less uncertain than software. In the software domain there are more informal rules controlling the right and duties of the developers, and less laws and rules to prevent uncertainties in the behavior of other people compared to hardware the domain. For the hardware domain more matured technology helps to avoid uncertainties combined with formal rules controlling the developers.

The social cultural, i.e. the cultural social class level, differences between hardware and software development groups play a much more important role when building integration between PDM and SCM. The general rule is that when people are moved as individuals, they will adapt to the culture of their new environment; when people are moved as groups, they will bring their group culture along. People in groups in respective domains have developed, as part of their culture, ways of interacting, which are quite stable and difficult to change. In interoperability achievements and for decreasing culture differences, people from the two domains should be organized together in smaller groups.

In addition to findings in [36], we have observed five more differences between hardware and software domains:

- Both domains are huge using completely different tools developed for the specific domain with their requirements.
- Users from the different domains do not have knowledge about the other domain. Low communication between the domains causes poor understanding of each other's problems and requirements.
- Users from both domains believe that the system they use can manage all situations from the other domain [2, 15].
- PDM and SCM users are often located at different departments within the company. Their geographical separation can increase the gap in their understanding of the other group. In managing and changing the organizational culture, two parties are crucial for culture innovations, one power holder preferable a person with charisma, and an expert.
- The hardware designer uses a lot of documents to describe the product. These documents are transferred to the production and manufacturing part used of another person to produce the actual product. Hence, the hardware designer focuses on documents. The software designer writes a lot of source code. The designer then generates the actual product, the load modules, with no other person involved. Hence, the software designers focus on source code more than documents and have small understanding of the importance of writing documents.

We can conclude that it is not enough to integrate the tools; people must be "integrated" too. This people integration belongs to organization issues, and can be achieved by building common training, workshops, moving the developers together (organizationally), and exchange people between the groups.

5 Different scenarios in an integrated environment

An important requirement of an integrated environment is uniformity of user interface, irrespective of where data is stored. A PDM user should be able to independently access a document stored in either the PDM system or an SCM system. Similarly, an SCM user should not perform any additional action when a document stored in the SCM system is related to metadata in the PDM system.

To illustrate type of interactions and information exchange between PDM and SCM tools, we discuss a hypothetical integration between a PDM and an SCM tool. First, we define prerequisites for the integration and decide where to store data. Then, we describe two scenarios, one case for a PDM user and one case for an SCM user.

The integrated system has the following characteristics:

- The PDM system will manage metadata of the delivered products, including its components (e.g. library and executables), and certain metadata for software documentation;
- The PDM system will manage the product structure, the revisions of included products, and related documents;
- Documents created in PDM, will be managed in PDM and links to the SCM system will be provided on demand;
- The SCM system is the archive for all software information i.e. source code and binary files, and the software parts included in the BOM;
- Document related to software are stored in SCM, and related metadata is stored in the PDM system;

- Items stored in the SCM system with related metadata stored in the PDM system, are marked with specific attributes for synchronization purposes;
- When an item is stored in the SCM system with related metadata stored in the PDM is changed, SCM automatically sends information to PDM.

5.1 Scenario: PDM – User interaction

In this section we analyze scenarios related to users of the PDM system and files stored in the SCM system. All functions required by a user are initiated in the PDM system. The use cases show the kind of information the PDM user requires from SCM and how the two systems exchange the information. We have identified following use cases:

- *Query for a document*. The user states the document identity (or other search criteria) in the PDM system and the system (i) search for the document in the PDM system, if not found in the PDM system, a search is performed in the SCM system and (ii) a list is presented for the user of all found documents that match the search criteria.
- *Get a document*. The PDM system (i) look up the actual file in SCM by using the path to it, (ii) copy it, and present the document for the user.
- *Check out a document*. The PDM (i) uses the stored path and search for the document in SCM, (ii) checks out the document in SCM, (iii) copy it into the work-in-process vault in PDM, and (iv) set attributes required too synchronize the states of PDM and SCM (e.g. user identity and status). The user may now update the document.
- *Check in a document*. The PDM system (i) check in the file in SCM, (ii) set all attributes required synchronizing states of PDM and SCM, (iii) and deletes the file in PDM.

- *Delete a document*. The PDM system (i) checks if the user has the access rights and is allowed to delete the specified document in SCM, and (ii) check if the document is included in a frozen product or not. If the document is not included in a frozen product, (iii) search for the document in SCM, (iv) delete it in SCM, (v) delete attributes in SCM, and (vi) delete all metadata and relationships in PDM.
- *Import a document into PDM*. The path to a document existing in SCM is created and inserted in PDM. If metadata exist in PDM for the document, PDM (i) issues a query for the document in SCM, and (ii) saves the path to the document in the metadata. If the metadata for the document does not exist in PDM, (i) PDM must create a business item and populate it with metadata, (ii) search for the document in SCM, and (iii) save the path to the document in the metadata for it in PDM.
- *Export a document from PDM*. A PDM user wants to make a link to the document in PDM from SCM. PDM (i) checks if the document exists in SCM. If the document does not exist in SCM, PDM checks in the document version, related path to the document in PDM, and related metadata. If the document does exist in SCM, (i) PDM checks if the document version is the same. If the document version is the same, (ii) PDM makes a diff between the versions to check they are identically, and inform the user if any discrepancy. Other wise, if the document version in the PDM system is less than in the SCM system, (i) PDM inform the user that the document in PDM is older than the one in SCM, and no changes are made in SCM. If the document version in PDM is higher than in SCM, the path in SCM and related metadata is updated.

5.2 Scenario: SCM – User Interaction

In this scenario, users only use an SCM system. Information generated in the SCM system must be automatically updated in the PDM system. The use cases show the kind of information the SCM user requires from the PDM and how the two systems should exchange the information. We have identified following use cases:

- *Register a new product*. The SCM user registers a new product in the PDM system. SCM (i) creates a new product item, (ii) specifies all of the necessary attributes of the product (such as owner and parent product), and (iii) receives the product identity, which may be used in SCM for a different purpose such as creating a baseline, or a new working structure.
- *Register a product revision*. The SCM system must know the identity of the next revision of the product. (i) SCM sends an inquiry to PDM for the next product revision, (ii) PDM allocates the product revision, (iii) PDM sets the appropriate metadata, and (iv) PDM delivers the new product revision identity to the SCM system.
- *Register a new document*. One or several files stored in the SCM system are registered as new items in the PDM system. SCM must provide PDM with (i) the full path to the files and (ii) attributes with relevant metadata such as document status and document revision. Further, SCM must provide PDM with (iii) the identity of the product to which the information belongs. When registering a library or an executable in PDM, (iv) the reference to all included software source code files are also registered.
- *Export a document from SCM*. The path to an existing document in SCM is provided PDM. If the document does not exist, it must be registered first, and then the path is sent to the PDM system along with certain attributes.
- *Check out*. The file is already registered in PDM and may be frozen. SCM (i) checks if the file has already been checked out. If the file is not checked out, (ii) SCM checks out the file in SCM, (iii) and if metadata about the file is stored in PDM, all attribute for synchronization purposes is set. The user may now update the file.

