Product Data Management
and
Software Configuration Management

- Similarities and Differences

Special edition for the conference September 27 2001:

"CM i ett produktperspektiv
- ställer hårdare krav på integration av SCM och PDM"

2

Preface

This report has been written by support of The Association of Swedish Engineering Industries (Sveriges Verkstadsindustrier, VI). The questions the report is trying to answer have been formulated by the 'Teknikråd 12', and of the steering committee for the study. The study has been sponsored by The Association of The Swedish Engineering Industries.

The study was performed from October 2000 until September 2001 by a working team and a steering committee. The report is based on interviews from persons from the industry, study of scientific and industrial literature, discussions with vendors and consultancy firms of PDM systems, discussions with researchers in the SCM and PDM area, study of literature in the areas, and experiences from the steering committee and the authors.

The steering committee has reviewed the report, and approved it. The report has also been reviewed by:

Geoffrey Clemm,   Rational Software, USA;
Jacky Estublier,   LSR Grenoble University, Grenoble, France;
Jan Johansson,   Ericsson Corporate IT
Johan Malmqvist,   Chalmers University of Technology, Göteborg, Sweden;
Tom Maurer,   SDRC, New York, USA;
Mikael Ström,   IV (Industriforskning och utveckling AB), Mölndal, Sweden.

Members of the steering committee

Annita Persson Dahlqvist, Ericsson Microwave Systems AB (chairman)
Ulf Asklund,   Lund Institute of Technology Lund University
Ivica Crnkovic,   Mälardalen University
Allan Hedin   SaabTech Systems AB
Jan-Ola Krüger,   Saab AB
Magnus Larsson,   ABB Automation Products AB
Tomas Nilsson,   Programvaruindustrialisering AB
Johan Ranby,   Eurostep AB
Daniel Svensson,   Chalmers University of Technology and
                         Ericsson Process and Application Consulting
Göran Östlund,   The Association of Swedish Engineering Industries (sponsor)

Authors:

Annita Persson Dahlqvist
Ericsson Microwave Systems AB
Transmission Mobile Systems
431 84 MÖLNDAL
Annita.Persson@emw.ericssons.se

Ulf Asklund
Department of Computer Science
Lund Institute of Technology, Lund University
Box 118
221 00 LUND
Ulf.Asklund@cs.lth.se

Ivica Crnkovic
Department of Computer Science
Mälardalen University
721 67 VÄSTERÅS
Ivica.Crnkovic@mdh.se

Allan Hedin
SaabTech Systems AB
175 88 JÄRFÄLLA
alhe@systems.saab.se

Magnus Larsson
Development and Research
ABB Automation Products AB
721 59 VÄSTERÅS
magnus.ph.larsson@se.abb.com

Johan Ranby
Eurostep AB
111 20 STOCKHOLM
Johan.Ranby@eurostep.com

Daniel Svensson
Engineering and Industrial Design   Ericsson Process and Application Consulting
Product and Production Development   Box 14
Chalmers University of Technology   164 93 KISTA
412 96 GÖTEBORG
daniel.svensson@me.chalmers.se   daniel.svensson@pac.ericsson.se

4

Table of Contents

Summary

Product Data Management (PDM) is the discipline of controlling the evolution of a product design and all related product data during the full product life cycle, historically with the focus upon hardware product design. The term PDM includes the overall description of the topic and the methodology, while a PDM system is a tool you use for managing the data and the processes you have decided to use it for. Software Configuration Management (SCM) is the discipline of controlling the evolution of a software product, with emphasis on the development phase. SCM is to a high extent based upon Configuration Management methodology (CM).

The PDM and SCM domains have evolved in parallel with none or little communication. The complexity of products is increasing rapidly. Products are often complex systems consisting of hardware, software, and related documents, developed by several groups. This put high demands on support on several levels, both for the system level as well as for each group, especially during the development phase. One important requirement is the possibility to integrate product information systems. PDM and SCM is part of this integration, which makes it important for companies to understand both domains. The possibilities to integrate PDM and SCM is one of the key factors in product information management of today.

This is the third report in the area of Configuration Management supported by The Association of Swedish Engineering Industries (Sveriges Verkstadsindustrier, VI). The study was performed during 2000/2001. The report is based on interviews from the industry, study of scientific and industrial literature, discussions with vendors and consulting firms, discussions with researchers in the SCM and PDM areas, and experiences from the steering committee and the authors.

The report describes both the PDM and SCM domains, their similarities, differences, and trends. We propose solutions for different types of integration and suggest further studies.

We found that despite the fact that these two domains are growing and develop functionality that are more and more alike there are still important differences. The history of PDM and SCM and the users within in each domain uses different terminology and they have different requirements in the tools they use. This is different to the integration of CAD/CAM and PDM, where the cultures are very alike. To make an integration possible the PDM and SCM users need a common terminology and information flow description to make communication possible.

The study shows that PDM vendors have not focused on the support for software management. Similarly, SCM vendors have concentrated on support for software development only. In general there is a lack of knowledge of the combined disciplines, and research is needed to find out ways to integrate and interact. We have noticed a trend within both domains to understand the need for co-operation between PDM and SCM. For an integration to occur, however, vendors from both disciplines must co-operate.

8

1 Introduction

1.1   Background

Product Data Management (PDM) is the discipline of controlling the evolution of a product design and all related product data. Historically, PDM have been focused on mechanical design, component classification and retrieval, revision tracking, workflow, sign-off control, version handling and parts of configuration management. Vendors provide interfaces to many different domains, e.g. to electronic computer-aided design framework systems as well as to compound document management systems found in technical publishing. Little has been done to develop capabilities in PDM systems for software management during the development phase.

Software Configuration Management (SCM) is the discipline of controlling the evolution of a software product. Historically SCM was first developed for the aerospace industry in the 1950's. It was used among others in the Apollo Space Program for tracking thousands of changes. SCM tools began in the 1970's with SCCS and Make. SCM is now a successful part of the software tools market. The SCM discipline is based upon Configuration Management (CM) methodology.

The trend in the industry today is to manage the entire product and not the hardware and the software parts separately. An integrated solution would enable users to find relevant product information for the full product life cycle. This requires an overview over systems involved. To make hardware and software design efficient, special tools will still be needed. Automation for these specific tools is an important property. The integration between PDM and SCM is based upon these requirements. Historically, the PDM and CAD environment have worked for a long time to integrate functions. This report is focused upon integrating PDM and SCM.

Today industry and vendors have started to develop interfaces between PDM and SCM systems. This is relevant for automotive, telecommunication, and other types of industry. The PDM and SCM interfaces of today are often implemented with low functionality since the differences between the areas create problems for implementation.

No ongoing research for common models or standard API's (Application Program Interface) between PDM and SCM have been identified as ongoing. On the contrary, projects with aim to cover both PDM and SCM have been closed due to low industry interest. Interoperability between PDM and SCM is a problem on method, model and system level. In all three areas there is a need for new initiatives from researchers, vendors and users across the industry.

This report addresses the area of PDM and SCM by giving an overview description of the areas and identifying similarities and differences. The subject is described both for methods and for the larger tools in the area. Different proposals of integration are described. With the current speed of development of both areas, new vendors and tools

will be established. The report includes descriptions of trends for both PDM and SCM in order to cover this future development.

1.2   Method

The study was performed during 2000/2001. The report is based on interviews from the industry, study of scientific and industrial literature, discussions with vendors and firms of consulstants, discussions with researchers in the SCM and PDM areas, and experiences from the steering committee and the authors.

With an established steering group the content of this report has been written and reviewed by experienced persons with academic competence or a long industrial experience. The review has been done with available experts both from PDM and SCM.

1.3   Intended readers

This report is written for persons in the situation of making decisions, implementing functionality or otherwise perform activities between PDM and SCM. This could be decision makers, project managers, designers, configuration managers or information technology consultants involved in PDM and SCM development.

The report can also be used as an introduction for persons interested in both or one of the areas, looking for a basic overview and basic connections between PDM and SCM. In this case chapters 2, 3, and 4 are well recommended.

1.4   Purpose

The purpose for the study was to:

- give large companies in-depth knowledge about PDM and SCM, and the most common tools;
- give smaller companies knowledge of how far they can use the tools they have today;
- give the target group for the report an in-depth knowledge in both domains;
- 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 from the researchers;
- explain the different terminologies within the two domains.

1.5   Outline of the report

Chapter 2 describs PDM in terms of user and utility functionality, and system architecture. The used terminology is from the PDM area and is explained in the chapter. The chapter is written by PDM community to give an overview for the SCM community.

In chapter 3 SCM is described by its definition, functionality, and architecture. The terminology is explained as well. This chapter is written by SCM community to give an overview for the PDM community.

Chapter 4 analyses the principles and key functionalities in the two domains. You will find an in-depth description of the differences of the principles and functionalities.

Chapter 5 gives an overview of current research and trends in both PDM and SCM, from researchers, vendors, and industries from Sweden and other countries.

In chapter 6 we analyze the needs and solutions for an interoperability between PDM and SCM. We also state an example of usecases describing how to manage information between a PDM system and a SCM system.

Chapter 7 contains the conclusion of the report, consequences and recommendations on future work.

Appendix A gives a description of some PDM and SCM tools. We describe just a few of the tools on the market, and we do not claim that all tools and all functionality will be described.

In appendix B you will find the interviews made. They can be seen as examples of current requirements, solutions, and present status of the integration of PDM and SCM.

2   PDM - General description of PDM

2.1   Introduction

Product Data Management (PDM) is the discipline of controlling the evolution of a product design with the aim to control and manage projects and products, and to provide different disciplines with the accurate product information at the right time in the right format. PDM is a part of the products entire life cycle, and is also a very broad area with respect to the means of how to achieve its goals, which are obtained by defining methods and processes to obey, making plans to follow and by using PDM tools that provide both producer and consumer of product information in their daily work. Please, see Appendix A for some common PDM tools. PDM has traditionally been used for hardware product development. PDM is used in different ways in different industry segments.

It is difficult to describe PDM, methods and computer support in a complete way. In this report (and chapter) our description will not cover the entire area nor all variants of tools. Instead the description below is aimed towards people working within the SCM area, to give them an overview of what PDM really is. A good overview is the one from Cimdata [Cim98], a PDM consultant firm. On the web we recommend John Stark Associates [Stark] and The PDM Information Center [pdmic]. For a more comprehensive description from a research perspecitve we can recommend [EFM98, EFM99].

In this chapter we state a definition for PDM, and we define the different functionalities and system architecture within the domain. PDM systems have been on the market for quite a while. These systems share some fundamentals. However, some common characteristics can be found and we will discuss them in this chapter.

PDM comes with many names such as collaborative Product Definition management (cPDm) and Product Information Management (PIM). For some more common abbreviations on PDM, see Figure 1. Even if there are slight differences in the definitions, the meaning of the terms is still the same.

PDT	= Product Data Technology
CPC	= Collaborative Product Commerce
ePDM	= electronic Product Data Management
ePLDM	= electronic Product LifeCycle Definition Management
ICM	= Intellectual Capital Management
PKM	= Product Knowledge Management
VPDM	= Virtual Product Data Management
PDM	= Product Data Management
CM(II)	= Configuration Management (II)
CM	= Component Management
PIM	= Product Information Management
……..	

Figure 1   *Some common names on PDM [IVF00]*
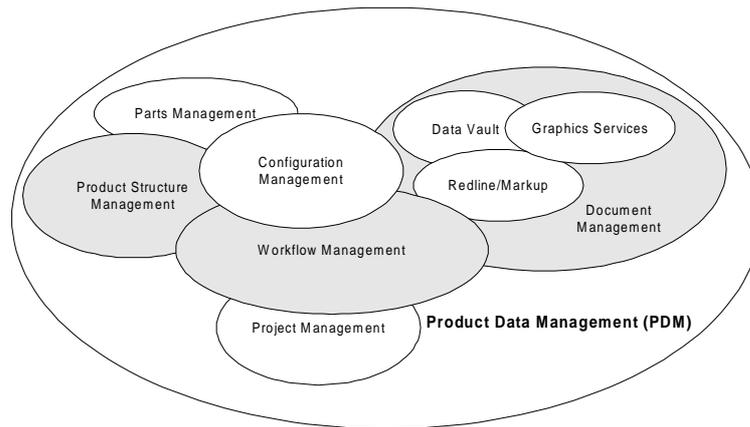
Figure 2   *An example of a PDM application domain, [CIM01]*

There is a discrepancy between PDM and PDM systems. *PDM* is the overall description of the topic and the methodology and is a de facto philosophy with origin from PDM systems. A *PDM system* is a tool you use for managing the data and the processes you have decided to use it for. Before PDM systems, no data management methodologies were referred to as PDM. In this report we mostly use the term PDM to refer to a PDM system.

As defined by CimData, a larger consultant company in the PDM area, a PDM system is:

> *"Product Data Management (PDM) is a tool that helps people manage both product data and the product development process. PDM systems keep track of the masses of data and information required to design, manufacture or build, and then support and maintain products—whether your product is an aeroplane, petrochemical plant, highway, railway system, pharmaceutical, automobile, consumer product, or service. PDM is used effectively in a multitude of industries." [CIM01].*

In the context of today's PDM vision, a PDM system is better described as an information infrastructure – an infrastructure that contains various functional models. Some functionality is delivered by the PDM application itself while others are delivered through integrations with the PDM system. Therefore, PDM is both an application and an information infrastructure that allows other applications to be integrated into the systems environment that it manages. Figure 2 shows one example of a PDM application domain.

2.2   PDM and the Product Life Cycle

The goal and a common definition for PDM is:

> *"PDM manages product data throughout the enterprise, ensuring that the right information is available for the right people, at the right time, and in the right format."[CIM01].*
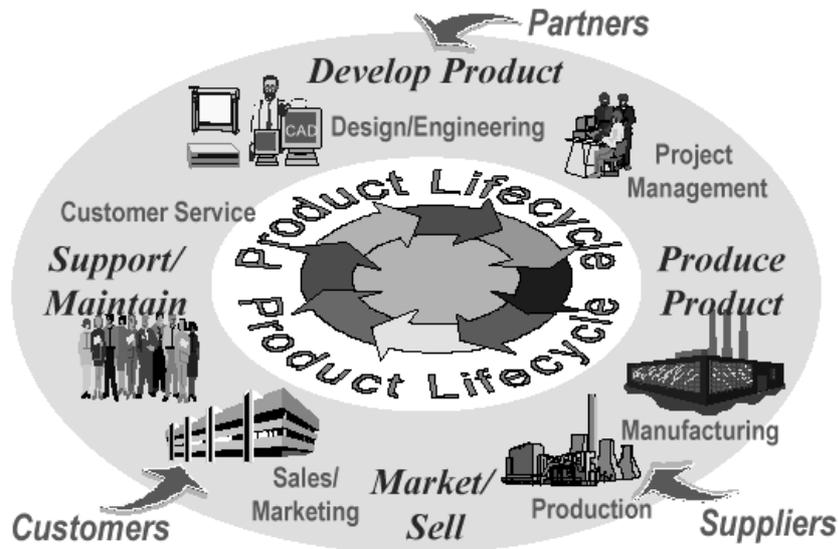
Figure 3   *PDM supporting the whole product life cycle, [CIM01]*

This may be expressed as PDM manages the logistics of the information created during the development of products. A PDM system supports process, e.g. development, manufacturing, marketing, sales, purchasing, extended enterprise (e.g. suppliers, partners). Much of the information in the system is created in the design phase, but this does not imply that it is only the designer that uses the information. All people involved in product development uses it, e.g. designers, manufacturing engineers, purchasers, sales, services, partners, suppliers and customers. The usage of PDM can not be treated as an issue concerning only the development phase, but supports the whole product life cycle as can be seen in Figure 3.

What's a product life cycle? Product Life Cycle Management (PLM) is an acronym that remains unclear to many [Stark]. Part of the problem arises because there are at least three, maybe four, product life cycles in readers' minds [Stark]:

- A market-oriented life cycle of product introduction, growth, maturity and decline;
- The life cycle of a particular product (e.g. a car) which is produced, purchased, used, serviced, and scrapped or recycled. (In this case there's also confusion between life expectancy, life span and actual working life.);
- The life cycle of the information defining the product (created, modified, deleted);
- A life cycle that corresponds to parts moving down the supply chain from suppliers to customers.

The market-oriented life cycle seems to be most common among people working with ERP (Enterprise Resource Planning) systems. The information life cycle seems to be most common among people working with CAD and PDM systems. This leads to the

situation that people from the same "user company" have different understandings of the same thing.

A word commonly used in PDM today, is *collaborative*, as in Collaborative Product Commerce (CPC), collaborative Product Definition management (cPDm), Collaborative Product Management (CPM). These terms describe initiatives aiming at linking the various life cycle perspectives together, see further description of CPC in Chapter 5.

This underlines the importance in any cross-functional system implementation of ensuring that everybody involved - from different parts of the company and among suppliers and customers - has a common understanding of what the target is (the vision) and how it will be achieved (the strategy). Without a common understanding, the implementation is unlikely to success. A similar phenomenon can also occur both when a vendor changes its strategy and when two vendors merge without clear communication of the vision and strategy of the new company to all the stakeholders.

2.3   PDM Systems – Common Functionality

PDM systems and methods provide a structure in which information and processes (used to define, manufacture, and support products) are stored, managed, and controlled. In short, any information required throughout a product's life can be managed by a PDM system, making correct data accessible to all people and systems that have a need to use it.

A PDM system manages the product development process as well as data; PDM systems control product information, status levels, approval processes, authorizations, and other activities that impact product data. By providing data management and security, PDM systems ensure that users can get and share the most recent, approved information. All functions are separated into two categories due to two different main roles, user and administrator. The functionality of PDM systems falls into two categories:

- *User functions*; provide access to the PDM systems including data storage, retrieval, and management. Different types of users may work with different subsets of these functions. A user may be a consumer (views information) or a producer (creates information).
- *Utility functions*; interface between different operating environments and they insulate their complexities from the user. Tailoring enables systems to operate in compliance with the specific user environments.