- *Check in*. SCM will (i) perform a check in of the file, and (ii) set all attributes required to synchronize the states of PDM and SCM.
- *Uncheck out*. SCM unlock the file lock and changes the status in PDM.
- *Update product status*. SCM sets appropriate attributes in PDM, depending on the company-specific development process.
- *Delete document*. An SCM user wants to delete a document. (i) SCM checks if the user has the access right to delete the document. If the user has, (ii) SCM uses the document number label and searches for the document in PDM. (iii) If the document is not included in a frozen product, SCM will delete the document in PDM and (iv) in the SCM including all labels and attributes. (v) If the user does not have access rights for deleting the document, an error message will be returned to the user and the document is not deleted. If the document is used in a frozen product, (vi) an error message is returned to the user and no document is deleted.
- *Query regarding product revision*. SCM will use the product number label and query for current product revision in PDM. The result will be presented for the user.
- *Query regarding a product structure*. SCM will use the product number label and request PDM to retrieve the full product structure in which the actual product is included. The product structure will be presented for the user in SCM.
- *Query regarding documents*. An SCM user searches for documents placed in PDM only. SCM will (i) use a search key and perform a query in PDM to retrieve the actual document. The result (ii) will be presented for the user.
- *Import documents into SCM*. An SCM user wants to create paths to documents stored in PDM for reading purposes. SCM (i) checks if the documents exist in SCM. If they do not exist already in SCM, SCM (ii) issues a query for the

documents to the PDM system, (iii) create appropriate paths to the documents, and (iv) sets the appropriate attributes in PDM and SCM.

5.3 Conclusion

We have set up integration prerequisites for a hypothetical integration between PDM and SCM. This resulted in two scenarios of user interaction, one for PDM and one for SCM users, including several use cases. The example shows requirements on intensive communication and interchange of information with difficulties in automatic synchronization and update of data. This can lead to a conclusion that either a tight integration of the tools should be achieved, or the processes should be isolated but at given and well defined points import/export of data should be achieved. The first case would lead to a technically better solution, but it is a question if it can be achieved – since there are too many vendors of PDM and SCM tools, too many different domains which these tools cover, to be feasible to define a common information model. The second approach is more feasible from today's perspective, although it provides significantly lower quality of services.

6 Case Studies

We have performed seven case studies from international companies based in United States, United Kingdom, Switzerland, and Sweden. The case studies serve as practical examples on how PDM and SCM are used in the companies. The products manufactured by the companies described range from pure software to mixed products containing both hardware and software. The case studies cover issues related to the development processes, product lifecycle management and to the tool set used to support these processes. The information was acquired through interviews and discussions with one or many persons in the companies who had responsibility for or experience using their PDM or SCM solution. The interviews were based on questions, which we provided the interviewees before the meeting. Each case study focuses on certain important issues from the company, ranging from technical descriptions of tool usage to descriptions of the development process.

The cases are detailed described in [2]. In this section, we select only two cases - one extracted from [2] and one case made after [2] and after the hypothesis about the three factors has been stated. This last case study is used for hypothesis validation.

Following is a short outline of all the cases we have performed:

- *Sun Microsystems, Inc*. The cases study targets the Sun's product lifecycle process, which is one of its core business processes. The four key process elements, structured process, product approval committees, product teams, and phase completion reviews, are discussed. Further, the tools supporting the product lifecycle and their deployment have been described.
- *Mentor Graphics Corporation*. The description of the case is focused on the development process and the product lifecycle at one division at Mentor Graphics Corporation. The requirements management, development process, and change management are discussed. Further, the tools supporting the development process have been described.
- *Ericsson Radio Systems AB*. Operational PDM/SCM concept from a concrete project is described. The processes and information flow, and tools and technology are discussed.
- *Ericsson Mobile Communication AB*. The study discusses the usage of the PDM tool acceptance among developers. Further, the product modeling and traceability are described.
- *ABB Automation Technology Products*. The case discusses the management of hardware and firmware; how the product structure is designed to efficiently manage different product variants realized in different ways, depending on the manufacturing volume.
- *SaabTech Electronics AB*. A case is described in which there exists a need to replace the current PDM system. A specification of the new planned process and the architecture of the new PDM system are discussed.

- *Industrial and Financial Systems*. In this case we discuss company developing software only. The company has realized problems in managing their delivered software in the SCM system only, and introduced some of their own products' modules to manage customer and product information. The case looks into the product lifecycle and the information flow.

6.1 Case study: Ericsson Radio Systems AB

Ericsson Radio Systems AB is a subsidiary within Ericsson AB. Ericsson AB is a supplier of a complete range of solutions for telecommunication systems and applications to serves and core technology for mobile handsets. With the establishment of SonyEricsson, the company is also a leading supplier of complete mobile multimedia products. Ericsson supplies operators and service providers around the world with end-to-end solutions for all existing mobile systems and third generation mobile systems, in addition to broadband multiservice networks, and broadband access. The solutions include network infrastructure, access equipment and terminals, application enablers, and global services to support both business and private communication.

The section contains a case study from Ericsson AB (formerly Ericsson Radio Systems AB).

The case study describes a project in which a product for the standard personal digital cellular was developed. The project was a large development project performed at three design centers, Sweden, Japan, and Germany. The product has been delivered to the customer. The developed complex product consists of a large number of subproducts, both hardware and software. The main project management group of 32 persons directed the work of hundreds of project members. The product was developed for one specific customer in Japan.

In this case study we describe the processes one specific project use and the information flow. Then we describe some of the used tools managing product data and supporting the development process. Finally, we discuss some tools integration requirements.

6.1.1 Processes and Information Flow

In the company several processes, e.g. project management process, hardware development process, and software development process, have been defined. These common company processes are used in a project when a product is developed. The processes we describe here are common company processes used in the project.

The product life cycle is divided in three subprocesses: “time to market”, “time to customer”, and “maintenance and support”, shown in figure 15. The time to market process spans from customer input to the delivery of the product design. It contains product management, design, and marketing processes. In this specific case, there was one specific customer. Hence, the input to the project was one single contact. The time to customer flow is the manufacturing process from the final design to the delivery to the final product. The maintenance and support flows are parallel with the time to customer flow. These activities start up before the actual product is delivered to prepare and educate the help desk and repair centre before any customer enquiries exist or products need to be repaired. This product lifecycle process is well understood and followed in the project.

After product release, the product is handed over to the local Ericsson Company for first-line support. Second-line support is managed within the design and maintenance organization.

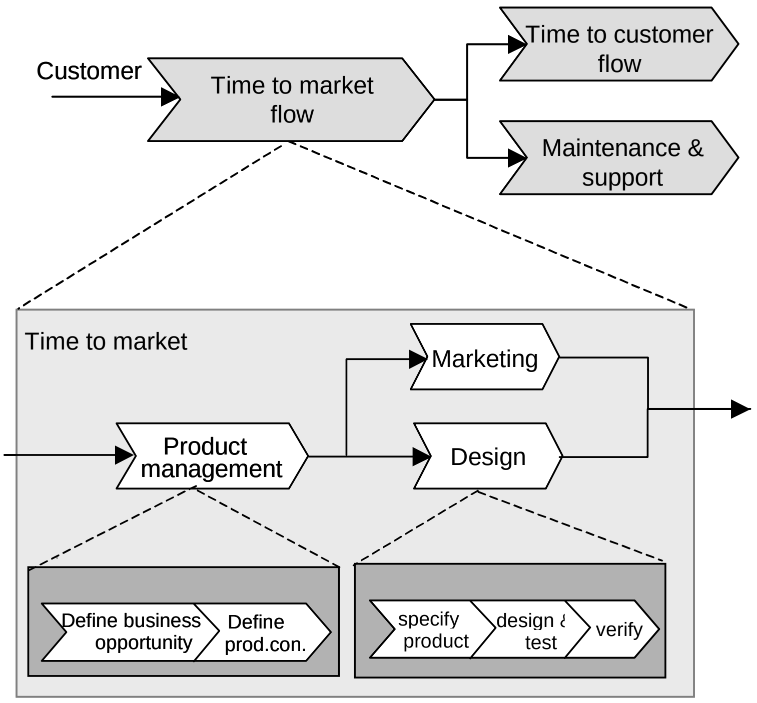

Figure 15. Time to market flow

The CM methods applied in the project were well defined and understood. The project was performed by building and delivering in a set of baselines. The baseline content is shown in figure 16. Many baselines are created during the project. Each baseline contains a report including the documents; minutes of meetings (MoM) from the configuration control board (CCB), trouble reports (TRs), change requests (CRs), audit reports, and test reports. All these documents are registered in the PDM system and stored in the project archive. When the product is ready for release, all documents are stored in the common company archive. Each baseline always documents the four CM corner stones: configuration, audits, deviations, and decision.

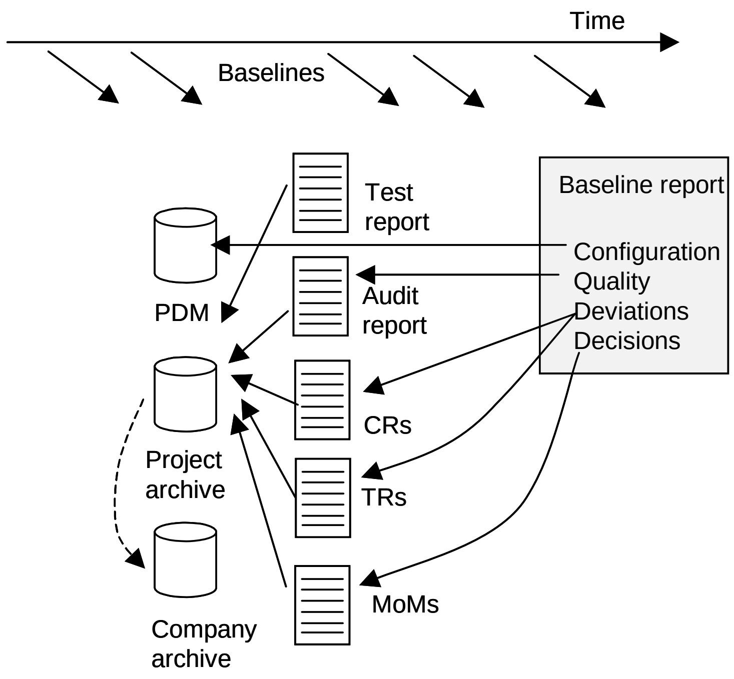

Figure 16. The baseline content

The changes are managed through a change process in form of change requests and their processing. The flow of a change request (CR) process is shown in figure 17. The product manger or a project member of the project team can introduce a CR. This CR will be posted to the mailbox for the configuration control board (CCB). The CR is stored in the project archive, not integrated with the PDM or SCM tools. New CRs will be added into the CR log, which contains all information related to the CR. The CR log is presented and prepared by the local CCB. The main CCB will then make a decision whether to include the CR on the implementation process.

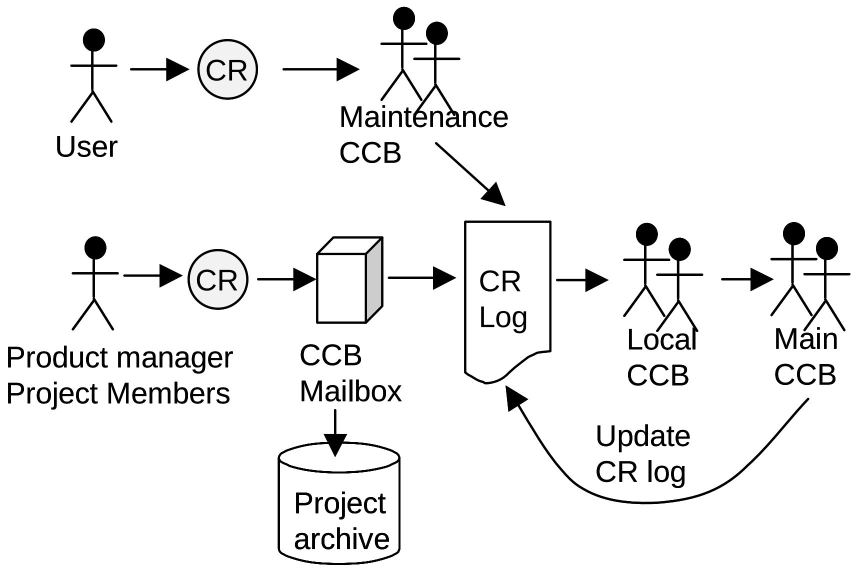

Figure 17. The CR flow

The product information generated in the project is managed in several tools. The different subprojects are managing their generated product data in their specific tools, e.g. hardware product data is managed in their tools, and software product data is managed in software specific tools. In figure 18 the project information flow, involving needed tools are shown. Customer requirements are registered in the customer requirement tool. New requirements generated either by product management or project team members, are stored and managed in the specific requirement tool DOORS® [37]. A main requirement specification (MRS) is written and updated from DOORS® and stored in the project archive. The MRS will be used for design for new or changed functionality. The design documentation is stored in the common company archive. Deliveries are stored in the software archive. PDM contains the product structure and is updated with the latest information. When the design is ready for a function test, it will be fetched from the different archives. Builds will be temporarily stored on local file servers. When a system test is performed successfully, the product is ready for deployment. Customer product information will also be produced. This information is fetched from the common company archive, refined, and stored in the customer product information archive before delivery to the customer.