The user functions can be divided into five categories:

- Data vault and document management
- Workflow and process management
- Product structure management
- Part and component management (classification & retrieval)
- Program management

The utility functions can be divided into five categories:

- Communications and notification
- Data transport and translation
- Image services
- Administration
- Application integration

2.3.1   User Functions
In this section we will describe each function more detailed.

Data vault and document management
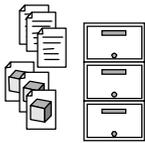
A definition of Document Management is "*functionality for managing documents that allows users to store, retrieve, and share them with security and version control*" [ODMA].

The Document Management area of PDM functionality includes the data vault. This includes security authorization for access to information, support for versioning and revisioning of parts (material masters) and documents, definition and maintenance of relationships among parts and documents and the files used to define the parts, and other related data. Routines for approval and release of documents may also be included in document management or release management.

Workflow and process management
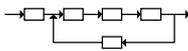
Workflow (or process) definition is a critical part in product definition processes (e.g. release, change, approval) to ensure the right information is available to consumers at the earliest time. This includes defining the ordered steps in the process, the rules associated with the steps, the rules for approval of each step, and the assignment of people to provide approval support. Process management provides the mechanisms to actually enforce the defined processes and approval authorizations. When a process is executed, the PDM system monitors and controls it.

Product structure management
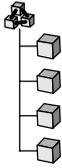
This area addresses to structure the product into a component or product structure where each part is related to one or many other parts.

- Configuration control; facilitate the definition and management of product configurations;
- Manage the development and selection of product variants including platforms, options, alternates, and substitutes;
- Link product definition data to the structure;
- Allow various domain specific views of a product structure;
- Transfer product structure and other data in both directions between PDM and ERP systems.

Part and component management (classification & retrieval)
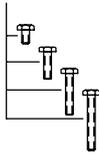
To support re-use of standard components, they can be classified and information (e.g. supplier, supplier part number, and supplier information management) about them stored in the PDM system. To find a standard component, searches can be performed based on the values of attributes defined in the system for a particular item or object. Attributes can include those that are standard when the system is installed as well as additional, user-defined attributes created for specific objects.

Program/project management

Program management provides functions to define a work breakdown structure; a hierarchical network of tasks and sub-tasks necessary to complete a project. This structure can relate to the product data, such as part, documents and change items, which makes it possible to see how a project proceeds in terms of the completeness (status) of e.g. documents, parts, assemblies, BOMs, and products.

2.3.2   Utility Functions

Communication and notification

Designers and others know as quickly as possible when a product is ready to be processed through next task and which information is the most up-to-date. The notification of actions is done automatically.

Data transport and translation

All data are stored and accessed under control of the PDM system, so a user does not need to know where in the computer network the data is stored. The system keeps track of the locations and allows the users to access it if they have the right permission. Information is easily moved between different systems – users do not need to be concerned with operating systems and network commands. In a PDM system pre-defined translators can be used for converting data between applications.

Image services
Images are stored and accessed as any other data. E.g. raster and vector image and PDF viewing allows users to view drawings and other design data.

System administration
A PDM system includes administrative functions such as role management, creating workflows and administer the data storage. PDM systems can be customized in many ways. The data model can be changed, new functionality implemented and the user interface tailored to satisfy particular needs.

Role management
Role management is often treated within system administration, but we have chosen to describe it separately. A product's life cycle involves many roles, such as mechanical designer, software designer, project manager, service support, etc. The roles perform various tasks and therefore need different access rights to product data and should only be allowed to perform particular actions on it. There are practical reasons for this, like preventing users to change data they are not allowed to. Role management includes functionalities such as setting up user accounts, groups, and access rights.

Application integration
Integration with other applications is an important feature. Integration with authoring, visualization, and other collaborative tools are important to establish a single source of product data where information is created once and used throughout the product development process, and to electronically co-locate geographically dispersed authors and consumers.

There are integrations ranging from less complex integrations with text editors to tight integrations with CAD/CAM and ERP (Enterprise Resource Planning) systems. To integrate the text editor (e.g. Word, FrameMaker) includes to add functions such as check-in and check-out of documents to the text editor. This will support the user automatic transfer of documents without manually work and less knowledge about the PDM system. Integrating CAD/CAM systems is more difficult since CAD/CAM systems manages various relationships between the parts, both parts structures and system specific relationships, a network of interrelated files. Integration with ERP systems includes transfer of parts structures with attributes. Part data is critical for the manufacturing process.

2.4   Product Structure and Document Management

The information in a PDM system is based on an object oriented product data model. Objects used to represent parts, assemblies, documents and other kinds of objects are named *Business Items* or *Business Objects*. A PDM system manages files as well. A file is represented as a *Data Item*. Further on we will use the terminology *Business Item* and *Data item*. Relationships are used between objects to relate them to each other. A product structure is built of objects and relationships together with attributes. Product structures uses a special class of *Business Items* (*Structured Business Items*) that have special
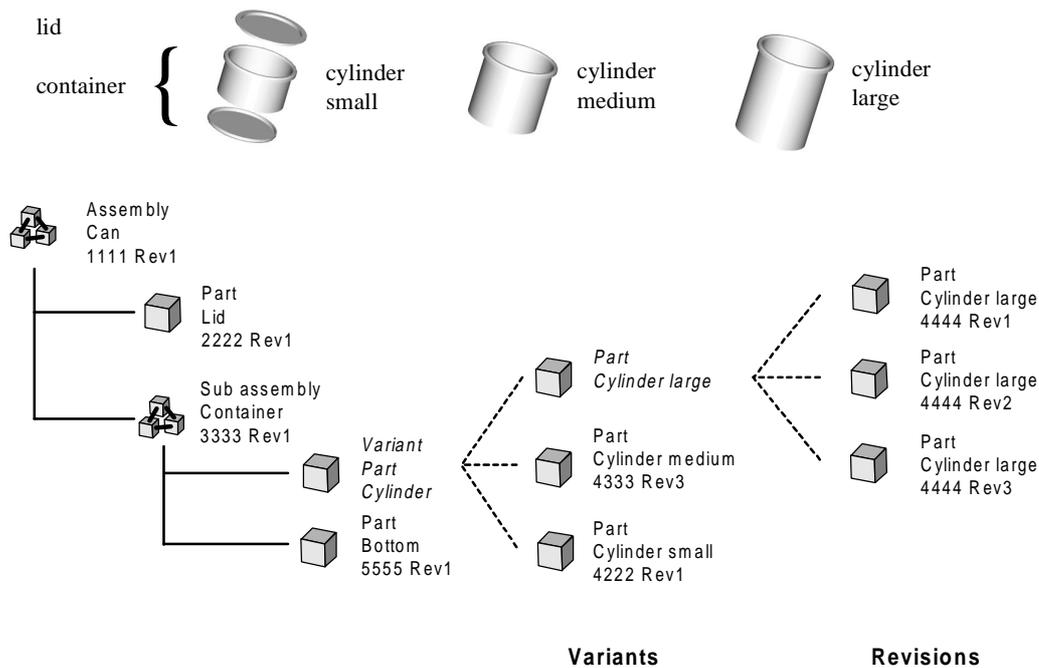
Figure 4   *Example of a product structure describing a can with three variants.*

relationships to support BOM (Bill-Of-Material) requirements for usage, effectivity and physical identification (serialization). An attribute can be defined on an object or on a relationship.

2.4.1   Product Structure Management

One definition of a product structure is:

A product structure is a division of parts into a hierarchy of assemblies and compo-nents. An assembly consists of other assemblies (sub assemblies) and/or components. A component is the lowest level of the structure.

The business items in a PDM system can be used to represent any kind of object describing a product, e.g. requirement objects describing the requirements of a product, functional objects describing the functions of a product. The most common kind of objects, describing a product, are parts (same as components in the definition above). They are connected with relationships to build the product structures, see Figure 4.

The product structure has its origin in the BOM. A BOM is used in manufacturing to collect all assemblies and parts building the final product. Historically, the BOM was created and maintained by the design department. Later on the BOM was implemented in PDM systems. All product related information is connected to the product structure.

The parts structure in Figure 4 is based on parts and assemblies. Each object is of a certain type (part, assembly, sub-assembly, etc.), has a name (Can, Lid, Container, etc.), a unique part number (1111, 2222, 3333, etc.) and a revision (1,2,3, ...). A product may have several variants; the "Can" in the example is manufactured in three sizes. The cyl-
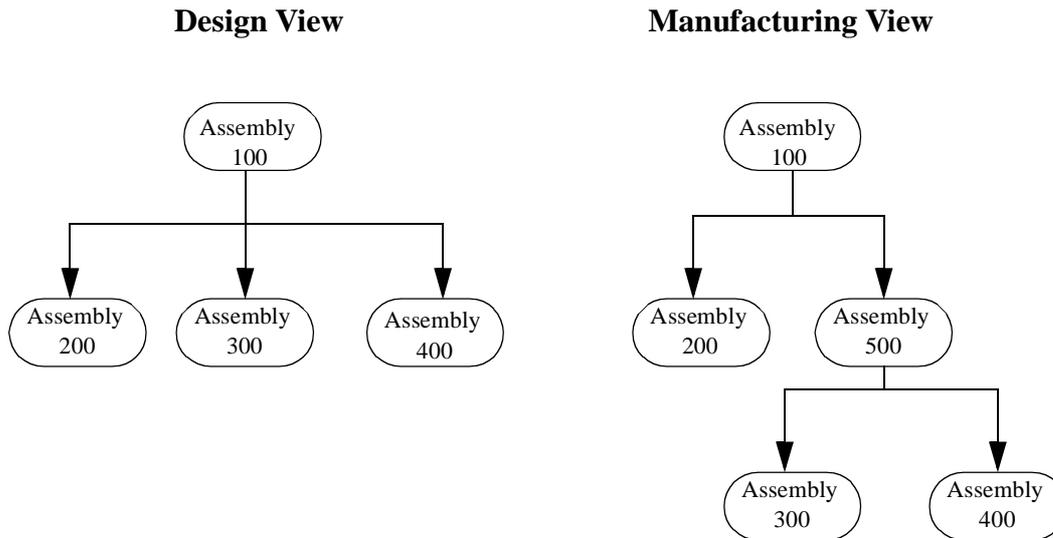
Figure 5   *An example of different views*

inder part has three variants: small, medium and large. For the variant "Cylinder large", all three revisions are shown to illustrate that an object can exist in several revisions.

All PDM systems support variants of parts. The relationships between the parts may contain rules, used for selection of alternative parts. The three different container variants: large, medium, and small, can be selected with rules. One rule can be: "if volume larger than 2 dm^3; then use Cylinder large". These kind of rules are defining what to include in a product. There are also other (exclusive) rules defining invalid combinations of parts and assemblies.

To define when a part is valid in a product, *configuration effectivity* is used. The effectivity of a part is either a time frame or a version interval. The configuration effectivity defines when the part is valid in a specific configuration. This information is used in manufacturing. The configuration effectivity is an attribute on the relationship, not an attribute on the part objects itself, since a part may exist in several configurations with different effectivity.

In an enterprise, product structures are used by different disciplines. The duties and responsibilities may require different views of a product structure. A designer and a manufacturing engineer need to see the structure from different views, as shown in Figure 5. Examples of other groups that could need their own views are maintenance, purchasing and sales.

2.4.2   Document Management
The purpose of a document management system is to make it easier to share documents across an enterprise while at the same time maintaining the integrity of the documents. PDM allows you to check in documents to central locations (referred to as vaults) and controls the altering of documents once they are checked in to these locations. The system allows you to query for a document, by searching for key attributes (e.g. document
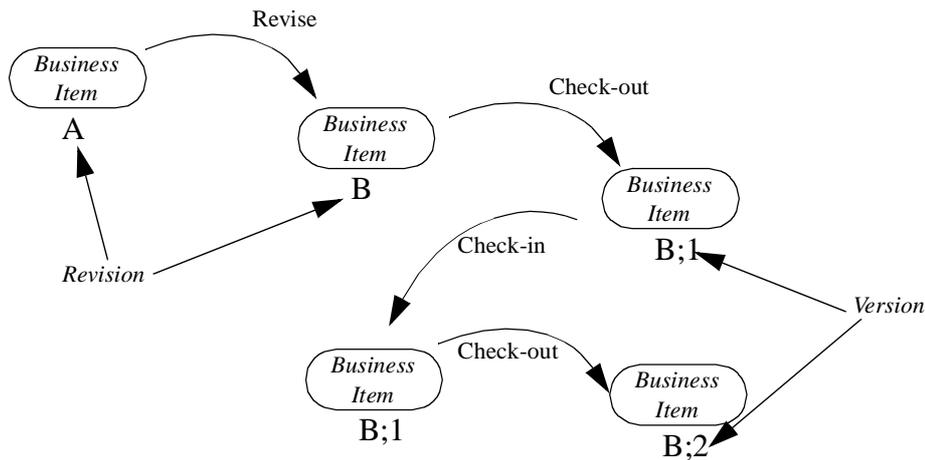
Figure 6 *Example of how the revisions and versions are connected to each other in PDM*

title or document number) or free text search. PDM allows multiple viewers to view (but not modify) checked-in vaulted documents; however, only one user at a time can check out a document and modify it. When you check in a modified document, PDM places the original version of the document in a vault or replaces it with the modified version, depending on site settings, and makes the modified version available to other users. By implementing version control, PDM ensures that only one controlled, current version of a document is available.

In many cases, PDM extends a function that is performed on a business item to also apply automatically to the data item. For example, when you copy a business item in PDM, the data item is also copied, given a new name, and attached to the new business item.

2.4.3   Version Management
Version Management in PDM consists of *revisions* and *versions*. A *revision* is a successor of a business item. A user freezes a business item and can then revise it. PDM creates a new succeeding revision and assigns a revision letter to it. The revision sequence for an business item is A, B, C... A user can work only on the most recent revision. A *version* is a sequence of a revision. When a user checks out a business or data item, PDM creates a version and assigns a sequence number to it, starting with number 1 (for example, revision A, sequence 1). Users can work only on the most recent version. Please see Figure 6.

2.4.4   Release Management
When it is time for delivery to an internal (e.g. system integration and verification, and manufacturing) or external customer, the product package (the content of the delivery) is well defined in the product structure. The product package component has relationships to included components and documents, and thus has full traceability.
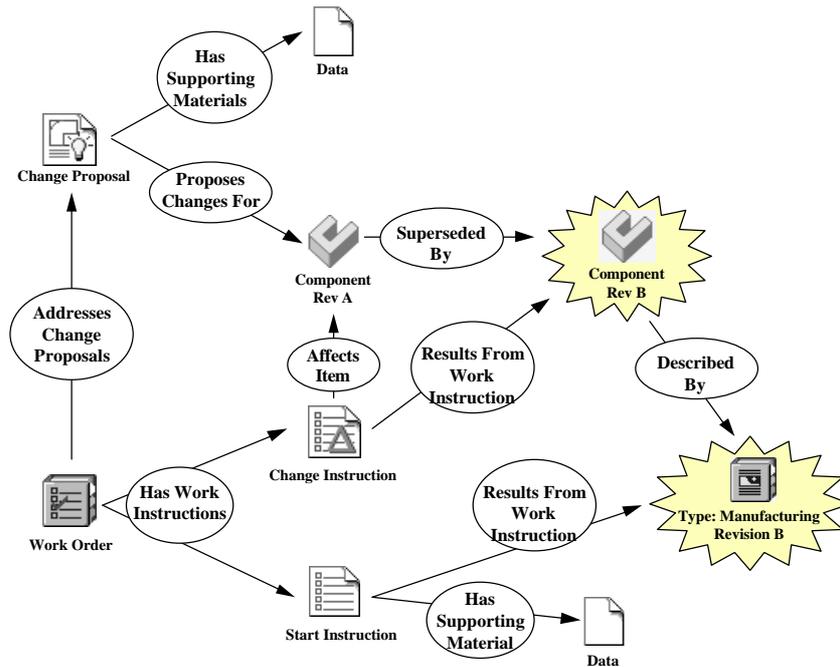
Figure 7   *An example of an implemented change management process [SDRC]*

2.4.5   Change Management

The Change Management performed in PDM systems is close to CM standards [ISO95] and practice. In a PDM system a controlled change environment is defined. A strict change control may be employed to ensure no work is done on product data without authorized work orders and change instruction. In PDM systems, rules can be added to the user environment to determine what classes of *business* and *data items* require authorization. As *business* and *data items* are modified or added, the relationships to the work instructions and work orders are maintained to provide a way to track progress of the change through the system.

In the example in Figure 7, a component Revision B has been added which implements the change documented by the change instruction. In addition, the document folder containing the manufacturing instructions has been revised to contain additional information about manufacturing the new component.

2.4.6   Change Approval - an Example of Workflow and Process Management

The final step in a change management process sends the modified and added business and data items through a life cycle to approve the changes (Figure 8). This step is the same whether or not companies employ strict change control. Upon approval, the new and modified *business* and *data items* pass from a WIP (Work In Process) to a released vault area, and the change becomes the new revision. After the work is complete, the newly revised items continue to refer to the work instructions and work orders that generated them. This provides a valuable historical picture of the evolution of the design
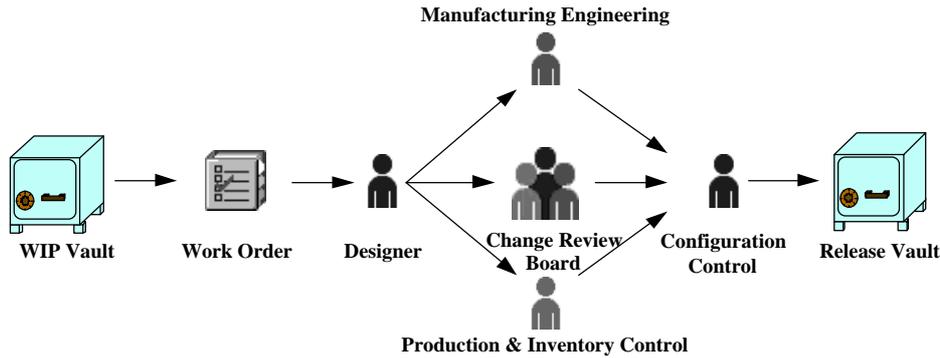
Figure 8   *An example of a workflow for change approval [SDRC]*

which allows users to learn from design approaches which have been implemented in the past.