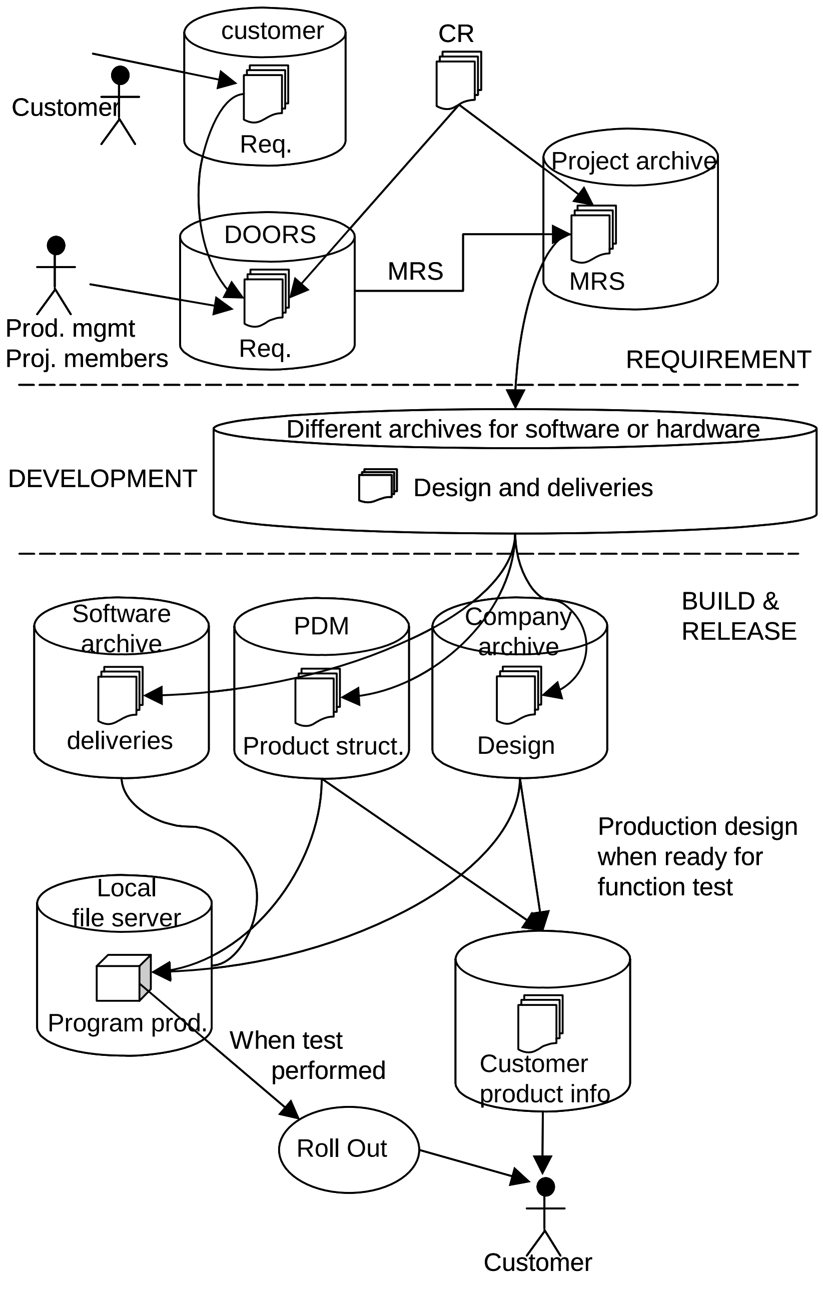

Figure 18. Overview of the project information flow

Information flow for software and hardware is shown in figure 19 and figure 20. The requirement phase for software and hardware is common. The software development phase is separated from the hardware development phase.

The software product is developed on the basis of the input from the requirements specification and product specification process stored in the project archive. During design the software is stored in different archives depending on the design organization and product (e.g. ClearCase® [23] or local archives). When the design is ready the code is archived in an approved archive, i.e. company archive and software archive. All documents written during software development are archived in local archives. All product documents are stored in the company archive.

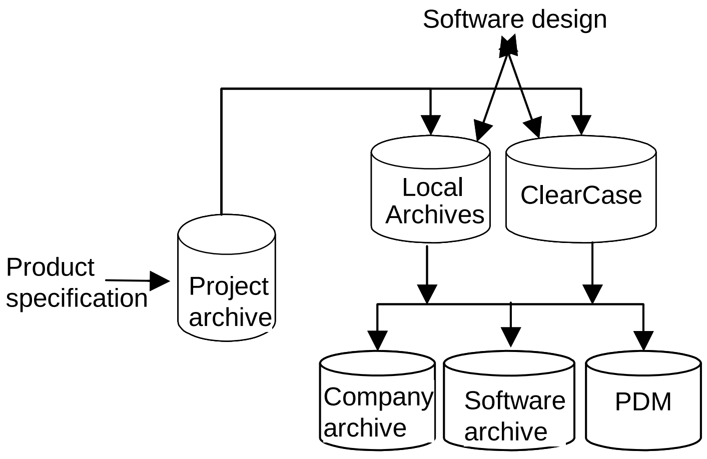

Figure 19. Information flow for software design

Hardware development is much more mature with respect to common methodology and tools in comparison with software. Usually, studies on functional block level identify the need of a new or updated hardware product. The study specifies the requirements and the specification of the hardware units. The prototype is designed in a CAD environment (see figure 20), ending with function tests. Then the realization phase begins. If needed, the printed board is manufactured in a pre-series production for testing properties and function. These measurements are compared with the stipulated requirements and if acceptable, the production unit approves the product as ready for production. Information prepared for time to customer process is exported from PDM and company archive into the order system.

Figure 20 shows the overall information flow for hardware design. Most often the printed board and software are structured into function blocks defining the needed software or hardware modules for a certain function. Mechanical parts, cables, batteries, power supply, cross connectors are structured in a separate structure or separate branches of the functional structure.

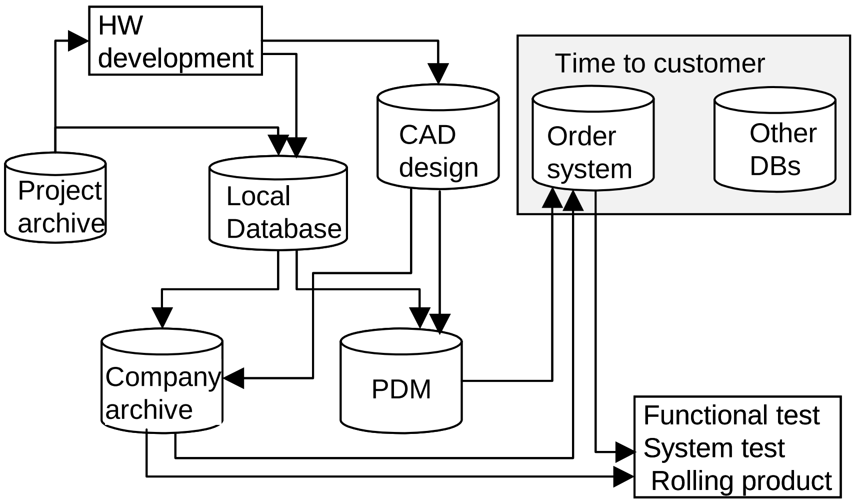

Figure 20. Information flow for hardware design

6.1.2 Tools and Technology

Within the project, hundreds of different tools are used during the products' life cycle. Certain tools are used over the entire company, and other tools are mainly used in a specific project. Some tools are used more often, especially in distributed development environments. Such tools manage product data, support archiving, and requirement management. A few tools are used for the entire company, and are mandatory to use for all projects. Such tools are the in-house built PDM tool and the company archive. We describe some of the mandatory tools and the used tools in the project.

The PDM tool is the central Ericsson product catalogue, in which all released products and related documents have to be registered and managed. Although other systems are used locally elsewhere, only PDM provides global access to the unique number of every product, and is the only comprehensive product register within the

Ericsson group. The PDM tool provides support for managing product data, product structures, document data, and information structure providing information related to a specific product revision. According to Ericsson Corporate Basic Standards, all released products must be managed in PDM.

The in-house built company archive stores documents and software. Ericsson Corporate Basic Standards demands the use of this archive for all documentation of released products. The company archive's security and global accessibility makes it suitable for use in a distributed environment. This archive is tightly integrated with the PDM system.

The project archive is used in many different projects, especially for development of public telephone exchange systems. The archive has an interface to PDM.

The commercial requirement tool DOORS®, manage a large volume of requirements of a high degree of complexity. The tool was used in the project for requirements introduced by project members.

For software management, the commercial SCM tool ClearCase® was used. The multisite functionality was used when geographically dispersed teams developed the product. Once a night all information was synchronized. A common methodology was developed and used to minimize risks and problems regarding naming, branching, and merging.

6.1.3 Integration Requirements

In the project was a good understanding of the product life cycle process. Configuration methods were following industrial standards such as ISO 10 0007 [25]. Many tools were used for managing product data. Although good knowledge and supporting tools, the project conclude there are needs for integration; process, tools, and culture integration. We discuss these integration views.

Process integration

Since the development of software components are separated from the development of hardware components, the integration of the components comes late in the project. Several tests are performed before the final system test. Function tests are the first step of tests. This test can start when the build process is in place. The build process is centralized, and all relevant information is collected at one site where the system is built. This build is then distributed to all subprojects for further test and development. When the development is completed, the software is delivered to a test CM subproject for further test. The deliverables are stored in the central company archive. Preliminary hardware components are manufactured. These components are function tested in the subprojects, where they are developed. When function test is finalized for hardware and software components, system test begins. The hardware and software components are integrated and tested for the system. In order to achieve an efficient system test, the processes should be integrated at certain points such as integration of the processes when software components and the hardware components should be function tested altogether.

Tool integration

Since the company and the project are using hundreds of different tools, many of them were not integrated and no APIs were developed to perform automatic exchange of information. We have found that local tools supporting CRs and TRs were not state of the art and not integrated with each other. This required many manual activities, which may be a source for introducing faults. Either integrating the tools or use a tool could solve this where both TRs and CRs may be managed. Figure 15 indicates the problem of establishing flow control and traceability. With many tools and many interfaces, the possibilities of accessing correct information are hindered by lack of an overall PDM system.

For software development, metadata and files are stored in different archives in different locations. This emphasizes the need for an integrated environment to secure and achieve a seamless information flow for software components.

For hardware development, the information flow is integrated. But when embedded software is needed to fulfill the function on the hardware subproduct, there are humans who are the integrators. There is a need for PDM-SCM integration.

Such as the in-house built PDM tool and the company archive are tightly integrated. If a document is stored in the company archive, the information structure and the document attributes are automatically updated in the PDM system.

The project concluded that too much manual work was required, because of the absence of a modern PDM tool integrated with the different tools providing with product data.

Culture Aspects and Organization

The hardware and software development teams are geographically dispersed. Hardware developers are organized separately from the software developers, which result in low communication between the development groups, and low knowledge bout the other domain. In the project, separate subprojects are managing the development of hardware and software components. In the company, a common terminology is used. This terminology describes the products and its describing documents. Both domains use their specific tools managing the product data. To achieve a more efficient development of the product managed in the project, the people must be integrated too.

6.2 Case study: Industrial and Financial Systems

Industrial and Financial Systems (IFS) provides component-based business software. This software is developed by using open standards. IFS operate in two areas: lifecycle management, and enterprise resource planning (ERP). In addition to product development, IFS provides services to customers. The IFS product, which has the name IFS Applications, uses functional component-based architecture with open interfaces and web services for extended connectivity. Technologies such as Java 2 platform, enterprise edition (J2EE) [38], .Net, and Web Services are used.

The company's headquarter is situated in Sweden. Development is geographically dispersed in different countries worldwide.

In the IFS Applications the software pieces of a business component are separated into several layers. IFS Foundation1 is the core of the IFS Application (see figure 21) implemented as a technology platform. On top of the IFS Foundation1, cross-functional components such as quality management, supply chain management, document management, customer relation management are grouped and used for all add-on functionality. Add-on functionality could be such as financials, e-business, sales and services, engineering used for specifying and configuring design elements and products, manufacturing, distribution, maintenance, and human resources. The add-on functionality consists of several functional components, which may be used according to the customer needs. Interoperation between functional components is achieved through XML-based interchange format, and open standards such as SOAP [39] and OPC [40].

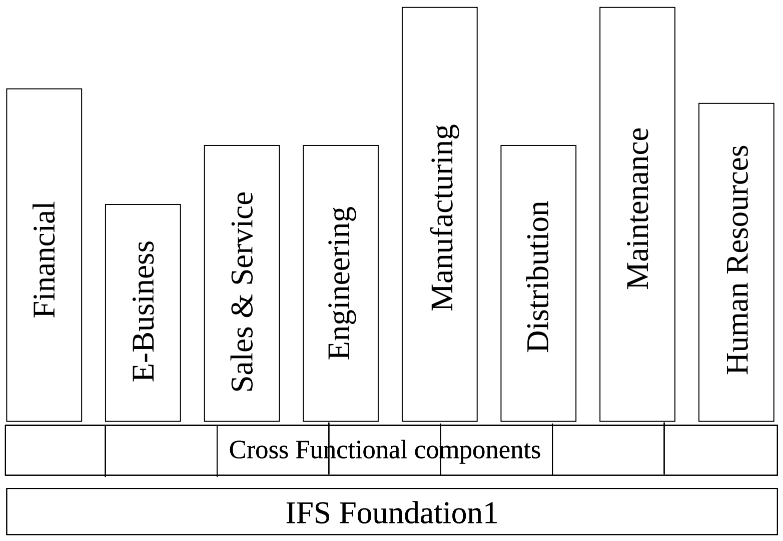

Figure 21. IFS Application functional architecture

IFS staff usually performs installations for the customers. Customers may download patches for installation, but a few do.

6.2.1 Development Process

The company has adopted a waterfall model for their overall product life-cycle process for development of their products and add-on functionality. It consists of six main parts: requirement management, development, release, product support, release marketing & communication, and translation management (see figure 22). Figure 22 shows the development process, and the information processed and saved in different repositories.