2.5   System Architecture

PDM systems are composed of an electronic vault or data repository (mainly document files), a set of user and utility functions, and PDM database (metadata)

In Figure 9 the corporate server stores common information used of other servers. This information defines where the servers must be able to access and modify data and where all other servers are located in the network, i.e. the corporate server is the master and the other servers are slaves. The local area server provides services to network geographically separated from the corporate server. A workgroup server runs one or more database servers. The workstation runs the client software.
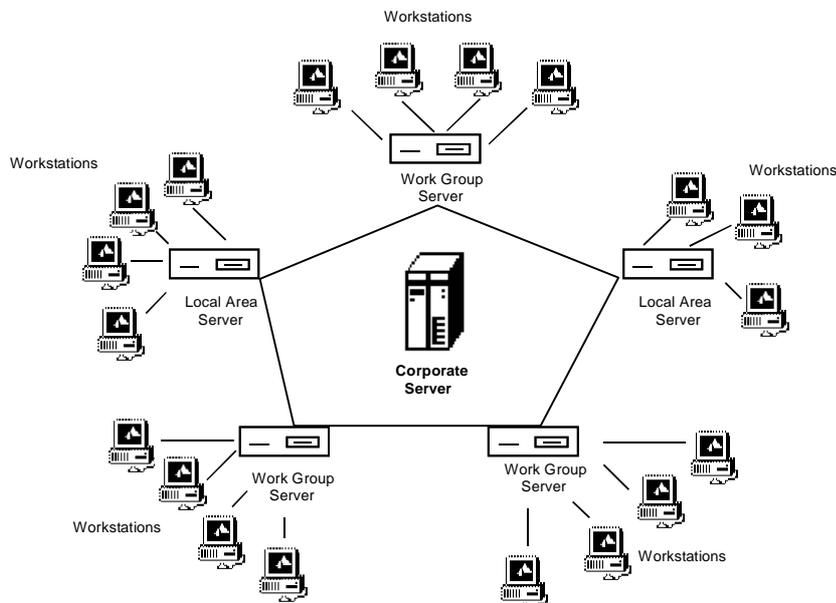
Figure 9   *An example of a system architecture of a PDM system [SDRC]*
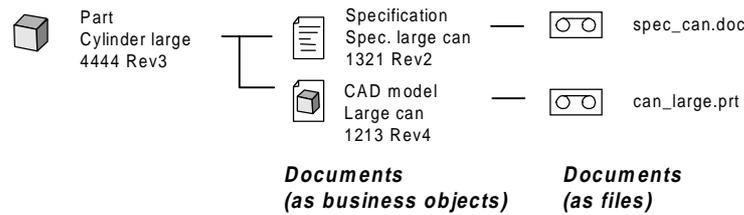
Figure 10   *Illustration of how documents can be represented by both Business Items and Data Items*

2.5.1   Data Representation

PDM makes a distinction between an object and the file containing the data, e.g. a document. The documents are stored as files in their native formats or standard formats in a *data item*. A document is both represented by a *business item* (containing the metadata) and stored as one or more *data items* (files) in the PDM system. The reason is to separate metadata from the content or actual data. A *data item* can be reused from several *business items.* This is not possible in a standard file system. Figure 10 shows how a document is managed by its metadata through the *business item* and the file itself in the *data item*.

2.5.2   PDM Database

PDM manages metadata and file data. Metadata is stored in the database and file data stored in PDM controlled file locations. The metadata is used to support PDM functions.

In a distributed environment, administration and user data is always replicated. Other metadata is replicated as needed. File data is also replicated as needed. Performance problems are mitigated by event driven replication. The metadata is used when the user does a query. Metadata contains a reference to the actual place where the *Data Item* (e.g. document) is stored in the network.

2.5.3   Electronic Vault or Data Repository

An electronic vault is used as a repository to control the product information. The vault is a logical data storage used to store and manage access to electronic documents and files that are produced by various applications. Depending on the chosen installation strategy, the administrator may choose either to incorporate all files (*data items*) into the PDM database or just to store the references to the actual file in the file system in the database.

A document that should be controlled by the PDM system is checked in to it. The document is now available to view or to check out for all users. To change a document, a user checks it out. While a document is checked out, it is locked for changes by other users. When you check out a document, the file will show up in your private work location, a place which is the working area for each user. You change the document and when the
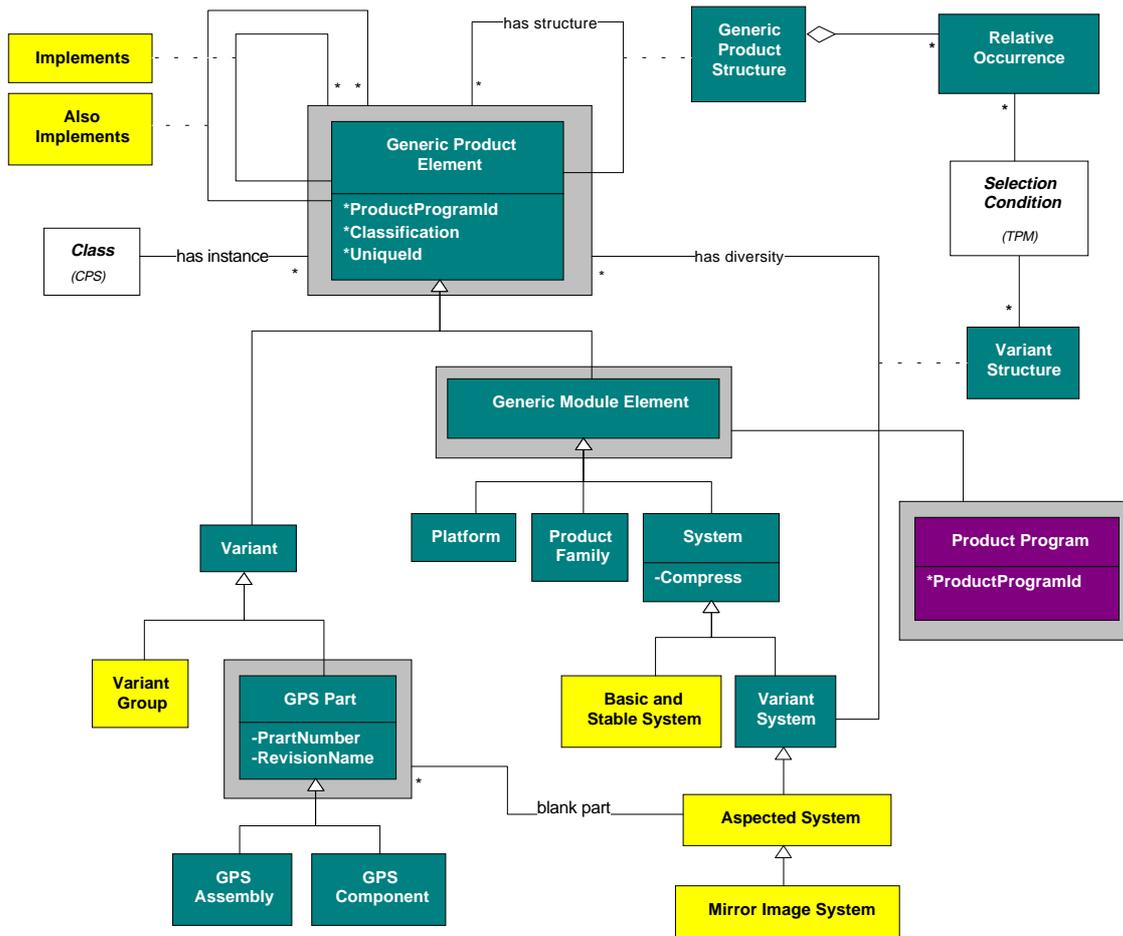
Figure 11   *Example of a Data Model*

changes are done, the document is checked into the PDM system again. The system unlocks the file and other people may check it out for other changes.

There is no concurrent development on documents or other files in the PDM systems.

2.5.4   Data Model

The database implements a data model, which describes the types of objects, relationships and attributes used in it to represent business items, data items and relationships between them. Several PDM systems (e.g. [SDRC, eMatrix, ENOVIA]) have a data model partly based on the standard [STEP 91]. However, most systems only partly follows the standard. The model can be changed to better match the needs of a particular business model in a company. In Figure 11 you see a data model describing a generic product structure. The example is selected from Metaphase, [SDRC].

2.5.5   Distributed Development

To be able to perform distributed development with people located at various geographically dispersed sites, data has to be available on all sites. This must be done in a controlled way, avoiding inconsistency of data. PDM has distributed replicated databases,
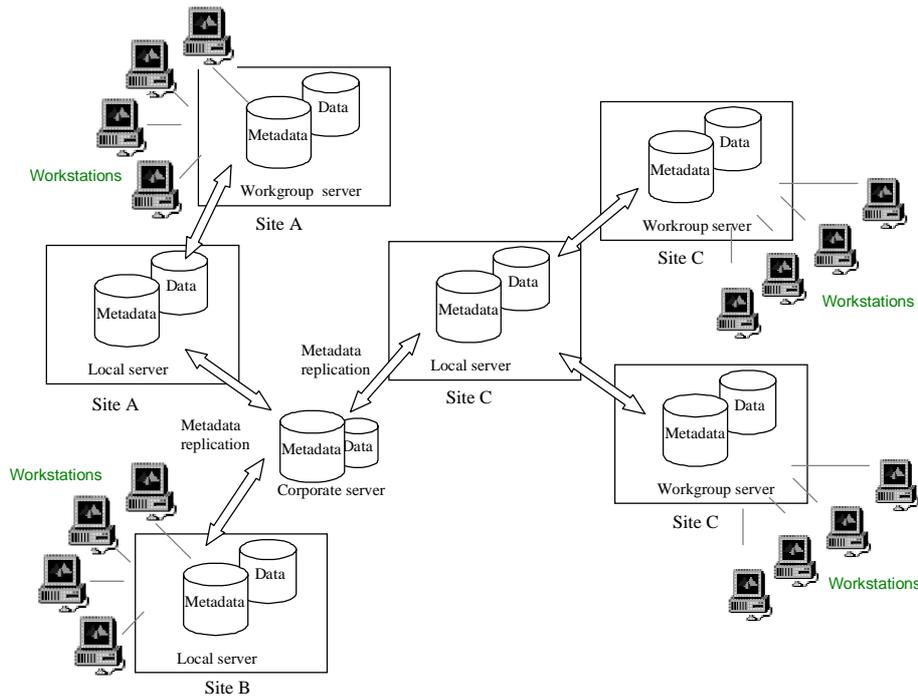
Figure 12   *Example of a distributed environment in a PDM system*

in which it is possible to replicate metadata or both metadata and data throughout the network, which is illustrated in Figure 12.

The metadata must always be distributed up in the hierarchy, towards the corporate server. Metadata stored on a workgroup server is distributed to the local server, and then to the corporate server. When a user search for metadata the workgroup server is searched first. If this metadata is not found on the workgroup server, the local area server is searched and finally the corporate server. If the metadata contains a link to a data item, this link can be used to fetch the data item from its location.

The data items (files) are stored locally on the workgroup servers. This gives the teams located at the sites good performance. In some PDM configurations the data is distributed to other sites as well. This will cause performance overhead.

3   SCM - General description of SCM

3.1   Introduction

Software Configuration Management (SCM) is a discipline within software engineering with the aim to control and manage projects and to help developers synchronize their work with each other. SCM is part of the entire development life cycle, and is also a very broad area with respect to the means of how to achieve its goals, which are obtained by defining methods and processes to obey, making plans to follow and by using SCM-tools that help developers and project managers with their daily work.

In this chapter (and in this report) we will not dive into the details of SCM, nor do we cover the entire area. Instead the description below is aimed towards people working within the PDM area, to give them an overview of what SCM really is. For a more comprehensive description and for low-level technical details we can recommend earlier reports from The Association of Swedish Engineering Industries (Sveriges Verkstadsindustrier in Swedish) about CM in general [Nym96] and CM for distributed development [Ask99a]. Some other good references are: books [BW98, Whi91, Kel96, Dar00], proceedings from ten international conferences (latest is [André01]), and a lot of information on the web [YP, App01a].

The base of SCM is CM. Most of the problems and solutions are almost the same for SCM and CM, but there are some additions and some parts that are more important and crucial when managing software than other items (e.g. hardware). One difference between software and hardware may be that for software the model used during development is almost the same as the product delivered, which is not the case for hardware (a design is not the same as the product produced following the design). Another difference is that transitions between the design phase and production is more vague for software development and it is often the same person or team doing both. Compared to hardware it is also very easy (automated) to build the product from the model, which makes it more tempting to allow changes also close to a release. Unfortunately this is rarely a good idea. Also a lexically small change may have a lot of effect on the behavior of the product. Due to these circumstances SCM has really focused on the problems related to changes, e.g. version control and change management.

In the rest of this section we discuss CM in general, e.g. a CM standard, and SCM. We then continue to talk about aspects important especially for SCM.

CM has during the past years been considered more and more important due to several reasons. One reason is the influence of the well known SEI Capability Maturity Model (CMM) [SEI95], which have pointed out SCM as an important ('key process') area to achieve level 2 on its 5 level scale. Another important reason is the fact that software is getting larger and more complex and needs the support from CM. One example of how the situation gets more complicated is the shorter time-to-market which requires incremental and concurrent development models which, in turn, increase the burden on CM.

Another example is the trend that developers are getting more dispersed, although still developing the same system.

A definition of CM made in previous reports from The Association of Swedish Engineering Industries is:

> *CM is the controlled way to manage the development and modifications of systems and products, during their entire life cycle.*

This is, however, only one of many definitions. Appleton have collected a lot of them on his home page [App01b]. One reason for the many existing definitions is that CM has two target groups with rather different needs: management and developers. The need for CM was first noticed by the managers who wanted more control and measurements over the development and in particular over the releases of the products. This need was met by manual routines often managed by a CM librarian. Today all developers are involved in CM, which is highly automated by sophisticated tools meeting not only the managers needs, but allow developers to be more effective and more aware of what is going on within a project.

From a *management perspective*, CM directs and controls the development of a product by the identification of the product components and control of their continuous changes. The goal is to document the composition and status of a defined product and its components, as well as to publish this such that the correct working basis is being used and that the right product composition is being made. One example of a definition supporting this discipline is ISO 10 007 [ISO95] meaning that the major goal within CM is "to document and provide full visibility of the product's present configuration and on the status of achievement of its physical and functional requirements". All CM standards [ANSI98, ISO95, MIL92, SEI00, ISO9000] and most CM books defines CM as consisting of four activities, or areas of responsibility. These are (extracted from the ISO 10 007 standard):

- *Configuration identification*
  Activities comprising determination of the product structure, selection of configuration items, documenting the configuration item's physical and functional characteristics including interfaces and subsequent changes, and allocating identification characters or numbers to the configuration items and their documents.
- *Configuration control*
  Activities comprising the control of changes to a configuration item after formal establishment of its configuration documents.
  Control includes evaluation, coordination, approval or disapproval, and implementation of changes. Implementation of changes includes engineering changes and deviations, and waivers with impact on the configuration.
- *Configuration status accounting*
  Formalized recording and reporting of the established configuration documents, the status of proposed changes and the status of the implementation of approved changes.

> *Status accounting should provide the information on all configurations and all deviations from the specified basic configurations. In this way the tracking of changes compared to the basic configuration is made possible.*
> • **Configuration audit**
> *Examination to determine whether a configuration item conforms to its configuration documents.*
> Functional configuration audit: *a formal evaluation to verify that a configuration item has achieved the performance characteristics and functions defined in its configuration document.*
> Physical configuration audit: *a formal evaluation to verify the conformity between the actual produced configuration item and the configuration according to the configuration documents.*

From a *developer perspective* (tool support), CM maintains the products current components, stores their history, offers a stable development environment and coordinates simultaneous changes in the product. CM includes both the product (configuration) and the working mode (methods) and the goal is to make a group of developers as efficient as possible in their common work with the product. From the developer's point of view, much of this work may be considerably facilitated by the use of suitable tools in the daily work. The definition by Babich [Bab86] stresses the fact that it is often a group of developers that shall together develop and support a system: "Configuration management is the art of identifying, organizing, and controlling modifications to the software being built by a programming team".

Standards are often written on a high level rather than operational. For a developer, on the other hand, it is the operational daily work that matters most. A list of tool aspects we regard to be most relevant, and which we will go into more detail below, are:

- *Version management* - makes it possible to store different versions and variants of a document and to subsequently be able to retrieve and compare them.
- *Configuration selection* - functionality to create or select, associated versions (or branches) of different documents.
- *Concurrent development* - controls simultaneous access by several users, i.e. concurrent development, either by preventing it or by supporting it.
- *Build management* – mechanisms for keeping generated files up to date, for instance during compiling and linking, preferably without generating anything unnecessarily.
- *Release management* - keeps track of which users have which versions
- *Workspace management* - provides a sandbox for each user in which he/she can work in isolation, still within the control of the SCM tool.
- *Change management* – is about both the process of whether or not a change request should be implemented and keeping track of all the change requests and their implementation.

These aspects are expanded on in further detail below. However, firstly we will briefly discuss other CM-related functionality, which may also be relevant for tool support.

Some of the aspects involve a terminology, which is often referred to, and is therefore introduced here:

- Reporting, status – the reporting of a current status with lists of which files have been changed during a certain time period, who made the changes, differences between products etc. These are important functions particularly in the support of the overall view, as seen by the project management.
- Process support – helps/ensures that the developers follow the development model and perform the actions prescribed by it and the CM plan, and that components are progressed through chosen life-cycle phases before being released.
- Accessibility Control (Security) – preventing inappropriate access to information without complicating normal work.

3.2   Version Management

Version management is central to SCM, and is the core functionality in many SCM tools. A lot of developers also, falsely, believe that version control is equal to SCM. Even though it is important SCM is more than versioning as explained in the other sections.

An element of software or hardware placed under version control is designated as a *configuration item (ci)*. The most common example of a configuration item is a source code file, but executables, products, and documents are also configuration items. Also a group of ci's can be defined as a ci, i.e. the group is version controlled itself. The possibility to store, recreate and register the historical development of configuration items is a fundamental characteristic of a SCM system. Every stable issue of a file's content is termed a *version*. The most important property of a version is its immutability, i.e. when a version has been frozen its content can never be modified. Instead new versions have to be created.

Versions of a file may be organized in a number of different ways. When organized in a sequence they are often called *revisions*. They may also be organized as parallel development lines called *branches*. Branches can be merged into a new version, which then has two or more predecessors, see Figure 13.

Revisions are usually deliberately created by a developer, e.g. when a task is completed. In addition, many editors maintain one or several micro-revisions of a file to facilitate its recovery following unsuccessful editing. These 'revisions' are not managed by the SCM tool.

Branches are created for several reasons. The primary ones being *permanent branches*, these adjust the file according to diverging demands for instance different operating or window systems, and *temporary branches,* to permit parallel (concurrent) work. In the latter case, the branches are merged when the reasons for the concurrent work disappear. Usually, a branch consists of a series of revisions and additional branches can be created from the original branches etc. Branches are created for a reason and are there-
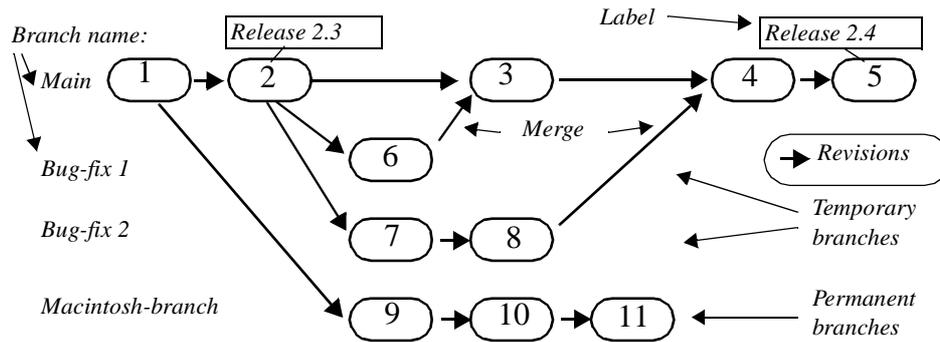
Figure 13   *Basic version control. The versions (1,2,3,4,5) are revisions in the same branch ('Main'). Version 2 has been named "Release 2.3", whereas version 5 has been named 'Release 2.4'. 'Bug-fix 1 and 2' are temporary branches (with one and two revisions respectively) which have been merged back to 'Main'. The example also illustrates a permanent branch with three revisions (9,10,11).*

fore not considered to be equal but to play different roles, for example, as the main line in the development process or as a branch for the implementation of a change, a bug-fix. To create strategies for creating and merging branches is often an important task for a CM manager.

A tool for version control can internally identify revisions, usually utilizing a numbering technique in several stages that may be user friendly to a greater or lesser degree. In addition, the user themselves can usually give the revision one (or several) optional names in the form of a string, often called a 'tag' or a 'label'. The tool can return a version identified by such a string. This facility ("*tagging*") can be used to realize a simple selection mechanism.

3.2.1   Variants

To manage variants is a very hard, and not entirely solved, problem. A common misunderstanding is to draw a parallel between a variant and a branch. We will now try to explain the difference. When whole products or configurations are adjusted according to diverging demands, this is managed with *variants*. For instance, different variants of a product may be developed for different operating systems or with different customer adaptations. The creation and maintenance of these variants can be done in, at least, four ways:

- with permanent branches of the included files. For a variant, file versions are primarily selected from the permanent branch created for the purpose. Secondarily, a file version from a variant independent branch, e.g. Main, is selected.
- with conditional compilation (compiling directives). This means that all variants are managed in the same version of the file and are therefore easier to keep together. However the variant management will not be visible at the CM level.
- with installation descriptions clarifying which functionality should be included in a certain variant. Variant dependent functionalities are implemented in different files, one for each variant.
- run-time check

Thus, to create branches is only one way to implement variants. The most important is not which implementation technique to use, but to manage the many variants resulting from the combinatorical explosion of several optional parameters. Read more about variants in [Tic94].

3.3   Configuration Selection

As shown above, there are often a great number of file versions, and which one should be used in a given situation is not always obvious. The situation is further complicated by the fact that a system consists of a large number of files such that the possible number of combinations is enormous. In a development situation, for files being worked on and for which a special temporary branch has been created, one usually wishes to have the latest revision in that branch. For other files, one typically wants an older, stable version, for instance one that is included in the latest release, or the most recently published stable version as developed by other groups. For files developed within ones own group there is a great need for flexibility, such that it is possible to control how close one wants to be to the others development. Several change requests require the modification of more than one file. In all situations, it is desirable to ensure that there is a consistent selection and configuration, in terms of the inclusion of versions with connected modifications.

A useful technique for the specification of a configuration supported by several systems, is to offer a rule-based selection mechanism. Typical examples of rules that one would like to be able to specify are:

- the latest revision in my own branch (for files that I myself /the group work with),
- the latest revision in a named temporary branch (for files that other groups work with),
- the latest revision in a named permanent branch (for files that differ depending on the product),
- fixed, named, version, e.g. the latest release (for other sub-systems).

A system being built using a rule specifying the "latest" is called a *partially bound* (sometimes "generic") *configuration*, as the exact versions that are included, will vary in time. A system being built without such a rule is called a *bound configuration* and is particularly suitable for deliveries, as the versions of all files included are fixed and therefore it can be guaranteed that the system can be recreated.

A certain bound configuration can form a *baseline*, i.e. is a basis for further development with formal change management, or a *release*, i.e. is delivered to an internal or external customer.

In the same way that the development of individual files can be considered to be a version history, so can a corresponding development of configurations. As an example, the user/customer sees the development of a system in large steps, namely the configurations, releases, that are distributed. Developers and project managers see many more

stages in the development of the system and also the division into sub-systems and configurations, with their own version histories. Therefore the perspective where a system and sub-system are regarded as the development of configurations in bound configurations may be useful at several levels.

The facility of naming versions ("*tagging*") can be used to manage the selection of bound configurations in that all files are tagged with the same name, e.g. "Release 2.3". Relations between such configurations, e.g. that "Release 2.3" is a sucessor to "Release 2.2", is rarely supported by the tool but have to be managed in a different manner, for instance in a release document.

Consistent naming may also be used to represent *logical changes*, i.e. changes arising from a *change request* and result in the modification of *several files*.

3.4   Concurrent Development

One major advantage of using a SCM system is that it enables teams to work in parallel, which is good for many reasons. It can be developers working concurrently on the same files fixing different bugs, or it can be a developer working on the latest release while another is fixing a bug in an old release. It also means that a test team can test the latest stable version at the same time as the development team work on the latest (unstable) versions. The SCM system enables all these situations by providing: (1) selection of versions building specific configurations for different needs, and (2) a model for synchronization of concurrent changes, e.g. by locking the files edited or by always allowing changes to be made but instead detect conflicts at check-in and then merge (often called optimistic check-out). For a more detailed description of synchronization models and their correlation to different situations of distributed development, such as checkout/check-in, long transactions, and change-sets, we refer to [Ask99b].

3.4.1   Distributed Development
It is sometimes the case that developers are geographically dispersed, although working on the same system, a situation we call distributed development or remote development. To better support this situation many SCM tools provide replicated repositories. In most of the tools supporting replication there is no global master repository, but all replicas are copies of the same repository automatically kept synchronized. When one replica is modified by clients at that site these updates are also sent to the other replicas (in batches in a predefined frequency).I.e. when data is replicated between different servers for the first time all data in the repository has to be sent/copied. Data sizes could be as large as several GB. Next time synchronization/replication is done, only update packages are sent with a typical size of 4-5 MB.

The implementation must be so that no conflicts can occur and the synchronization always can be made totally automatically. In ClearCase [RationalCC], for example, they have a site ownership on each branch. Only the site holding the ownership can create versions on that branch. In this way it is always possible to send new versions created on a branch and 'install' them on the other sites without any conflict. Versions on
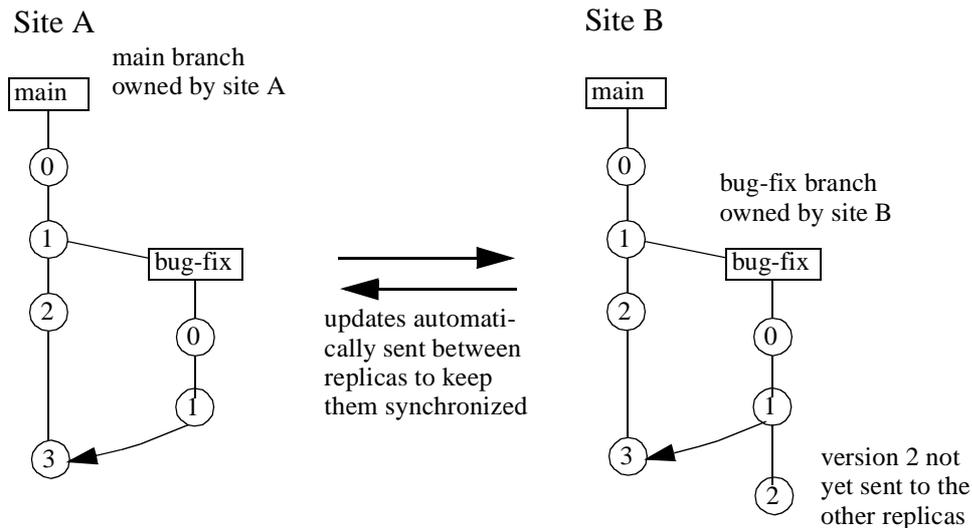
Figure 14   *Replication of repositories*

branches not owned can still be viewed and used to merge from to a branch owned by the site.

3.5   Build Management

Build management supports the user by collecting source code for a particular release and then using build tools, such as Make to automatically create configurations. Make describes the dependencies between source code files at build-time and ensures that the dependent source code is built in the correct order.

Since building large systems may take days, and an inefficient build process can waste hours of developer time, it is important to reuse as much as possible of components not changed since last build. This is particularly important during test and integration, when you need to build the whole system to test a small change. An intelligent build process can reduce build time dramatically by re-using partially built items from previous builds.

Many SCM tools have further developed the ideas from Make [Fel79]. The build proce-dure is automatically created by the tool and often stored in a 'project' file managed by the development environment.

3.6   Release Management

The identification and organization of all deliverables (documents, executables, librar-ies, etc.) incorporated in a product release is designated release management. Release support makes it possible to track which users have which versions of which compo-nents and, therefore, to be sure which of those will be affected by a particular change.

It is possible with appropriate release management to create installation kits automatically to ease the task of the build manager. The build manager is responsible for providing the packed product with the correct configuration and features. Products such as Windows installer [Microsoft] and Install shield can be used to create installation kits. Hoek et al [Hoek97] describes a prototype, designated Software Release Manager (SRM), which supports both developers and users in the software release management process. SRM has the notion of components and helps in assembling them into systems. Dependencies are explicitly recorded so that users can understand and investigate them.

3.7   Workspace Management

Introducing SCM in an organization is cumbersome without effective support from tools. Changing an existing culture requires massive education, support and, above all, motivation. To motivate developers to use all the tools and methods available with SCM, support for integrated tools in the development environment is needed. Workspace management makes it possible for developers to work transparent with the configuration management. When developers are totally focused on solving particular problems and have less interest in administrative tasks, a workspace works as a sandbox in which they can work in isolation, still within the control of the SCM tool. Versions of files are checked-out and put in the workspace, still with a mapping between the versioned objects in the repository and the user files and directories in the workspace.

Not only files modified are checked-out to the workspace. Often all files needed to build and test the product, or part of the product, are checked-out (possibly, some of them read only). Thus the workspace also makes it possible to maintain a certain degree of quality on the files checked in to the common repository, e.g. that all files changes due to the same change request actually works together.

When several developers are working concurrently in their private workspaces, control is needed between the different copies of the same object as described in section 3.4, "Concurrent Development".

Some tools also support cooperative versioning as described in [EFM98]. In short, this means that local versioning within the workspace is provided. When a file is check-in to the repository again, only the latest local version is check-in. The other, intermediate, versions are removed.

An example of integrated features is when the developer "logs-in" to a project environment in which project structures and data repositories are already prepared for the developer (e.g. by the CM group). The developer then enters a transparent environment in which the development is done with configuration management handled behind the scenes. This approach is supported in such major software configuration management tools available on the market today as Clear Case and Continuus [RationalCC] [Continuus].

3.8   Change Management

The reasons for changes are multiple and complex. Changes can originate from many different sources. Change management handles all changes in a system. The reason for a change can be an error, improvement of the component or added functionality. Change management is often supported by separate tools integrated to the main SCM tool. Examples of such tools are PVCS tracker [Merant], Visual Intercept [Elsitech] and Clear Quest [RationalCQ].

Change management has two main goals to achieve: (1) provide a process in which change requests are prioritized and decisions to implement or reject a them are made, and (2) to make it possible to list all active and implemented requests and to track all the changes really made to implement them.

Change management process
When a change is initiated, a *change request (CR)* is created to track the change until it is resolved and closed. Figure 15 depicts how a change proposal creates a change request as defined in [ISO95]. The configuration control board (CCB) analyses the change request and decides which action is to be taken. If the change is approved, the change request is filed to the developer responsible for implementing the change. When the developer has performed the change its status becomes "implemented" and a test is performed. The CCB also decides which changes are to be included when a new release is to be built, and the customer receives a patch including documentation of all the changes made. The latter is also part of release management.

Traceability
Change management includes tools and processes which support the organization and track the changes from the origin to the actual source code [Crn97]. For each CR it should be possible to see which versions of the modified files were created due to that
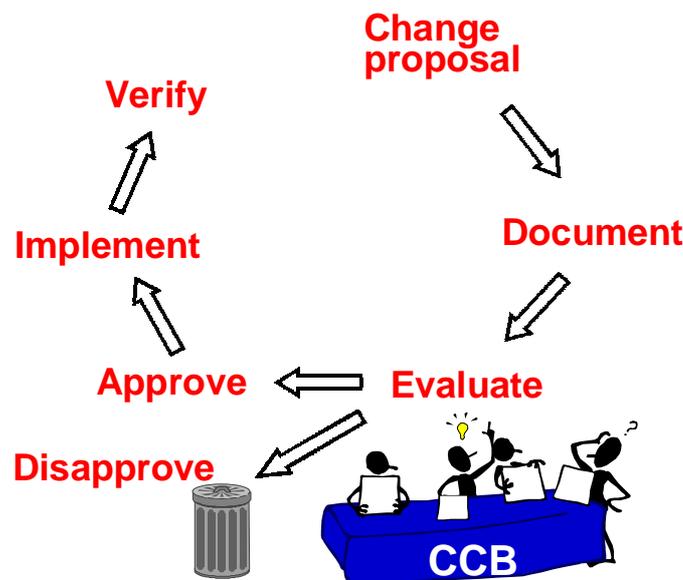
Figure 15   *An example of a change request process*

request. The other way around, it should also be possible to answer the question, "for what reason (which CR) is this version of this file created".

Various tools are used to collect data during the process of tracking a change request. Change management data can be used to provide valuable metrics about the progress of project execution [Crn00]. From this data it can be seen which changes have been introduced between two releases. It is also possible to check the response time between the initiation of the change request and its implementation and acceptance.

4 Comparison of Principles and Key functionalities

Chapter 2 and 3 described PDM and SCM respectively. In this chapter we focus on a comparison of the two, both on a principal level and on specific functionality.

Companies use various PDM and SCM methods and tools. Many people believe it is easy to compare a PDM and a SCM tool, but it is not. Similar terminology but different meaning, different scope of functionality, and different cultures are some examples that make a comparison difficult.

In this chapter we start with a comparison of some principles, i.e. system architecture, product model, evolution model, and process model. We do not describe the functionality of PDM and SCM in detail, but concentrate on the comparison between them. A more coherent and detailed description of the functionality can be found in chapter 2 and chapter 3 respectively.

This chapter shows some of the similarities and differences between four different aspects. In [EFM98] Estublier concludes that both PDM and SCM domains appear to be very similar, but only on the principle level, while the implementations are very different. To analyze the similarities and the differences between domains, we categorize them as follows:

- System architecture (database, distributed development)
- Product model (data model, configuration)
- Evolution model (versioning)
- Process model

We also compare some key functionalities which are:

- Version management
- Product structure management
- Build management
- Change management
- Release management
- Workflow and process management
- Document management
- Concurrent development
- Configuration/selection management
- Workspace management

There is an overlap between the comparison of the principles and the comparison of the key functionalities. In the comparison of principles we discuss the differences and similarities in a more conceptual way. In the comparison on functionalities we compare tool functionality.