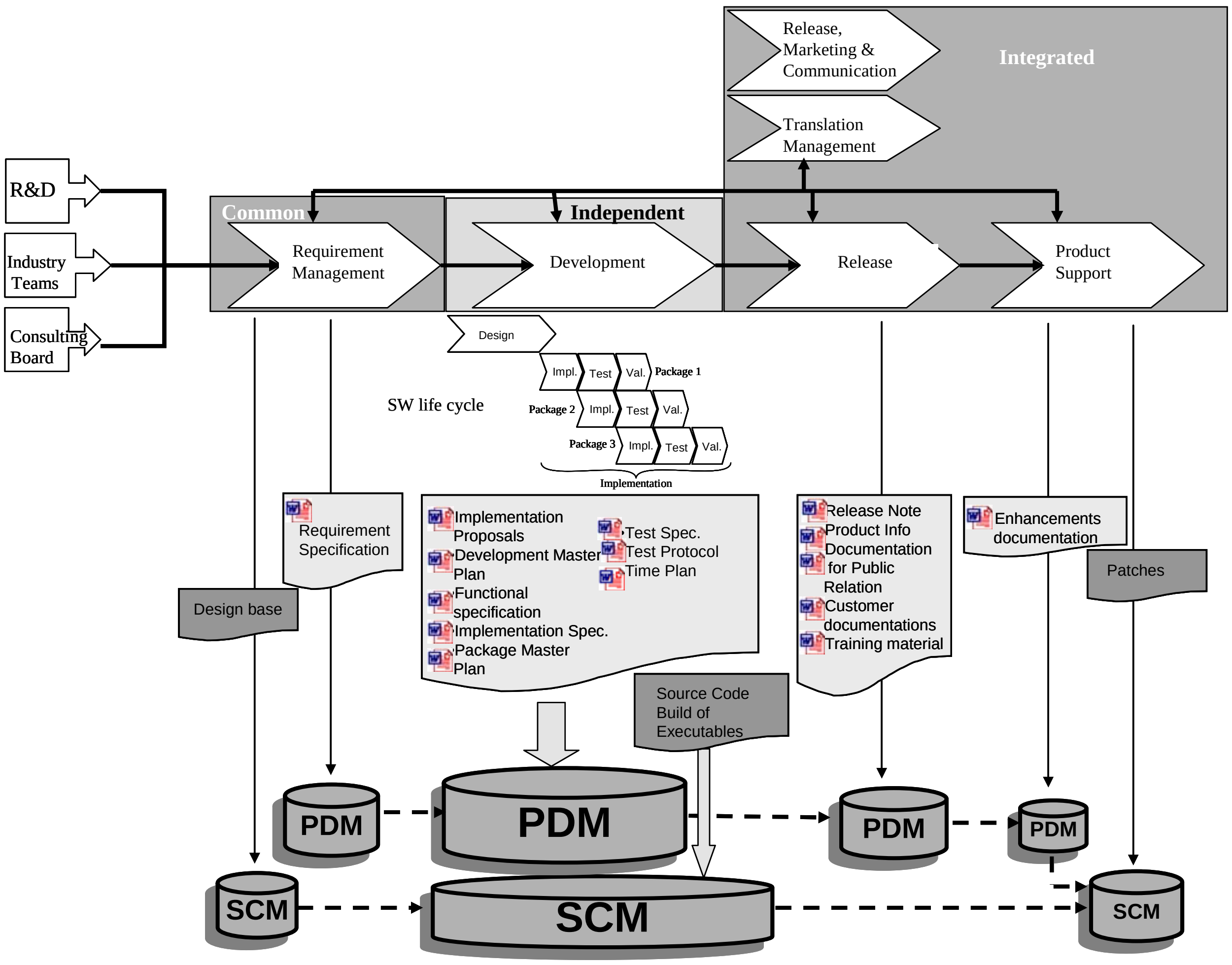

Figure 22. Product Life Cycle Process and connections to PDM and SCM systems

Requests on the product originate from three different sources: (i) industry segment, which is providing functional requirements, (ii) research and development (R&D), which ensures future-proof technology, and the (iii) consulting board providing requirements on installations, upgrades, migrations tools and other implementation tools, and applications management functions. These requests are managed during the requirement management phase in the process (see figure 22). The requests are documented in requirement specifications. The requirement management team for each project writes implementation proposals and a development master plan. A project is set up together with subprojects by appointing a project manager and the subproject managers. The requirement management phase belongs to the common part of the product life cycle. All documents written during the phase are managed in the IFS document management, one part of the PDM system, see figure 23.

The development phase consists of the subphases design and implementation (see figure 22). During the design subphase, time plans are made, dependencies between the components and new functionality are set, functional specifications are refined, and then communicated to and approved by the request sources. The function specification is archived in the PDM system and will be used for writing customer documentation. The requirements are broken down into manageable functions, grouped into packages (a concept reused from the used SCM system characterizing a container including documents or software files), specified in the document packet master plan, and planned in time. The packages are implemented during the implementation subphase. The source code is tagged with an id number, same id-number as defined in the functional specification document, for traceability reasons. Each package is visible and tracked in the SCM system. The company has adopted the Unified Process [35] for one part of the overall product life cycle process, the package implementation process, using the daily built methodology within each package. This incremental development phase, the incremental software life cycle, is depicted in figure 20 in the independent part of the product life cycle. All documents provided during the development phase are managed in the PDM system.

During the release, marketing and communication phase, material for public relations are written and published. During the phase translation the customer documentation is translated into several languages. During the release phase, the new functions provided by the projects are integrated into the IFS Applications and for demo purposes. During product support, customer questions, request and bugs resulting in service packs or patches are managed. All these phases belong to the integrated part of the product life cycle (see figure 20). Documents provided during these phases are managed in the PDM system.

Since the company is developing software and no hardware components, they do not need to integrate hardware processes with their development process. Furthermore, the development process is in-house developed and is built on concepts used in their SCM tool. If the SCM tool has to be replaced by another tool, the development process either has to be adjusted to follow the new tool or a tool-independent process has to be introduced. Still the PDM system is used due to manage customer information and related release information. The PDM system consists of components which the company develops in-house and sells.

6.2.2 Technologies and Tools

The PDM system consists of the Life Cycle Support (LCS) system, application management system (AMS), and the document mangement system IFS Document Management. The IFS Document Mangement system is in-house built and is a commercial product too. In figure 23 the different systems are shown.

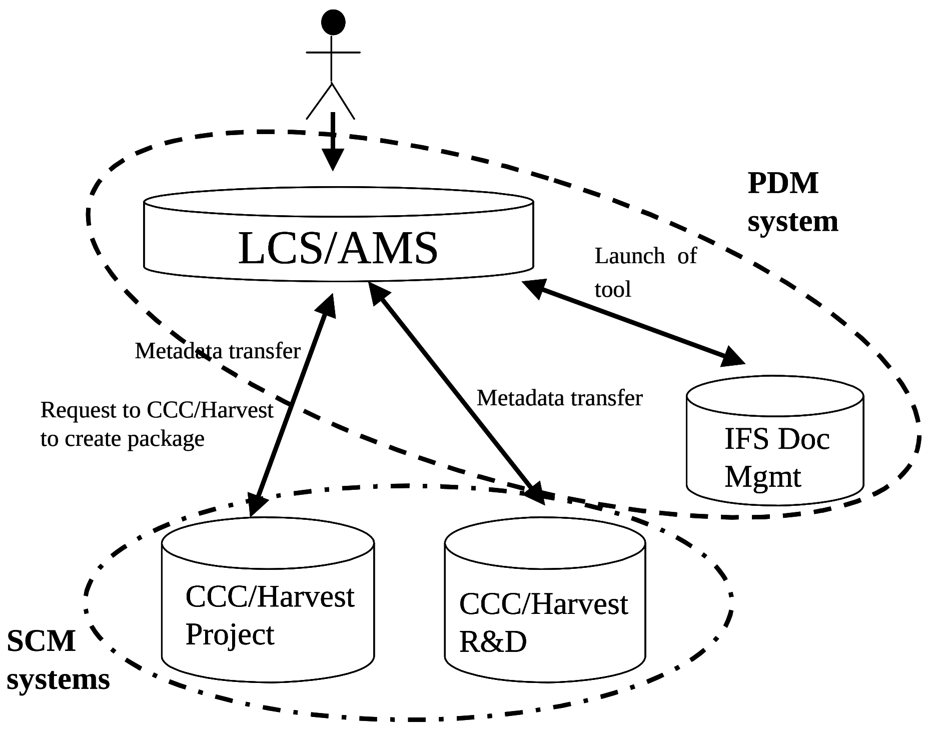

Figure 23. PDM and SCM tools and their interoperability

The systems LCS and AMS build the daily environment used by all employees. The in-house-built system LCS is managing customer information, and is the global product configuration system. AMS (in-house built system) is managing all marketed products and product packages. The IFS Document Management is used for managing all documents.