4.1   Comparison of Principles

4.1.1   System Architecture

System architecture describes how a system is built-up, e.g. its client-server architecture, and infrastructure. For both PDM and SCM tools the server contains a database (based on the data model) and the server application. The data representation relies on how the data model is realized, see Data Representation. In the client you will find the user interface and some application functionality. The infrastructure defines the client-server and server-server communication. A well defined and planned infrastructure is important if you have a distributed development.

Data Representation

The data representation in PDM and SCM tools is fundamentally different. PDM uses an object oriented data model where data is represented via a business item connected to a data item, i.e. PDM makes a difference between business items and one or more data items (files). A business item is an entity in a database with metadata describing it, while a data item is the actual file. A business item makes it possible to manage the metadata of a file separately stored in the PDM database. The data items are often stored separately in one or several file servers, and not in the database of the PDM system. Hence, PDM separates metadata and data to be able to manage heterogeneous data (different file types, objects, etc.). In a PDM system hundreds of GB of data can be stored and managed. When distributing (replicating) the data not all of it has to be replicated. See "Distributed Development" on page 43 for further information.

The data representation in SCM is more or less a file system with directories and files. Anything represented as a file or directory may be managed and stored in SCM (all kind of file types and objects). The metadata is stored together with the file itself and not in a separate database. Also SCM systems manage hundreds of GB.

The data representation reflect the history of SCM and PDM. SCM manages source files, which therefore, together with directories, are their primary data model. It is possible to store, compare and merge this data within the SCM tool. The need for managing metadata has been realized later on and is implemented as an 'add-on', using attributes. The user interface to manage attributes is, however, in most cases rudimentary and rarely used by the developers (end users).

For PDM it is the other way around. Metadata is the most important. This data is stored in a well (user) defined data model which can be searched through, looked at through different views, etc. The data items themselves, however, are treated as atomic objects, often stored by some other tool and just referred to from the business item.
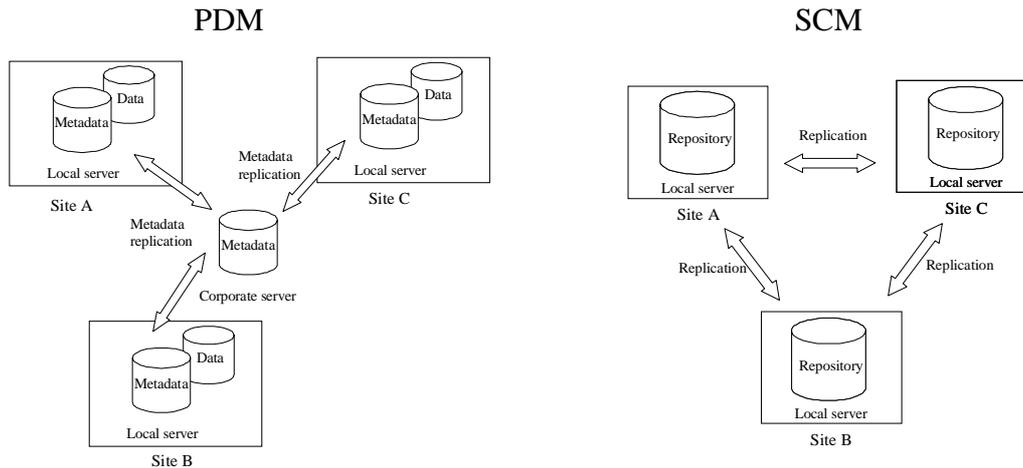
Figure 16   *Server replication in a typical PDM and SCM tool respectively.*

Distributed Development

Both PDM and SCM support distributed development. There are, however, differences, as illustrated in Figure 16. In a typical PDM tool the corporate server is the master server. It contains common information about access rights for other servers, location of the other servers in the network, etc. SCM tools usually do not have a master-slave architecture but more like a peer-to-peer communication between servers.

PDM tools support distributed replicated databases where it is possible to replicate either the metadata or both the metadata and data. The most common is to replicate the metadata but to store all data items un-replicated at their respective servers (one data item exists in one server only). Assuming a user has access permissions, he/she can update (check-out/check-in) a file and change the file itself and its metadata, independent of where the file is located in the network. A distributed locking mechanism prevents two users to check-out the same file at the same time.

In SCM it is not possible to separately manage metadata and files. When a replication is done in the SCM system both files and metadata are copied and kept synchronized. A global lock is seldom used, instead most SCM tools implement replication using site ownership of the branches, i.e only one site (the owner) is allowed to create new version on a specific branch. Versions on a branch not owned by a site is read-only with no possibility to do an update (check-out/check-in) without requesting the ownership.

Application Integration

A PDM system is integrated with various applications and builds an information infrastructure where data from applications is gathered and exchanged. Integrations range from simple ones, where the appropriate application is launched when a file is viewed, to tighter ones, where the PDM system retrieves information from the applications, either through APIs or by direct access to application repositories.

The role of a SCM tool in a complete environment is somewhat different. It can be used as a stand-alone tool, or set of tools, but also as a set of offered functions that can be used by other tools. SCM tools are often designed to provide information and data to other applications (e.g. a file is checked out and sent to a compiler to be compiled). Many SCM tools are integrated in other tools. Typical examples are IDE (Integrated Development Environment). For this reason many SCM tools include APIs with basic SCM functions (check in, check out, baseline, etc.). There is also another phenomena - SCM functions are built-in into different tools. For example Microsoft Word contains versioning functionality. All this makes SCM tools easy to encapsulate in other tools.

The historical difference in the way to integrate to other applications affects the possibility to integrate a PDM tool and a SCM tool. While the PDM tool often is the central process that creates activities in other tools, the SCM tool is more passive offering an API. The data format is also different. PDM has standards defining transfer protocols. SCM uses plain files only as the data format.

4.1.2   Product Model

A product model is an information model used to describe the structure (building blocks such as parts) and behavior (outputs, effects, properties etc.) of a product. A PDM system has an explicit product model. The basic principle of product modeling in PDM is the composition relationship, which is used to the form tree structures, often referred to as product structures. The product structure is not hidden in the system, but presented and edited by the user. On the other hand, product modeling in SCM is very weak. SCM systems do not contain a product model. The differences in approach come from fundamental differences in the nature of hardware products: In PDM, the product has a physical existence and consists of physical parts. For that reason the product structure can be represented by a part structure. In software there is no such real structure; parts are arbitrary abstractions with loose relationships. The product structure (application, operating system, platform) exists only in the development phase, where tools such as Make [Fel79] specifies relations (dependencies) between different components. The final product is often a single executable file or a library.

Behind the product model is a data model, which describes the types of objects, relationships and attributes used in it. In a PDM system it is possible to change this data model, which is often done to better fit the particular needs of a company. A set of industry specific data models are included in the STEP standard [STEP91]. The purpose of STEP is to facilitate data transfer between various information systems within and between companies. It is an comprehensive standard and includes among other things descriptions of geometry and product structures. Several PDM systems support data transfer based on this standard, and some do even have a data model based on this standard. In SCM, a customizable data model is only available in a few systems. Most SCM systems structure information by using the file and directory structure used in the operating system. SCM has no standards like STEP.

4.1.3   Evolution Model

The *evolution model* manages changes during the product life cycle, and is related to version management. SCM has more advanced versioning functionalities than PDM, which arises from the differences in the products' natures: Since software may be changed more easily than hardware, SCM must manage versioning in a more sophisticated way (includes branch and merge) than CM for hardware [WC98].

PDM recognizes three different concepts for versioning: historical versioning, logical versioning and domain versioning. Historical versioning is conceptual and similar to SCM versioning, dealing with revisions/versions of a product. Logical versioning manages versions of parts as alternatives, possible substitutes or options. Finally domain versioning is not actually versioning but more the generation of different views of product structures, e.g. as-planned, as-designed, and as-manufactured.

SCM emphasize on historical versioning, including the possibility to create and merge branches and to present the differences between versions. Logical versioning does not exist. The concept of "view" exists in many SCM tools and is related to the flexibility to create configurations by selection of correct versions of the files included in a specific configuration. This is used both to create private workspaces and to build the product.

The version models in PDM and SCM tend to become more alike since the objects managed have become similar. Software artifacts, (i.e. program packages, electronic documentation, software web and other information services, etc.), are being more and more part of PDM management.

4.1.4   Process Model

The process model is conceptually similar for both SCM and PDM. A State Transition Diagram (STD) describes, for a product type, the legal succession of states (and optionally which actions produce the transition), and thus describes the legal way to evolve for entities of that type. The alternative way to model processes is the so-called activity centered modeling, in which the activity plays the central role, and the models express the data and control flow between activities.

PDM systems has two process related concepts: Object states and workflow. The object state defines the life cycle for an object, e.g. preliminary, approved, and freezed. Hence, the object states are described by STDs, but the term is not used in PDM systems. Workflows are based on a description of the process, which includes activities, their sequence and relationships between them. Workflows can be used to control data management activities within or between processes. Hence, a workflow is an activity centered model.

Since SCM aims to control software product evolution, it is no surprise many process models are based on STDs. It is a product-centered modeling, [EFM98, Est00]. As [Est00] concludes, experience shows that complex and fine-grained process models can be define that way. Unfortunately, experience also shows that STDs do not provide a global view of a process, and that large processes are difficult to define using (only)

STDs. The activity based model is not used to the same extent as in PDM, but some SCM tools provide process support similar to the workflows in PDM.

4.2   Comparison of Key Functionalities

This section outlines the differences of the functionality in PDM and SCM. Although both domains in general are very strong on CM, some parts of CM are treated differently. The following areas of functionality are compared:

- Version management
- Product structure management
- Build management
- Change management
- Release management
- Workflow management
- Document management
- Concurrent development
- Configuration/selection management
- Workspace management

There are other functionalities closely related to PDM and SCM not treated here. These are managed by separate tools or modules within the tools. One such example is requirement management tools or modules.

Version Management
The version models are different in PDM and SCM systems. Versions in SCM form a hierarchical structure, in which two versions of a file can be developed simultaneously in branches, and be merged together again if needed. Later, when a baseline is created, it is performed by manually selecting the versions to be included in the configuration. These versions are then marked using a special attribute often called 'tag' or 'label'. In PDM systems, versions of a business item (an object) are denoted revisions. The revision number is a flat structure (only one main branch) compared to the hierarchical one in SCM. If the company uses another versioning set, which is not possible to automatically generate, an attribute called 'revision' is used instead and manually updated. A business item represents and carries metadata for both products and documents. Only major changes of a business item are tracked by revisions. To change a released business item or its associated data items (files), a new revision of the business item must be created. When the business item is approved, its new revision is frozen. If the business item has data items associated with it, it is most often the data item that is changed. While the data item is changed, it may be checked in and out several times. To manage the sequence of data items, versions are used internally in the PDM system.

PDM   **SCM**

Revisions. Visi- **Versions.** Internally   **Attributes.** Rarely   **Versions.** Visible
ble and heavily   used and normally   used and normally   and heavily used.
used.   not visible.   not visible.

Label. Special
attribute used to
define configurations.

Attributes. Heavily
used and visible.

**Version of an
object**

**'Revision-
of' relation**

**Version of a
file**
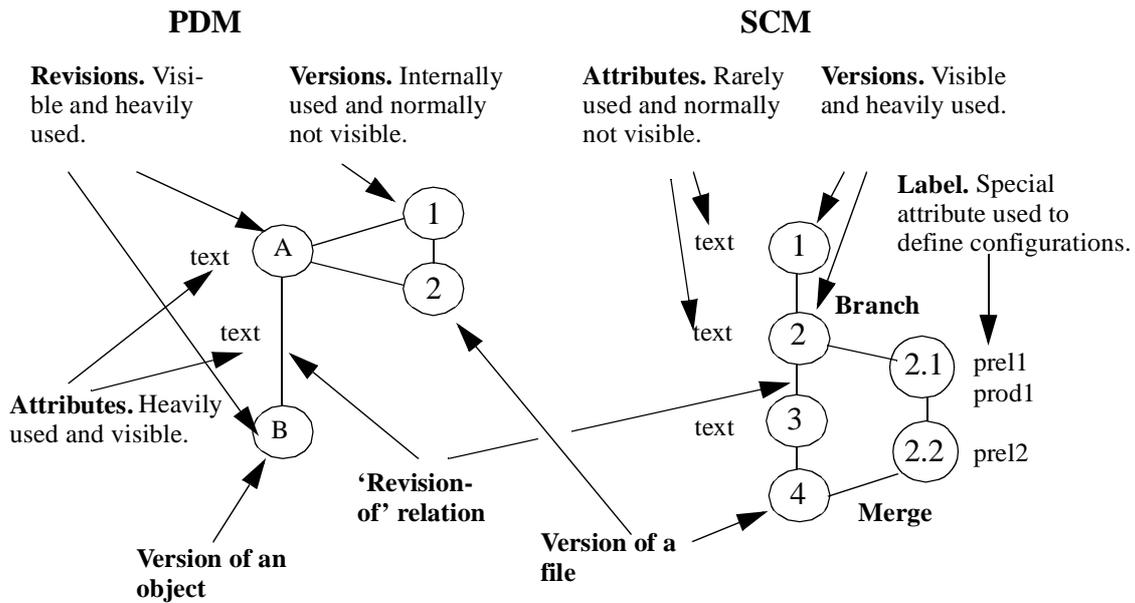
Figure 17   *Version Management in PDM and SCM*

Figure 17 shows the similarities and differences of Version Management in PDM and SCM:

- PDM manages objects, SCM manages files and directories;
- PDM uses revisions for major changes, SCM uses versions for all changes;
- SCM has branches and merges, PDM does not;
- In SCM several people can work on the same file at the same time using the branch facility, which is not possible in PDM;
- Both SCM and PDM tools have attributes. SCM has a special attribute called label which is frequently used. General (user defined) attributes are rarely used in SCM due to bad usability. PDM strongly support customized attributes through their data model.
- PDM has relationships, SCM does not (apart from the 'revision of' relation implementing historical versioning);
- Relationship in PDM may have attributes, SCM has not.

Product Structure Management
A PDM system describes a configuration by arranging the parts in a product structure. SCM tools do not use an explicit product structure. Only rudimentary support in the form of directories in a file system is supported.

Build Management
In SCM build is essential and supported. Build is not supported in PDM.

Change Management
Change management is similar in PDM and SCM. In PDM there are add-on modules which support change management. In SCM there are specific tools integrated with the SCM system.

Release Management
In SCM a simple support for release management (e.g. packaging of the executables, related documents, and installation program) for customer do exist for pure software products. In PDM the support for release management is strong. The package sent to the customer is a component in the product structure with relationships to including artifacts.

Workflow and Process Management
In PDM systems it is possible to define and execute workflows or processes. All processes are possible to change and to adapt for specific projects. Some SCM tools have similar functionality included in the tool or provided by tightly integrated tools within the 'tool suite'. Most SCM systems, however, only provides triggers that can execute scripts written by the users. This is no real support and results in a mess of scripts hard to overview and maintain.

Document Management
PDM has built-in Document Management facilities like query and access control. SCM has not. The trend is that more and more documents are stored within SCM, but this will probably result in problems like lack of query functions, and no access control.

Concurrent Development
Both PDM and SCM provide shared databases/repositories and locking to prevent simultaneous updates. SCM also provides workspace management together with branch and merge, which makes it possible for developers to work concurrently within the same file, please see Figure 17. The support of merge also makes it possible to provide 'optimistic check-out' instead of locking as described in "Concurrent Development" on page 35.

Configuration/Selection Management
In SCM there are usually many versions of the same file. The SCM tool provides support to select the correct version of each file. This is important both when retrieving versions to the workspace and when the product should be built. In PDM there is no functionality compared to the selection function in SCM. In PDM you may have configuration effectivity, which is a time or revision limitation. However, this is not the same thing as selection management.

Workspace Management
In a SCM system the user checks out all the files he/she needs to change. The files are stored in his/her workspace. The SCM system registers all files checked-out; which version, by whom, and in which workspace the copy is stored. If many users check out the same file (and possible the same version), these are coordinated under control of the tool following the synchronization model used. Workspace management in this form do not exist in PDM. In PDM you check out one file at a time and update it.

Role Definitions
While roles are an important part of PDM tools, SCM tools have different approaches. Many SCM tools separate users only by their identity, possibly an administrator role can be defined, but not more. A few SCM tools contain the concept of roles, but this is different from tool to tool.

4.3   Summary and Conclusion

We summarize this chapter by stating advantage and disadvantage for each domain. Then we present a summary of the compared functionality.

We summarize for PDM:

- PDM tools have been on the market for decades.
- They are strong in product modelling.
- They have a long tradition and standardized product evolution control know-how.
- Some tools partly use a standard, some do not.
- They are strong in Workflow and Process management.
- They are strong in Document Management.
- PDM is strong in data representation where metadata and data are separated.
- PDM is strong in the data model where an object oriented data model is used.
- PDM do not support Concurrent Engineering for a single file.
- They do not support Workspace Management.
- They do not support Build Management.
- They do not support Configuration/Selection Management.

We summarize for SCM:

- SCM tools are more recent than PDM tools.
- They are good at managing files and directories.
- They are strong in Version Management.
- They are strong in Build Management.
- They are strong in Concurrent Engineering on a single file.
- They are strong in Configuration/Selection Management.
- They are strong in Workspace Management.
- They are weak in product modelling.
- They are weak in Release Management.
- They are weak in Document Management.
- There is no standard for SCM.

The conclusion of this is:

PDM tools do not have functionality enough for software management during the development phase, but complement SCM tools for managing product related information.

Functionality Summary

Table1 gives a short overview of the functionalities compared above.

Table 1   *Summary of functionality of PDM and SCM*

Type of Functionality	PDM	SCM
Version Management	Yes	Yes, with branch and merge
Product Structure Management	Yes	No
Build Management	No	Yes
Change Management	Yes	Yes
Release Management	Yes	Yes, but weak
Workflow and Process managment	Yes	No
Document Management	Yes	Partly
Concurrent Development	No	Yes
Configuration/ selection Management	No	Yes
Workspace Management	No	Yes
Roles	Yes	Yes, but weak

5   Research and Trends

Research and trends in industry have been different for PDM and SCM. This chapter gives a short overview of history and trends in research and in industry separately for PDM and SCM.

5.1   PDM

5.1.1   History Overview

Product data management is not a new activity. The corporate basic standards for identification of documents and products at Ericsson was defined already in the 30s. Traditionally, text documents and drawings have been archived on paper. It was a cumbersome task to find a document, and it was often difficult to tell whether the document was the most recent issue. As computer technology evolved, more and more product data was created in digital form, such as digital text documents. Digital documents has made it possible to store documents on file servers, thus creating a need for systems managing those files.

The management of design documents were the most common task when commercial PDM systems were introduced. CAD drawings and CAD models are extra important in the manufacturing industry. Many companies started to use PDM to be able to better control their CAD documents. PDM systems were developed both as separate systems and as data management modules in CAD systems.

Besides CAD, PDM has its origin in manufacturing. Most companies were using databases to store product data before digital documents were used. Databases were used already in the 70s to store product structures, which were used by the manufacturing plants to manufacture products. For this reason, companies developed their own database applications. The design departments were responsible for feeding the databases with information. Since design departments and production facilities had separate locations and designers and manufacturing engineers had various demands on the information content and breakdown, the design and manufacturing used separate databases. In the late eighties, commercial PDM systems became available. The PDM systems used a product structure similar to the manufacturing structure to organize the documents. Since the product structure was used in the system, product structure management became an issue for PDM. In the beginning, not many companies used the structure management capabilities of the PDM systems, but used them as advanced file systems for design documents. The in-house developed systems were still used to manage product structures, and still are at some companies. Legacy systems are hard to replace, because they handle critical data, have many users and connections to other systems.

PDM systems are not only used in manufacturing industries, but are used in a variety of industries. The recent trend in PDM is to support the complete product life cycle and to enable collaboration. These trends will be further described in this chapter.

5.1.2   Trends in Research and Development of PDM

The research in the PDM area is different from the one on SCM. In SCM there is a solid theory base and a common and accepted terminology. In PDM on the other hand, the definitions are less precise. The reason for this can be that PDM has a wider scope than SCM. PDM manages product development information across the entire product development and support phases, including integrations with authoring and collaboration environments, while SCM manages the software development environment. It is easier to develop theories and knowledge in a more comprehensive domain. Another fact is that few PDM researchers and developers use PDM systems on a daily basis for their ultimate purpose, i.e. to help design products, while SCM tools has been developed by programmers for programmers.

Research on PDM has a wide scope and emphasizes more areas than only information systems and data management methods. The way data is represented and structured is an important issue. To be able to discuss these matters theories for how to structure the things represented, the products, are needed. This has grown to an own area of research, product models, which deals with how products should be represented by information models and represented in databases. This area and other important areas of research on PDM will be presented in this section. The Sections *Product models* and *Product data representation and modelling* are extracts from a report written by a group of researchers within the Swedish national graduate ENDREA (Engineering Design Research and Education Agenda) [Isa00].

5.1.3   Product Data Management

PDM technology can be seen as a strategic technology supporting product life cycle management (PLM). Collaboration is a key objective for PDM solutions. PDM solutions address business drivers of:

- Time to market,
- Total product quality,
- Product development cost,
- Innovation,
- Globalization, and
- Leverage intellectual property.

These business drivers are addressed with PDM strategy components such as single source of information, life cycle support, knowledge management, system integration, collaboration support and support for distributed work. Harris [Har96] states that there has been little research done on the connection between PDM systems strategy and a company's over all business strategy, either directly or through a higher level information systems strategy.

It has earlier been observed that PDM systems are mostly used in design [Abr97]. The integrational capabilities of the PDM systems are not yet fully exploited. The main challenge in future research in this area is to merge all local solutions used in the different product life cycle phases into a system supporting the whole life cycle. PDM tech-

nology has a key role in such a system. The commercial systems available today addresses support for various life cycle phases better than before, by offering a set of applications supporting various life cycle phases.

Several processes in the product life cycle are affected by a PDM implementation. Pikosz *et al.* [Pik96] discuss different strategies for introduction of PDM systems. PDM can be introduced by starting with one functionality implemented company wide, across several projects, or as a complete PDM system introduced in a project. Risks were observed with the full introduction. These risks are nowadays met with a phased introduction, were the functionalities are implemented step-wise.

PDM technology has evolved from document management to collaborative product commerce or collaborative product management, and now supports the evolution of information management to the extended enterprise. The integration paradigm has changed. J2EE (Java 2 Platform, Enterprise Edition) architecture has enabled connection to legacy corporate systems through Java application servers utilizing Java connectors and reactors to transfer information or provide aggregate views of information stored in different systems. In this way, single user views of product information can be aggregated, such as product definition that consists of configuration information from PDM, cost and quantity information from ERP and supplier information from CSM (component and supplier management).

The introduction of a PDM system is not only a question of technology. Good knowledge of company specific processes and the requirements they put on the PDM systems is needed. Research is focussing on the technical aspects of product data management, such as computer systems and data representation. Questions related to the product data management processes have not gained much interest yet. More research should be performed on how to introduce PDM systems.

5.1.4   Collaborative Product Commerce

PDM has traditionally been associated with design, while the vision of PDM has been to cover the management of product data throughout the complete life cycle, from the early phases until manufacturing and maintenance. Now it seems like we are almost there. Commercial PDM systems on the market announce their tools now support Collaborative Product Commerce (CPC) or Collaborative Product Management (CPM). This development addresses the needs in industry. Companies today are globalized, focus their supply chains, extended enterprises, strategic partnerships and virtual teams [Mil01]. This calls for collaborative tools that can bridge geographical distances and allows people to work in virtual teams. Example of processes supported by collaborative tools are [Mil01]:

Change Management and design review: To review the design of a product developed at several sites can involve sending information several times between the sites. If the information is stored at a single location and the process is controlled by a workflow, less information is sent between the sites and control is gained over the process. The next step is to add synchronous collaboration and allow virtual team members to work

together simultaneously. This is already possible for CAD models, but can be used for other types of documents too. A group of designers, working at separate locations, can view the same model, annotate it and communicate with each other using text messages, audio and even video communication.

Sales and bidding: To make it easier for your customers, you can let them get access to your product information. The request for quotation process can be automated. The customer sends in a request, it is checked if it is valid and it can then be followed on its way through design and manufacturing.

Maintenance and support: Engineers, operators and maintenance staff need access to up-to date maintenance information. To make it easier to understand the documentation, it is possible to use animations to illustrate operations. The feedback is also important; personnel on the field can send in reports and suggestions.

Manufacturing planning: Manufacturing personnel can use collaboration for many purposes including reviewing design and change orders with the design team, interfacing early with tool designers, verifying tooling assembly and operation, reviewing manufacturing process plans and factory layouts, discussing manufacturing problems with suppliers and coordinating tooling among dispersed sites.

5.1.5   Product Structure Management

Product structure management (PSM) has a key role in product data management in the manufacturing industry. The trend in product structure management is toward life cycle BOM management capability with functional, physical, planning, as built and support configurations that support field update to customer configurations, total product life cycle management.

The research on product structure management is strong on the subject of product structures used to support manufacturing. The use of bill-of-materials (BOM) has been quite extensively covered, [Jan93], [Sve00], [Sch93], [Heg95]. From a PDM perspective, the most challenging problem is to handle variants of products, see Section *Product configurators*.

Different disciplines in a company have different demands on how the product structure should be decomposed. In design, a decomposition into systems and subsystems is likely to be preferred, while manufacturing uses an assembly decomposition of a structure [Jan93]. The conflicting demands lead to that several structures are used in a company, stored in separate information systems. Strategies for cooperation between systems used in the PSM process is discussed by Svensson and Malmqvist [Sve00].

5.1.6   Configuration Management

CM is an activity that originates from military and space applications in the late 50's, where complex products are built from detailed specifications. Traditional CM is on a coarse grain document management. PDM enables product centric, fine grain configuration management (EAI-649) where configuration identification and control is directly

on product level, but many CM driven implementations of PDM create not much more than better management of the standard CM documents.

An important part of configuration management is change management. Even if a company does not perform configuration management by the standards, companies often need to control the changes of their products. PDM technology offers fine grain change management and allows change to be effected at the finest object/ attribute level. Impact can quickly be identified through relationships minimizing the over all disruption of change.

Wright [Wri97] has performed a thorough review of research into change management. One of the questions raised here is how engineering changes (EC) drives the incremental and step-wise design in different types of companies. While engineering changes can be seen as evil from a manufacturing perspective, they can actually have a positive impact on the product design process. A comparative study of change management has been done by Pikosz and Malmqvist [Pik98], to find company specific factors for how change management is performed.

5.1.7   Product Configurators

Product configurators are used to put together a product, which offers choices between alternative components. A configurator contains a set of rules for selecting the correct components. Configurators are used by salesmen to automatically configure a product based on a customers requirements and in manufacturing to collect the correct parts to build a specific order.

A sales configurator tries to solve a complex problem. An advanced sales configurator is able to optimize a product based on several customer requirements, that may be in conflict with each other. In the eighties, artificial intelligence was a popular approach for solving the configuration problem. Nowadays, the approach is based on knowledge management. To configure a product, it is essential to be able to handle both the process of configuring a product and the data and knowledge associated with it [Sch93]. Product configurators are often in-house developed software and are difficult to update with new products and rules. Mesihovic and Malmqvist [Mes00] propose that the information needed for configuring could be handled by the PDM system during the development phase. Configuration data would then be available to all who need it and it would be easier to change the data.

In manufacturing, the problem is associated with production control issues. The BOM must be able to handle variance. A generic BOM is a BOM capable of handling several different variants of a product in a single structure. It is difficult to plan a production system if you do not know exactly what the product produced in it will look like. The production planning system must be able to work on aggregate information [Heg95].

5.1.8   Document Management

Document management is an old profession, but still under development. Until now, most information has been handled as it were written on paper. Standards for mark-up

and structuring of documents, such as SGML and XML, may change this situation. Mark-up languages can for instance be used to mark up which parts of a document that describe which parts of a product. Depending on which parts that build up the delivered product, the correct documentation can then be automatically generated [Gra98]. This will create new requirements for document management. Information can no longer be managed document by document, but rather as information elements.

5.1.9   **Product Models** *(Extract from [Isa00])*

Until relatively recently, an engineer's view of product modeling has been a purely technical one; concerned with the problems of how to create a robust geometric model in a CAD or CAE system. However, as computer aided design and engineering environments have improved and data management systems allowed this data to be more easily shared, new problems have emerged.

In order to generate a sound product model, it is important to have a clear idea of how to decompose a product. The science of engineering design has created elegant theories, such as the Theory of Technical Systems (TTS) [Hub88], describing how product models should be structured, but these are hardly, if ever, used in industry. More widely implemented techniques used to organize and structure information is Systems Engineering (SE) [Bla98], [INCOSE]. It was initially created to support development of complex systems, while the TTS is the result of European research into engineering design. Sharing some features, SE is more concerned with analysis and life cycle management and has been formalized in at least two standards, IEEE 1220 and EIA 632, while the European Design research has a more product oriented view, decomposing specification into a component structure using a, so called, chromosome model of a product [And92].

These techniques (TTS and SE) are concerned with high level structuring of product data. At a more detailed level, interest in techniques to formalize engineering know-how, often referred to as knowledge based engineering (KBE), is increasing. KBE has been around for many years; one of the best known systems being iCAD which dates from the mid 1980's.

It is clear that the need for a systematic way to describe products is increasing in importance since product models now include complex constructs such as rules, variants, requirements and product configuration possibilities.

5.1.10   **Product Data Representation and Modeling** *(Extract from [Isa00])*

Product modeling theory only deals with how to structure the product, but not how to represent it. For this we need some kind of language. The capabilities of representation have evolved with the development of object oriented techniques. The capabilities progressed from lists of values defined by text documents to product models that also take care of the semantics and internal relations within the model. The modeling language that describes the product model must therefore also be able to define these semantics. It is important that both humans and computers can interpret the modeling language, so misunderstandings and misuse can be avoided.

There are a number of languages capable of handling product information together with semantics. The two languages used most today are the lexical language EXPRESS [Sch94] and the visual language UML [Eri97]. Both these languages are defined by international standards and are not based on any specific implementation technique or specific programming language.

The advantages of EXPRESS, compared with UML, are its capacity to handle rules and the fact that it specifically was designed to describe product information. UML was originally designed to describe software development projects, but the general structure of the language also makes it suitable also for describing product information. UML also has the capability to describe processes and display different views of the product information.

5.1.11 Databases

An information management system often uses a database to store data. Databases are used to store data in a structured way. A database also offers a query language, that is used for instance to make queries to the database. A database can also handle multiple users, security and backup of data. Most databases used today are relational or object relational databases. Object oriented databases are a promising technology, but has not yet had a breakthrough, mainly due to performance problems.

Database technology has existed since the early 60's [Elm89] and is a mature area both from a practical point of view and a scientific point of view. However, handling product data in a database is different from many other applications:

- Large hierarchical structures need to be stored.
- The data models differs between companies.
- Many different types of objects and attributes are needed to describe a product.
- The frequency of creation and change of data is high.

5.1.12 Trends in Industry

To describe the status of PDM today and the problems companies experience, the situation in two larger companies will be described: Ford Motor Company and Boeing. The description is based on a report from a study visit at those companies by a group of researchers within the Swedish national graduate ENDREA (Engineering Design Research and Education Agenda) [Fux00].

Ford Motor Company

Ford is a company in the automotive industry. The PDM activities at Ford is centered around a project called C3P. Ford acts globally and has a wide variety of brands: Volvo, Mazda, Lincoln, Ford, Mercury, Jaguar, Aston Martin and Land Rover. About 60 car programs are developed in parallel. Ford uses an in house developed system to manage product data for many years, but finally decided to develop a new computer supported environment for product development. Partners and suppliers should also use this environment. One year later SDRC was chosen as the provider of CAD and PDM systems,

IDEAS and Metaphase respectively. Now, four years later, all new car programs are developed in C3P and the first cars developed in the CP3 environment are introduced on the market.

The Goals of C3P
The vision of C3P is "Integration of Ford Enterprise Product Development through Global PIM and Advanced CAD, CAM and CAE functionality". The driving forces pointed out to be behind this investment are the demands of concurrent systems engineering and supply chain integration. The measurable targets of C3P were ambitious:

- Lead-time reduction from 50 to 36 months,
- Prototype cost reduced by 50%,
- Re-use of components increased by 30%, and
- Late product changes reduced by 50%.

The TTM (time-to-market) goals of C3P are now achieved at both Ford and Mazda.

The C3P Concept
C3P involves more than information technology. Both the tools and the way things are done has to be changed. To change the working procedures is the big challenge and a large part of the introduction was training of people at Ford and it's partners and suppliers. The concept is divided in four CAE views and PIM (Product Information Management). Digital mock-up (at Ford referred to as Digital Buck) represents the product definition, digital factory the production process, digital clay the styling surfaces and analytical prototypes are simulation models. Digital mock-up is the most mature area. A digital mock-up is a simplified representation of a CAD geometry and is used to represent large assemblies, which could not be displayed in their native format du to performance problems. Ford uses it to study how parts fit together, by project managers to follow-up projects and to support decision making at meetings.

PIM manages the information, which is done with Metaphase. The PIM installation is based on a tight integration with the CAD tool IDEAS and a distributed environment. A team of designers (about 50) work in their CAD tools, which are connected to a TDM, Team Data Manager. TDM is a mini-PDM system included with IDEAS. The CAD models in the TDM are grouped in master packages. These are copied to a central vault, to which other teams have access. The central vault is connected with the digital mock-up program. This way it is possible to visualize and analyze the complete vehicle, though it has been developed in geographically distributed teams. The product structure is not explicitly stored in Metaphase, only a list of the master packages representing the product. Release of parts is not supported in Metaphase, but has to be done in an other system.

Problem Areas
PIM needs further development. The information infrastructure is causing problems. The connections with eastern Europe (based on telephone lines) were too slow. Other performance problems were found in the TDMs, when large teams of designers worked

with large assemblies. The integration with legacy systems, for instance for release, also have to be improved.

Boeing

Boeing is in the aerospace industry. Boeing makes most of the design in house. Boeing has chosen to work towards standardization of information instead of systems. Boeing has decided STEP should be corporate standard for Product Data Exchange. Therefore Boeing will only consider using products that complies with the STEP standard. The partners Boeing work with must be able to use STEP to exchange data.

Boeing has to deal with a situation were several CAD systems are used. Take an engine supplier as an example. It supplies engines to other customers than Boeing. It is too costly for the engine supplier to make the design with the CAD tool each customer demands, since it is a very complex design. Boeing therefore has to accept suppliers use various CAD tools. To verify the complete product with fuselage, wings, engine etc., a digital mock-up is used. The mock-up is based on a neutral geometry format included in the STEP standard, AP 203.

Boeing uses several PDM systems: Metaphase, IMAN, Sherpa and Windchill among others. They do not disagree with the idea of having a single PDM system, but only if it fulfills Boeing's demands. No system available on the market is considered to do this. The usage of STEP will make it easier to change PDM system.

Boeing has tested in a project to exchange geometry, parts lists and information about changes with STEP. The information is exchanged between engineering and manufacturing data bases both within Boeing, but also between Boeing and it's partners.