The commercial SCM system CCC/Harvest [40] is used for managing software during development, patches and customer adaptations. There are several SCM repositories worldwide in the company; the CCC/Harvest R&D repository used for latest delivered source code, and CCC/Harvest project repository for all customization and patches.

In the PDM system data represents of metadata, e.g. the patches and modules delivered to a specific customer, component description, part numbers, package id, and versions of delivered parts, and in which SCM repository the software is stored.

In the SCM system data is represented as software files packed into product packages or patches. A package is a container with the files.

The PDM system manages metadata (business items) and no files (data items) except for documents. A new business item is created when a new product version or patch is provided. In the SCM system packages (containers) are managed. A package is created when one or several included files have to be changed. No branch and merge facilities are provided in the SCM tool.

The PDM system is installed at one geographical site and in accessible through a user web interface. No distribution of data is managed. The SCM system does not support distributed environment or replication of data.

From this and from the figures 22 and 23, the PDM system together with the SCM system supports the product lifecycle process for their software product. The SCM system supports the concept of packages inherited by the company as their internal development process.

The IFS document management tool is used for managing and archiving the documents. The tool includes a document management process.

Since the SCM system supports the package concept, the selection file versions are built-in in the package. Software files included in a specific package are all compiled. No selection for a build is needed.

6.2.3 Integration Initiative

When a customer order is registered into the PDM system, the metadata is transferred to the SCM system, see figure 22. Information about the customer and packed id is transferred. If an order for a new customer is received, PDM requests the SCM system (CCC/Harvest R&D repository, see figure 22) to create a package including the specific customer components and its versions. If a patch needs to be developed, metadata from PDM is automatically transferred to the SCM system to identify the specific customer product versions. An empty package (container) for the deviations (bug fixes) is automatically created in the SCM system (CCC/Harvest project repository). If a package is manually created, isolated in the SCM system without knowledge in the PDM system, the package metadata can be transferred to the PDM system.

The different systems forming the PDM system are integrated. These interfaces are in-house built. The interface between PDM environment and SCM is in-house built. PDM transfers metadata automatically to the right SCM system when triggered. This integration is an example of a loose integration [2] between an in-house built PDM system and one specific commercial SCM tool. Since the commercial SCM tool does not support branch and merge, the integration does not require that much efforts. Integration between in-house built systems does not require that much effort compared to commercial integrations.

Benefits for the company with the integration is (i) secure the information in the systems are correct, (ii) increased traceability, and simplified naming conventions. The company has adopted the terminology and the methodology from the commercial SCM system into their processes to reduce misunderstandings and for an easier integration.

The drivers behind the integration initiative were (i) increased traceability of customer installations of software and their versions, and (ii) simplified environment for all users in the company to enable common naming conventions and common product lifecycle process.

We can conclude that there are fewer problems to integrate systems when (i) the PDM system consists on in-house built tools into an environment, (ii) the integration is internally built, and (iii) there are no hardware components and supporting tools to manage. In addition, there is no problem with differences in processes.

6.2.4 Culture Aspects and Organization

In the company all stakeholders use both the PDM and SCM tools. The tools form an environment, which all stakeholders are using in their daily work. No major problems were detected in using both PDM and SCM tools. Since most of the tools were in-house built, new or enhanced functionality could be introduced more often compared to a commercial tool.

Common terminology, reused from the SCM tool, is used within the processes. The company specific dictionary is mainly used for acronyms and not heavily used. No specific collisions between different terminologies within different tools are found due to mostly in-house built tools and reuse of terminology from the commercial tool. There is a systematic support for education. One specific course has been accomplished where different culture aspects such as thinking and acting in particular cases are discussed and highlighted. Roles, titles, and organizations are described. General update meetings are held for increasing knowledge of the total product portfolio. These meetings are held for all staff.

6.3 Findings for the case studies

We found, that all companies are in dynamic state, where new tools, processes and technologies, and faster time to market, demand a more efficient development and management of complex products. Lack of integration of tools managing information of the products is one of the obstacles for a more efficient development, which we found in most of the cases. Even companies developing software products only, build integrations themselves and to use simpler PDM systems in order to manage the products sent to the market. In some cases, integration efforts of PDM and SCM tools show how complex this task is. Most of these integration efforts are done on in-house built tools, which are easier to perform due to full control of the functionality growth, compared to commercial tools. The integration efforts we found in our cases are based on loose integrations only. In most of the cases, distributed development was performed in developing products. For software products, the SCM tool supported the distributed development, except when a low-level SCM tool was used with no such functionality available. For hardware products, distributed development was either not used or documents were sent by e-mail. The product structure was in most of the cases used to minimize concurrent engineering on the same product.

We have also seen how complex the development process and information flows are. A lot of development tools are used, and often the humans are the integrators of the information. In some cases, there is a clear need for simplifying and improving the development processes. When integrating the product on a system level, we found that in most of the cases, the information was time-consuming to find. Not always the right version was found and used. No case show a product life cycle process with integration points for hardware and software components.

Culture differences, which can be observed between hardware and software developers, are found in most of the cases. This has in some cases been taken care of by internal education in both areas, especially on the processes. Mostly, the hardware and the software developers are organized separately and do not have the possibility to spend time for discussions and further understanding of each other's domain. Further, on most of the cases, a common company terminology were defined and implemented in order to minimize misunderstanding.

A trend, however, is that many companies understand the benefits of building interoperability between PDM and SCM, and thus minimizes manual transfer of information.

We can conclude that the case studies confirm the hypothesis of important factors such as tools and technologies, processes, and culture are important when providing a successful integration between PDM and SCM.

7 Conclusions

In a rapid expansion of computer-based systems developers from different engineering domains are enforced to work together. This collaboration enables significant improvements when complex products are developed and manufactured, i.e. when the development process has high demands on efficiency and quality. However, the challenges to achieve this quality are many, not only in the technologies of the particular domains but in the coordination, interoperability and integration of these domains. A characteristic example of such challenges is the integration of PDM and SCM tools, which provide information and management support for the development and maintenance of hardware and software assets, respectively. Many companies developing and manufacturing products that include both software and hardware components face this problem of building up an integrated support of these products. The initial steps towards an integrated development and production environment and an integrated process are painful; there are a number of unsuccessful or only partially successful attempts to integrate functionality available from these tools. In this report we have shown why such integration is so difficult. First, the functions that the tools from these domains provide are in general similar but in principle very different. Second, the pure technical solutions for integration are not sufficient; a total coherent and integrated process is as important as the technical ability of integration of the tools. Finally we have experienced that the cultural differences between domain engineers play an important role. A lot of efforts must be put in removing cultural barriers, in education and in building common understanding to make it possible to introduce a new integrated support for the entire development process. Our findings are also that loose types of integrations in which developers can keep their old tools and local processes are more feasible than tight integrations requiring a new information model and entirely new processes. Again, the reasons are not only of technical nature, but very much of cultural.