Boeing commercial airplanes is improving the way it builds airplanes by implementing two critical initiatives, Lean Enterprise and Define and Control Airplane Configuration/ Manufacturing Resource Management (DCAC/MRM) [Boeing]. DCAC/MRM focuses on streamlining the way commercial airplanes are designed and built. Both activities will help Boeing meet its internal goals to reduce costs, cycle time and defects and deliver more value to its customers. The system support consists of a single PDM system (Metaphase), an ERP system, a product configurator, and a process planning tool among others. Some facts of the implementation:

- A total of 40.000 users, located in eight states, Canada and Australia
- Almost 1.000.000 part numbers in database
- 20 production databases ranging from 10 to 130 gigabytes in size
- Estimated Boeing savings: $100M/month

Problem Areas and Future Work
A common objection to the use of neutral formats is that they do not keep up with the progress of the tools. It can take up to six years before a new functionality in a CAD

tool is introduced in the standard. There is a trade off between neutral formats and high functionality of the tools.

A time plan has been formulated, which outlines the introduction of STEP in various areas. Boeing's vision is that all product data should be stored in open formats. The DCAC/MRM initiative is planned to increase it's number of users further to 50.000 users.

5.1.13   Conclusions

Many companies today have realized the strategic importance of a PDM system implementation. PDM investments (software and services) in the industry continuously grow, from $1.1 billion in 1997 to $1.7 billion in 1999 [Cim99]. The implementations have often been associated with problems and large costs. Services account for a greater part of the implementation cost. However during the last years more successful implementations have been achieved. That can be explained with software and hardware development and the growing PDM interest in industry that results in a stronger commitment to the PDM projects. Even if there is an strategic importance in PDM systems, PDM projects today have larger requirements on them when it comes to return of investments and better calculations of cost estimates and cost savings are required.

The trends in research together with the material from Ford and Boeing point out some trends and common characteristics of PDM today.

- Commercial PDM systems have proven their capabilities in large scale implementations.
- Time and cost reduction are important drivers behind every PDM implementation.
- PDM is evolving to PLM and Collaborative Product Commerce, with greater emphasis on information evolution and life cycle management, and global access, control, collaboration.
- PDM is successfully used to support distributed development.
- Configuration management moves to fine-grained product centric control.
- Integrations between PDM and SCM systems exist.
- Digital mock-ups enables faster verification of products. To build a physical prototype (a physical model of the product as it will be manufactured) is time consuming. It is now possible to verify that the components fit together directly in the computer, which allows managers and project managers to instantly get the status of all geometries of a project. This is somewhat similar to the daily build in software development, and requires more frequent check-in of new versions, for the mock-up to reflect the actual status of a project.
- Product models play an important role in PDM. Compatibility between information models is a requirement to be able to exchange data.
- Two main strategies to exchange data between systems have been observed: Require partners use the same systems as your company does, like in the Ford example, or to use neutral formats, like Boeing is using STEP.

5.2   SCM

5.2.1   History Overview

The history of SCM development follows the software development. In 60s when software consists of monolith programs mostly implemented as one source module, there was no need for SCM. The software engineering community was focused on the programming art - producing as good as possible algorithms occupying as smallest size of memory as possible.

In the 70s and 80s software became more complex, the programs were built in two stages, compilation (producing binary modules from the source code) and linking (putting together binary modules in a program or a binary library). In this period the first versions of SCM tools appeared: Make [Fel79] and SCCS (Source Code Control System), covering the basic SCM disciplines: configuration and build management, and version management. Later an improved version of SCCS, RCS (Revision Control System) [Tic89] replaced SCCS. At this time SCM was focused on programming in the large (versioning, rebuilding, composition). It is interesting that these tools have dominated on the SCM market during many years and the principles that they were introduced are widely used in majority of modern tools. Make exists today in many forms (such as imake, gnumake, different "project-files", etc.) and RCS is used as a base tool by many other, more advanced SCM tools.

During the 90s the focus of SCM has moved to the "programming in the many", i.e. emphasis on teamwork (process support, parallel development, concurrent engineering). Change management, workspace management and process support became the new issues of SCM tools. As the complexity of software increased, the complexity of SCM tools grew as well. In that period several new advanced SCM tools appeared, very complex and very expensive. In addition to their complexity they were difficult to adjust into the development process. For this reason many companies developed their own systems, or used simple SCM tools, executing other SCM procedures manually.

In the 90s the popularity of SCM has significantly increased. There are several reasons for that: The software development became more complex, and a need for tools managing this complexity became obvious. At the same time the software development became more important in business and many new software companies appeared on the market. Finally, the software development focused on development processes, with CMM (Capability Maturity Model) [SEI95] in front. The process models emphasized the role of SCM, and defined the SCM process. An SCM process includes a number of activities and their planning (with emphasize on the planning) during the entire product life cycle.

As the software industry grows amazingly today, the need for managing software also grows. The new focus of the software development turned to programming in the wide (web remote engineering, distributed development). Although the problem domain is closely related to SCM, it is not clear if the problems will be solved by SCM community or by other domains.

5.2.2   Trends in Research - an Overview

During the entire SCM "evolution" period the SCM community has tightly collaborated with industry, or even was involved in both research and product development. Some of the examples of products, being developed in such way are, Adele, Odin, DSEE and ClearCase, Continous, etc.

The leading SCM researches have gathered around Software Configuration Management Symposia, starting 1988 (SCM-1), having the latest symposia in 1999 (SCM-9) and 2001 (SCM-10).

These symposia were characterized by close cooperation between researchers and SCM tools providers. Even SCM customers, both large and small companies, participated in the symposia, so usually an "Industrial Experience" session was a part of the symposia. Swedish industry and even Nordic-country industry have been well presented. One conclusion from the discussions at the last SCM symposium was that considerable classes of research-problems in 90s were solved and the solutions gathered from the vendors. New classes of the problems will come in the next decade. For this reason the SCM-10 symposium was much more research-oriented.

The main topics in the recent years comprised the problems the researches discussed and vendors tried to solve:

- Versioning Models
- Workspace Management
- Distributed Configuration Management
- Component Configuration Management
- Process Aspects
- SCM on the Web
- PDM and SCM
- Integration of SCM tools with other tools

Some of them are presented in more details below:

5.2.3   Workspace Management

Almost all models and tools have acquired check-out, check-in model. Module versions are saved in a repository that manages versioned entities. Users can check-out specific module version, work on it in a local structure (workspace), and finally they can check in the modified module which creates a new version of the modules in the repository. In some research modules the check-out/check-in process is under tool control, not seeing from the user, or the development tools incorporate versioning and workspace management, as for example ref [Ask99c].

The workspace management is related to a question of what happens with a module when being checked out from the repository? The simple tools leave this control to the users, the more advanced tools keep the work space within their controls, such as ClearCase [RationalCC] and partially Continuus [Continuus]. In all present solutions the

workspace is mapped to the file system, since the modules are usually implemented as files.

Although several vendors provide smart solutions regarding workspace, there are several questions related to workspace management that must be worked out:

- How to manage workspace in a distributed development?
- How to manage workspace for entities different from files, for example entities in development environments, databases, etc.?
- How to implement different views of workspace, with completely different structures and levels than a hierarchical structure?
- How to integrate Workspace with structures required by other tools (typically project structures in different IDE (Integrated Development Environment) tools)?

5.2.4   Versioning Models

The model with version branches and delta algorithms (which save only the differences between successive versions) become the standard way of implementation for module versions, revisions and variants.

The main stream of solutions is based on branching mechanism described in Chapter 3. The repositories do not contain complete versions, but the differences between the versions are saved using delta algorithms. Many tools use line-based delta algorithm, where the differences between the lines of two versions are saved. There are however binary-based delta algorithms, where binary differences, byte per byte, are recorded. The versioning models work best for pure text (ASCII) files.

A trend in versioning models is to distinguish different levels of versions. The main question what make differences between two versions remain: is it a number of different lines or different bytes, or the differences managed on the higher level - differences of properties of entities, a functional changed introduced in the system, or something else?

One higher level of managing changes is use of change sets, logical changes introduced in the system. Managing changes on this level gives better separation of the abstract, functional change from its implementation. It can, however, produce module versions which never existed before and never have been tested. By time it was realized that the classical versioning and a change-set versioning are complementary and many SCM tools use them both. Many SCM tools combine these two methods today. Change management is usually provided in form of tasks which have multiple purpose: They define an action, i.e. a change to be introduced in the software, the actors (responsible to do perform the action) and finally they collect information about which parts of the system have been changed.

Another level of managing differences is in the semantic level - recognize the "logical" differences between two objects. The demands for viewing differences on this level have increased with the complexity of software entities, which cannot be mapped to files or to line of files. Such examples are objects in object-oriented programming,

tables in relational database, CAD models, different type of documents, etc. To recognize these types of differences, SCM tools must understand semantics of the objects. As these semantics are not standardized, there is no possibility to have a SCM tool that will understand all of them. For this reason we can see that other tools (such as Integrated Development Environments - IDE) include some of SCM functions, typically version management. This approach is efficient within particular domains, but closed for the management of the entire system which usually exists of many different types of entities with completely different semantics.

Another possibility is to formally specify the common, standardized version management on standard infrastructures and semantics of entities. In that perspective XML looks promising and flexible and powerful enough to specify different formats that can be used from different tools. Still, not too much have been done with XML within SCM, except in WebDAV (see SCM and WebDAV section below).

5.2.5   Distributed Configuration Management

The importance of Distributed Configuration Management has grown with a tremendous increase of development and usage of software, with the globalization of software development and software marketing. Specialization and outsourcing have also influenced to the development process. Finally the Internet and WWW have dramatically improve the communications possibilities utilized by companies in distributed development. Requirements of distributed development are different from local development. More and more SCM tool vendors provide support for distributed configuration management, and this have been a topic for researches during several last years.

There are different aspects of distributed configuration management. Management on the source-level is an extension of the "local" management. There exist several excellent examples of successful management in the Open Source domain, such as Linux, KDE, Godzilla (Netscape 6.), using simple SCM tool such as CVS. In most of these cases the central repositories accessible via Internet contain all data on one place. Different SCM tools offer support for a more complex management where repositories are distributed over the networks.

Another level of management is support for distribution components, or packages, related to a distributed release management [Hoek97]. In this case the management of already-built and ready-for-integration objects is taking place. This type of management become more important since more and more the components ready for use are available on the market.

A detailed analysis of SCM and distributed development, and use cases in Swedish industry can be found in the report [Ask99a] and in Asklund's licentiate thesis [Ask99b].

5.2.6   SCM on the Web

With the WWW explosion, the number of files to be managed increase dramatically, and a life time of a document or a document version decrease in the same way. A new

challenge for SCM appears - how to manage this enormous amount of items? A similar question related to dependency management arises - how to manage unlimited number of hyperlinks, that are being changed literally every second. But, another type of questions erase - how can SCM use benefits of the Internet and WWW?

SCM trends related to WWW can be divided in two categories:

- How to integrate WWW and SCM? How SCM can utilize WWW?
- How to keep under control evolution of Web-based information.

SCM and WWW - WebDAV
WebDAV (Distributed Authoring and Versioning) defines HTTP extensions necessary to enable distributed web authoring tools to be broadly interoperable. Some view DAV as a network filesystem suitable for the Internet, one that works on entire files at a time, with good performance in high-latency environments. Others view DAV as a protocol for manipulating the contents of a document management system via the Web. Web-DAV provides a network protocol for creating interoperable, collaborative applications. Major features of WebDAV are:

- **Locking and Versioning** (concurrency control): long-duration exclusive and shared write locks prevent the overwrite problem, where two or more collaborators write to the same resource without first merging changes. Versioning support, similar to that provided by RCS or SCCS supports operations such as check-out, check-in, and retrieval of the history list. The ability to directly retrieve a previous version of a resource (allowing links directly to previous revisions) is also supported. Built on top of the versioning layer is the configuration management layer, which provides support for workspaces and configurations, allowing versioned collections of versioned resources to be worked on. Both layers support concurrent development.
- **Properties**: XML properties provide storage for arbitrary metadata, such as a list of authors on Web resources.
- **Namespace manipulation**: Since resources may need to be copied or moved as a Web site evolves, DAV supports copy and move operations. Collections, similar to file system directories, may be created and listed.

How does DAV benefit (geographically disperse) teams of web site authors and developers? Web sites typically gather together information from diverse sources, often from people who are geographically separated. Using a DAV server, the HTML pages, images, and other information that comprise a Web site can be directly authored by the primary sources of the information. Even sites which use a staging process, DAV provides significant benefit for the first stage, where information is first entered into the approval process. DAV can also be used by the tools which transfer pages from server to server along the approval workflow.

WebDAV is ongoing project, but it is already built in some of the Internet Browser. As we can assume that collaboration over the Internet will become more and more common, we can expected that the importance of WebDAV will increase in the feature.

SCM clients as thin clients
There is a trend in many domains to use client applications plugged-in Web-browsers. A very important advantage of this approach is possibility to use the applications without installing them. Technologies such as cgi or asp scripts, and Java applets or COM components are usually used for that purpose. It is interesting that there are no SCM tools using these technologies. A prototype based on Java [Hum97] has been presented on SCM-7, but it led to a conclusion that TCP/IP or Java RMI level are more appropriate for client/server connections than WWW-based connection. A probable reason why WWW technology is not used for SCM lies in the fact that SCM deals with configuration items which in practice are files, or directories, so, close to the operating and file systems. WWW-based applications are, on the opposite, designed to be independent of file systems.

Web Engineering supported by SCM
The Web explosion in recent years has the enormous impact on many activities in everyday life, but also impacts on many engineering disciplines. For SCM is it a big challenge - how to support activities related to Web processes. Web itself meets a number of challenges related to configuration management. Some of them are [DRT00]:

- Speed of change
- Variant explosion
- Dynamic content
- Process support

These challenges define the requirements of configuration management. Many vendors are aware of these requirements as well as of large opportunities but also of growing competitions and many vendors include WWW-support in their SCM tools. The first three factors from the list above determine somewhat different goal usually defined by SCM: Instead of emphasizing the reliability and accuracy, the proper amount of work is placed on the first priority. The artifacts with short life (days or even just hours, such as web-pages) should be managed as simple and as fast as possible, while more critical data will be treated in more formal and rigorous way.

Most of the SCM tools give support for mapping structures of web-pages to SCM structures. Some of the vendors [QNN00] talk about Web Object Management, defining configuration items on the higher abstraction level. In the Web application we find different types of objects including:

- Programs (Java and ActiveX typically on the client-side)
- Graphics
- Sound
- Video

- Links (that is, relationships between objects)
- HTML (which usually, but not always, encapsulates simple text)

Web Objects will comprise these artifacts different in form and thus differently treated by SCM (for example in versioning, merging, differences).

There are several books discussing support for WWW processing. One of them is "Configuration Management - the missing link in web engineering", written by Susan Dart [Dar00], where the challenges, requirements and SCM processes for managing Web pages are described in details.

5.2.7   Component Configuration Management

In recent years we have recognized a new paradigm in the development process: From a complete in-house development, to a development process which has focused on the use of standard and de-facto standard components, outsourcing, COTS (commercials off the shelf). The final products are not closed, monolithic systems, but are instead component-based products which can be integrated with other products available on the market. The new paradigm increases the efficiency of development and the flexibility of delivered products, but at the same time increases the risk of losing product configuration consistency. Software systems based on standard components are the results of a combination of pure development and integration of components. The requirements for conventional use of SCM remains, but new requirements related to component management appear in all phases: in the design, integration and run-time. The integration part, i.e. configuration, and version management of the components becomes more important. New aspects of SCM arise in the run-time phase, as components are usually loosely coupled, and their update is allowed in the run-time environment.

When delivering components or products, which are part of a target system, we face two problems:

- We cannot predict the behavior of the entire environment of the target system. The system may contain another product, which uses the same component as our product. The relations between components, and the changes we may obtain by introducing a new version of a component, are uncertain.
- A more serious problem is the dynamic behavior of the system configuration in the run-time environment. If we permit component-updating during the run-time, by updating dynamic libraries, we could be facing a situation in which a new component version works for one product, but not for another. There are also different aspects of updating, such as moving or copying an application from one computer to another, or automatically generation of code.

The basic requirements of component-based systems are identification of components and keep them under version control. One of the basic problems when developing component-based systems is that it is difficult to keep track of components and their interrelationships. This demand emerges already in the requirement phase, in which we want to identify and select the most appropriate components. Later, during the assembly and

deployment process, or when upgrading components, the problem of components identification and dependency management becomes even more important. One way to maintain control over upgrades is to use component identification and dependency analysis. These are well known techniques for managing system configurations during development, but are rarely applied in managing run-time dependencies. Knowledge of the possible impacts of an update is important, since it can be used to limit the scope of testing and be a basis for evaluating the potential damage of the update. The dependencies can be showed in a form of a dependency graph. The dependency graphs can also be used to facilitate maintenance by identifying differences between configurations, e.g., making it possible to recognize any deviations from a functioning reference configuration. Some early works have been done in this field [Lar99, Lar00] and we expected that Component Configuration Management will be a new SCM discipline addressing these problems.

In close relation to Component Configuration Management is the Product Line development approach. In Product Line many variations of products are built from core components, always present in the products, and many optional components or different variations of components. In component identification, selection and integration process configuration management plays an important role. The SCM methods are still not established in Product Lines. Problems in both Component management and Product Lines are similar and deal a lot with identification and structures, which approaches similar problems (and maybe solutions) present at PDM.

5.2.8   Process Aspects

In late nineties the basic problems with version, selection and workspace management have been solved and a support for middle-size projects established. Support for large-size development projects came into focus. One of the main challenges for large projects is to keep track of the project's process. The process issues also become interesting with the wide acceptance of Maturity Capability Model (CMM) from industry. One of basic functions of level 2 of CMM is configuration management. Many SCM vendors claim today that they in general support CMM level 2. By including process support, SCM tools have enlarged its area - SCM support is not only focused to developers and tools for automation of certain actions, but also to management, planning and project follow-up. An SCM-plan, including planning for resources, efforts, SCM milestones (most often baselines) and even SCM metrices, become a standard part of project documents.