From a system level, there are requirements on managing the whole product irrespective of its contents of hardware and software components, i.e. interoperability in the information flow. The development processes for hardware and software development, although similar, distinguish on a detailed, practical level. SCM and PDM have different production phases; PDM with high cost, long lead-time, and another organization involved, and SCM short and cost effective with no other than the developer team performing the product manufacturing involved in the production phase. PDM-related and SCM-related standards in CM exist, but they are too vague and too little integrated in PDM and SCM to be used as a common integration factor between PDM and SCM.

From the analysis of basic characteristics of PDM an SCM tools we find that there are similarities in them, but that the underlying concepts are quite different. PDM tools support, document management, product structure management, distributed development and awareness of changes of documents. Of these features an SCM tool does only support awareness of changed documents and an effective replication between sites. On the other hand SCM tools support concurrent engineering on file level, and replication without locking on file level. A PDM tool does not support these features. Using PDM tools for development of software would be very difficult and inefficient. Using SCM for hardware products would be practically impossible.

Since hardware and software designers are focusing on different activities, they have both low knowledge and understanding for each other's requirements due to organizational, cultural, and domain specific behavior. On top of this, the terminology is almost the same but with different meanings. For integration purposes, terminology and cultural differences are key factors to highlight. A common understanding for both domains and terminology is essential to provide when integrating these domains.

8 References

[1] U. Asklund, I. Crnkovic, A. Hedin, M. Larsson, A. Persson Dahlqvist, J. Ranby, and D. Svensson. *"Product Data Management and Software Configuration Management - Similarities and Differences"*, The Association of Swedish Engineering Industries, 2001.

[2] Crnkovic I., Asklund U., and Persson Dahlqvist A., *"Implementing and Integrating Product Data Management and Software Configuration Management"*, ISBN 1-58053-498-8, Artech House, 2003.

[3] Estublier J., "*Software Configuration Management: A Roadmap*", In Proceedings of 22nd International Conference on Software Engineering, The Future of Software Engineering, pp. 279-289, ACM Press, 2000.

[4] Svensson D. and Crnkovic I., "*Information Management for Multi-Technology Products*", International Design Conference - Design 2002, IEEE, 2002.

[5] C. Karlsson, E. Lovén, *"Utveckling av komplexa produkter – integrerad mjukvara i traditionellt mekaniska produkter"* Institute for Management of Innovation and Technology, IMIT report IMIT 2003:127.

[6] Estublier J., Favre J-M., and Morat P., "*Toward SCM/PDM Integration?*", In Proceedings of Software Configuration Management SCM-8, Lecture Notes in Computer Science, nr 1439, pp. 75-94, Springer, 1998.

[7] B. Westfechtel, and R. Conradi, *"Software Configuration Management and Engineering Data Management: Differences and Similarities"*, Proceedings of 8th International Symposium on System Configuration Management (SCM-8), Lecture Notes in Computer Science, No. 1439, Berlin Heidelberg, Germany: Springer-Verlag, 1998, pp. 96-106.

[8] CIMdata, www.cimdata.com, November 2005.

[9] The PDM Information Center, www.pdmic.com, November 2004.

[10] A. Leon, *"A Guide to Software Configuration Management"*, ISBN 1-58053-072-9, Artech House, 2000.

[11] D. Whitgift, *"Methods and Tools for Software Configuration Management"*, ISBN 0-471-92940-9, John Wiley & Sons Chichester, New York, Toronto, Brisbane, Singapore, 1991.

[12] M. Kelly, *"Configuration Management – The Changing Image"*, ISBN 0-07-707977-9, McGraw-Hill, Berkshire, 1996.

[13] F. J. Buckley, *"Implementing Configuration Management"*, ISBN 0-8186-7186-6, IEEE Computer Society Press Los Alamitos, Washington, Brussels, Tokyo, New York, 1996.

[14] S. Dart, *"Configuration Management – The Missing Link in Web Engineering"*, ISBN 1-58053-098-2, Artech House, 2000.

[15] Persson Dahlqvist A., Crnkovic I., and Larsson M., "*Managing Complex Systems - Challenges for PDM and SCM*", In Proceedings of International Symposium on Software Configuration Management, SCM 10, 2001.

[16] Persson Dahlqvist A. Crnkovic I., and Asklund U. *"Quality Improvements by Integrating Development Processes"*, In Proceedings of Asia-Pacific Software Engineering Conference, APSEC2004, 2004.

[17] Silberschatz, H., F. Korth, and S. Sudarshan, *"Database System Concepts"*, 3rd ed,. New York: McGraw-Hill, 1997.

[18] Kroenke, D., *"Database Processing: Fundamentals, Design and Implementation"*, Upper Saddle River, NJ: Prentice Hall, 1996.

[19] W. Tichy, *"Configuration Management"*, ISBN 0-471-94245-6, John Wiley & Sons Chichester, New York, Brisbane, Toronto, Singapore, 1994.

[20] Asklund, U., *"Configuration Management for Distributed Development – Practice and Needs"*, Licentiate Dissertation No. 10, Department of Computer Science, Lund, Sweden: Lund University, 1999.

[21] Feldman, S. I., Make, *"A program for Maintaining Computer Programs, Software – Practice and Experience"*, Vol. 9, No. 4, April 1979, pp. 255-265.

[22] UGS PLM Solutions, vendor of the PDM system TeamCenter, www.ugs.com, November 2005.

[23] IBM Rational vendor of ClearCase, www.ibm.com, November 2005.

[24] Serena vendor of PVCS, www.serena.com, November 2005.

[25] Swedish Standards Institute, Quality Management – Guidelines for Configuration Management, ISO 10 007, 2004.

[26] Microsoft vendor of Windows Installer, www.microsoft.com, November 2005.

[27] Install Shield, www.macrovision.com, November 2005.

[28] Open Document Management, API, Version 2.0, Association for Information and Image Management, September 19, 1997.

[29] Feiler, P., *"Configuration Management Models in Commercial Environments"*, Technical Report CMU/SEI-90-TR-11, Software Engineering Institute, Carnegie Mellon Institute, Mars 1991.

[30] ANSI/EIA-649-2004, National Consensus Standard for Configuration Management, American National Standards Institute, New York, 2004.

[31] D. B. Leblang, *"The Challenge: Configuration Management that Works"*, Configuration Management, Edited by W. Tichy; J. Wiley and Sons. 1994. Trends in Software.

[32] J. Estublier, S. Dami, M. Amiour, *"High Level Process Modeling for SCM Systems"*, SCM-7, LNCS 1235 pp 81-98, May, Boston, USA, 1997.

[33] Royce, W. W. *"Managing the Development of Large Software Systems"*, Proceedings of IEEE WESCON, August 1970.

[34] Sommerville, I., *"Software Engineering"*, Sixth Edition, Harlow, UK: Addison-Wesley, 2001.

[35] Kroll P. and Kruchten P., *"The Rational Unified Process Made Easy"*, ISBN 0-321-166009-4, 2004.

[36] G. Hofstede, *"Cultures and Organizations Software of the Mind Intercultural Cooperation and its Importance for Survival"*, ISBN 0-07-029307-4, McGraw-Hill, 1997.

[37] Telelogic vendor of DOORS, www.telelogic.com, November 2005.

[38] J2EE, java.sun.com/j2ee/, November 2005.

[39] SOAP, www.w3.org, November 2005.

[40] OPC, www.opcfoundation.org, November 2005.

[41] Business Service Optimization vendor of CCC/Harvest, www3.ca.com, November 2005.