A tool-based process support originates from change management in which the role of change sets, originally used for generating new module versions. Different notations have been used to management of change process: tasks, change requests, ToDo lists, etc. Their role is to connect the activities performed with the available resources, modules being changed, etc. Several tools, for example ClearCase, Continuous or PVCS have support for workflow. The workflow functions cover cases beyond SCM functions and are similar to support form other tools, including several PDM tools.

SCM deals with huge amount of data that can be used for measurements. There are numbers of different software metrics and they can be classified in relation to the relevant development process, products or resources. Typical product metrics are size metrics (number of lines of code, number of documents, etc.), quality metrics, etc. Process metrics are on the other hand a result of measurements related to the different phases of the development process. Process measurements help us to understand the processes concerned, to control them, improve and predict them. Size metrics are relative easy to obtain, especially if SCM is systematically used, and size metrics are easy to interpret, since they are often obtained by direct measurements. It is more difficult to provide the process metrics. SCM tools which integrate process management with SCM can provide process metrics. Change management controls relations between logical changes and physical changes implemented in files within a system and give information about the system changes and in this indirect way provides information for process metrics. Since a process is a set of related actions distributed in time, the time parameter must be considered in such measurements. Using SCM data to obtain different measurements related to the changes is widely recognized as an efficient method. By placing Change Requests under version control we get a new dimension of the measurements and it is possible to trace the history of changes themselves. Software measurements and SCM have been of interest of research and practice in industry [Crn00].

5.2.9   PDM and SCM
As it is stated in this report the domains covered by PDM and SCM are very similar in principle, but the implementation and processing is different in may details. Although PMD uses CM and SCM deals also with products, the unification of these fields was not of big interest neither for researches nor for tool vendors, with few exceptions - Jacky Estublier [Est98] gave an extensive analysis. As products and systems consist more and more of both hardware and software the requirements for the integration of PDM and SCM has become more genuine, and this report is an excellent example!

5.2.10   Integrated Environments
During the last decade most of the SCM tools have passed a transformation from being large, cumbersome, expensive, complex and user-unfriendly and first of all isolated from other tools, to be more user-friendly, easier deploy and much better integrated with other engineering tools. Still, there is long way to go, this is especially true for the integration with other tools. As this is a more practical, and first of all marketing issue, it is of primary interest for vendors and not for researchers, as the widespread opinion is that "in principle the problem is solved". The main idea of integration is to hide SCM as much as possible in everyday developers' work. Many SCM tools have API compatible with MS Visual Source Safe which makes it possible an easy integration in MS Visual Studio. Still, today there exists no standardized API for SCM functions.

5.2.11   Trends in Industry
SCM discipline has achieved a level of maturity that allows its implementation in many variants: There are over 100 SCM tools available on the market. In addition to these stand-alone tools, many other tools have built in SCM functions. Even if SCM tools are often seen as too complex, too expensive and inflexible, development organizations buy

or use free SCM tools. This was not the case in the 80s when most of larger organizations developed their own SCM tools. In most cases the companies use basic SCM functions, i.e. versioning and baselines, while additional functions, such as change and process management are not used so often. The most important requirements from industry are not sophisticated functions, but simplicity and interpretability with other tools.

5.2.12   SCM Trends- Summary
To summarize the trends and nearest future of SCM, we cite Estublier (SCM: a Roadmap, [Est00]):

Despite the many limitations and expected improvements discussed in this paper SCM proved to be one of the very few successful software engineering technologies. Indeed, the market is booming, with over $1 billion sales in 1998 and has excellent perspectives since only about 25% of mainframes, 15-20% of workstations and 5-10% of development PCs currently has today an SCM system. Forecasts are about $2.1 Billions sales by 2000 and $3.3 Billions by 2002.

The future of SCM research and tools is unclear. The basic services will become understood, mature and stable enough to be standardized. They will fall into the public domain, as basic services anyone can expect from a platform, for instance versioning, rebuilding, basic work space support and so on. Vendors will lose their control on these low level services.

Vendor added value will come from their ability to build, above this level, an SCM system providing core topic advanced services, targeted toward a specific client: a specific data model and versioning capability, specific concurrent engineering facilities and so on. A major change is that second layer will be considered as a basic SE platform kernel, rather than a stand alone product. The issue will be to standardize and "componentize" SCM systems in a way allowing us to build easily, tailored and highly efficient solutions.

Indeed, above this kernel, will be plugged a number of specific tools dedicated to a facet of SE, like process support, concurrent engineering control, project support; or dedicated toward a specific application domain like Web support, PDM control, deployment, electrical or mechanical tools and so on. SCM research and vendors are currently starting to address all these issues, but with limited scope, and not necessarily with all the requisite expertise. On this layer, for each tool, it is unclear who will take the lead, not necessarily SCM vendors. The way all these tools will cooperate, to build a fully fledged, evolutive and efficient SE environment, is an active research topic (mega programming, COTS federations etc.). This last issue, interpretability control, can become another area where SCM research and tools can contribute; further, as quoted "several researchers believe that Configuration Management environments are the real process-centered environments".

Nevertheless, in the near future, provided the number of core topic issues yet to be solved and the efficiency, scalability and usability issues, no one of these evolutions will

be seriously addressed by SCM vendors. SCM tools will still grow, propose proprietary solutions and still consider SCM as an isolated domain. SCM research should take the opportunity that many SCM challenges are currently under work in other SE domains, to foster a synergy between these research domains and SCM, bringing in the experience and know-how that made the strength of SCM, showing a path toward the useful and successful tools software engineering needs.

6 Analysis of needs and solutions

A complex product is often developed by several groups/teams, developing their specific type of documents/components, e.g. mechanical, ASIC, electronics, software design, etc. Depending on the type of product developed, different types of teams are involved. The result from each team is assembled (on system level) to one final product ready for production and marketing. Common for all product development activities is a strong need to manage data both on the system level, between the teams, and within the teams.

On the system level, information about the contents of the products, customers, vendors, traceability of included components and their related documents, suppliers, baselines, releases, prices, markets, etc. are needed. The project manager needs to know if the project is following the time schedule, the CM manager needs to know the current configuration on the system, the production engineer needs to find the product ready for manufacturing, the sales person needs to find information about the product to present for a customer. All data needed by these roles is collected and managed as metadata at, what we call, the system level, see Figure 18.

The different development groups need support for their daily routines, especially during the hectic development phase. In practice, this means tight integration with the development tools, such as tools for mechanical design and graphical viewing, tools for electrical design and test, etc. The functionality needed is e.g. support for concurrent development, awareness, document management, version control, and workspace management.

PDM tools are very good at managing the system level. Many of the tools are also well integrated with development tools, e.g. within the CAD domain. However, it does not exist any, good enough, integration with a software development environment. This means that for a product configuration as depicted in Figure 18a, it is possible to find a tool meeting all the requirements for product data management, but in configurations as depicted in (b) and (c) it is harder (impossible).
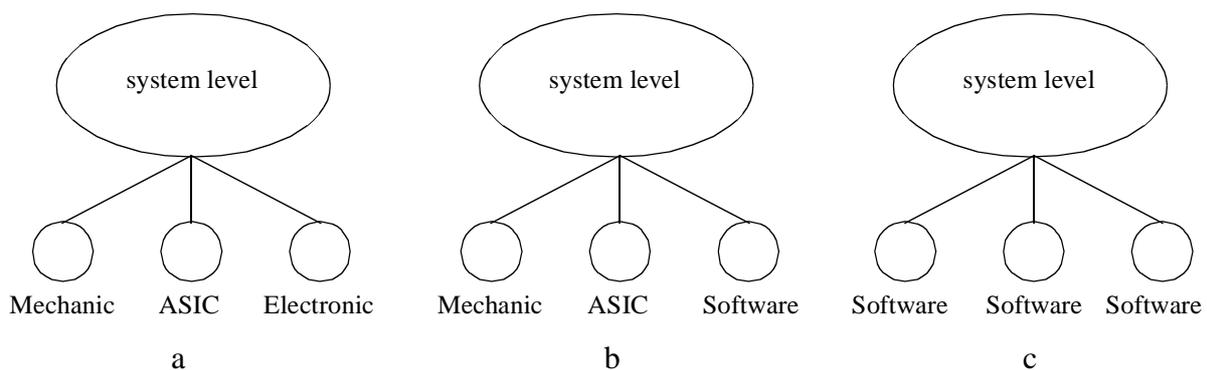
Figure 18 *Different product configurations. (a) is a product consisting of hardware only, (b) consists of both hardware and software, and (c) consists of software only. They are all complex enough to require a 'system level'.*

A short analysis is thus:

- SCM-tools do not cope with the requirements on the system level.
- PDM-tools do not cope with the requirements during the development phase of software.
- It does not (yet) exist any integration between a SCM tool and a PDM tool good enough.

This means that all products containing software (development), complex enough to require a 'system level' are 'in trouble'. I.e. even pure software products may require PDM support not easy to integrate with software development.

In the rest of this chapter we more thoroughly describe the requirements for hardware and software development respectively going through scenarios. Based on these requirements we then analyze the PDM and SCM domains and their functionality. Finally we suggest some solutions how to integrate PDM and SCM tools.

6.1   Requirements and needs

The basic needs for data management are the same for the two domains. The developers work in their various tools for development and data management, such as CAD/CAM with PDM tools or modules in hardware development, and compilers, text editors, and SCM tools in software development.

When developing a complex product the following development scenario may be defined: The development starts from the requirements on the total product or system level. All the system requirements are broken down to requirements on several sub-systems. It is not always possible to decide directly whether the sub-system should be implemented in software or in hardware. This can be found out when the requirements have been broken down, or after an investigation of which kind of implementation is the most suitable, or simpler, a decision may be based on experience from previous product development projects. These sub-system requirements will be treated and implemented in sub-projects. A sub-project may be either a hardware or a software implementation of the requirements. When the sub-system is released and tested, all components from the different sub-systems will be integrated and verified together.

However, in most cases the development scenario is different for a product involving both hardware and software. Concurrent sub-projects developing either hardware or software are running. In this case common information has to be available for all in the total project. Roles like project managers, product managers, and CM managers need to follow-up the whole product and project. The developers in each sub project have to perform their various tasks as efficient as possible; the needs and requirements at this level are the same as for pure hardware and software product development.

Pure hardware-based products require a development process different from the one used for pure software products. When developing systems that combine both hardware

and software, the process is more complicated, and thus the roles of SCM and PDM. In chapter 6.1.1 we will state demands on PDM and SCM for hardware, and in chapter 6.1.2 we will state demands on PDM and SCM for software development.

6.1.1   Development and maintenance of a pure hardware product

The most important PDM related requirements for hardware development is to manage documents and product structures in a PDM system, but the information is used in other parts of the development process too. The documents and the product structures are used to develop, manufacture, and maintain the product.

- Customer demands are translated to system requirements. These requirements are combined with internal requirements from stakeholders such as production (e.g. low production cost, fit in to existing production processes, and low assembly cost) and maintenance.
- The mechanics and electronics designers need to manage documents and product structure, which they do in modules in their CAD/CAM and other tools, or integrated PDM systems.
- The developers need to version control their parts (components in the product structure) and documents (not necessarily concurrently on the same document or part). A part can be extremely complex. Some CAD systems allows the developer to divide a part in different zones, where you may check out a zone separately, but not the whole part. To each part there are associated files (e.g. FEM, CAM), in which several users may change and update the same information.
- The manufacturing and purchasing need to be able to purchase and manufacture components, assemble and deliver the product to a customer. They need a BOM to do forecasts, purchasing and production line planning, tool ordering and assembly. Drawings and other documents are used to specify how to make tools and manufacture parts.
- To establish a baseline or a release, it is necessary to freeze all included products and documents. Beyond the release point, formal change management is needed to manage changes during the design phase. In hardware development, late changes are very expensive, due to large costs to redesign tools and/or production process.
- The information needs to be available for many other roles, e.g. marketing and sales.

Development of this product category does not contain any kind of software, and has no requirements on SCM.

Figure 19 shows an example of where the information is created during the development phase. The information is stored in different tools, e.g. requirements will be stored in RM (Requirement Management) tools, Change Requests or Trouble Reports in CR/TR tools, product structures and BOMs in the PDM system or in CAD/CAM tools.
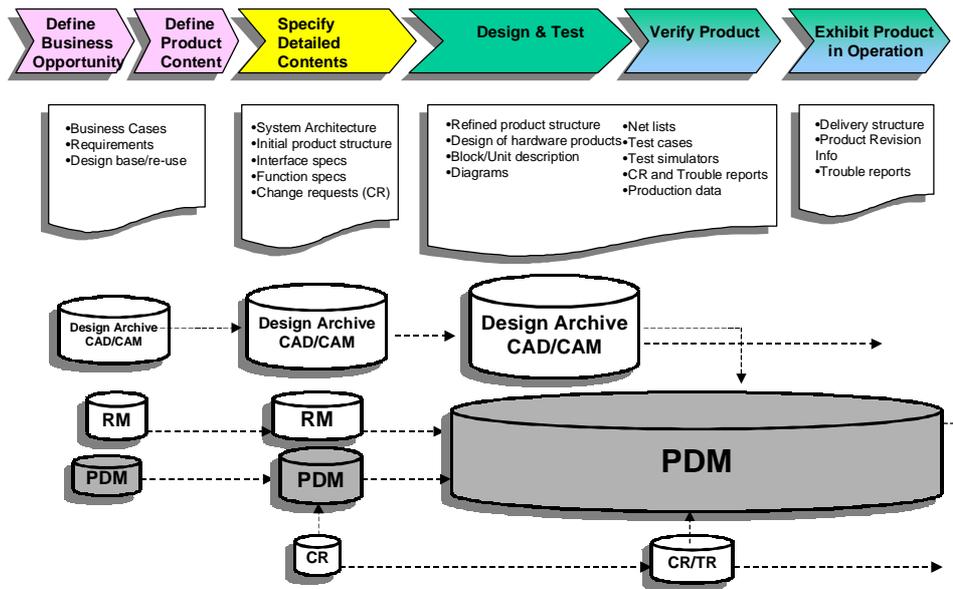
Figure 19   *An example of processes, information and systems for hardware development*

6.1.2   Development and maintenance of a pure software product

The most important needs for software developers during software product development are to manage documents and source code files.

- The customer demands are translated to product (functional and non-functional) requirements. As one of the main characteristics of software is that it is easy to change, many requirements address the improvement of the existing products which are part of the maintenance process.
- During the development phase, developers use different tools such as text editors for writing source code and compilers and linkers for generating executable code. Tools can be integrated in a common development environment, exchanging product and process data.
- A complex software system is generated from a large amount of source code files. To concurrently develop such system, software development has a need for detailed version control (using revisions, branches, and merges). This is done in a SCM system. There are many more versions of software files and databases than for hardware. For an efficient development process, SCM tools must be well integrated with other tools.
- There is a need of pointing out a baseline, with the frozen source code files that build an executable. When the baseline is verified, the developer may release it for further testing or delivery to the customer. When the software product is released, formal change management has to be performed. In comparison to hardware development, it is much easier in software development to introduce changes (which in general does not require as much effort as for introducing a change in the hardware product, but it may be very costly if the change is not introduced in a controlled way).
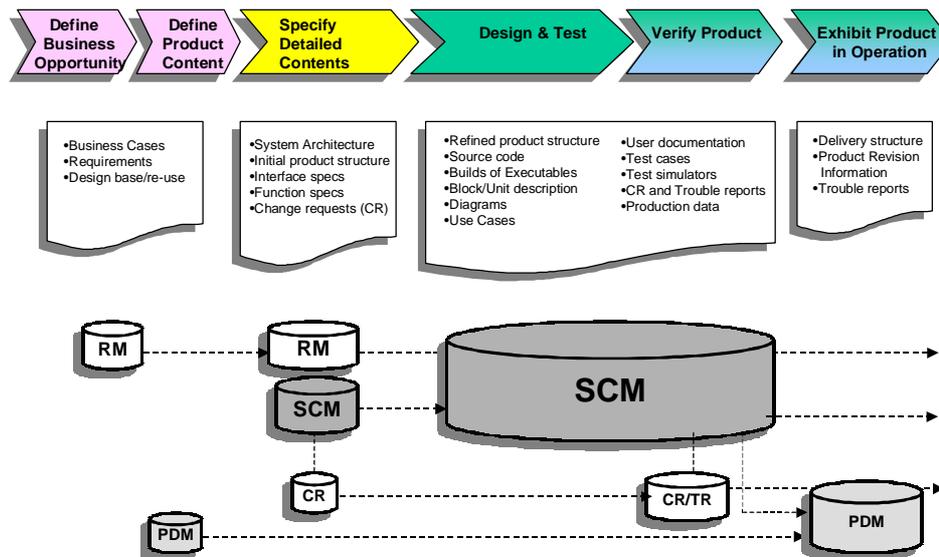
Figure 20   *An example of processes, information and systems for software development*

- The source code and deliverables must be stored in archives. All components and their documents must be available in the right revisions for maintenance purpose and for future releases of the product.
- To deliver the product to a customer, manufacturing needs to be able to produce the product packages suitable for delivery, for example the product can be burned on media or published on the web. The manufacturer also need to retrieve relevant documents.
- For marketing and sales the information (e.g. brochures, product information, product descriptions) need to be available for communication with customers.

This product category does not contain any kind of hardware and the requirements for having a PDM system are not obvious. Possibly it can be used in the latest phase in the manufacturing and partially in the maintenance phase.

Figure 20 states an example of where information is created during the development phase, e.g. requirements will be treated in requirement management tools, Change Requests and Trouble Reports in CR/TR tools, and software components in SCM tools.

6.2   Analysis

6.2.1   Cultural differences
There are cultural differences between software and hardware development.

- People from both domains do not understand the requirements for the other domain. Still many software developers believe that SCM tools can manage data at the system level. 'PDM-people', on the other hand, often believes that software
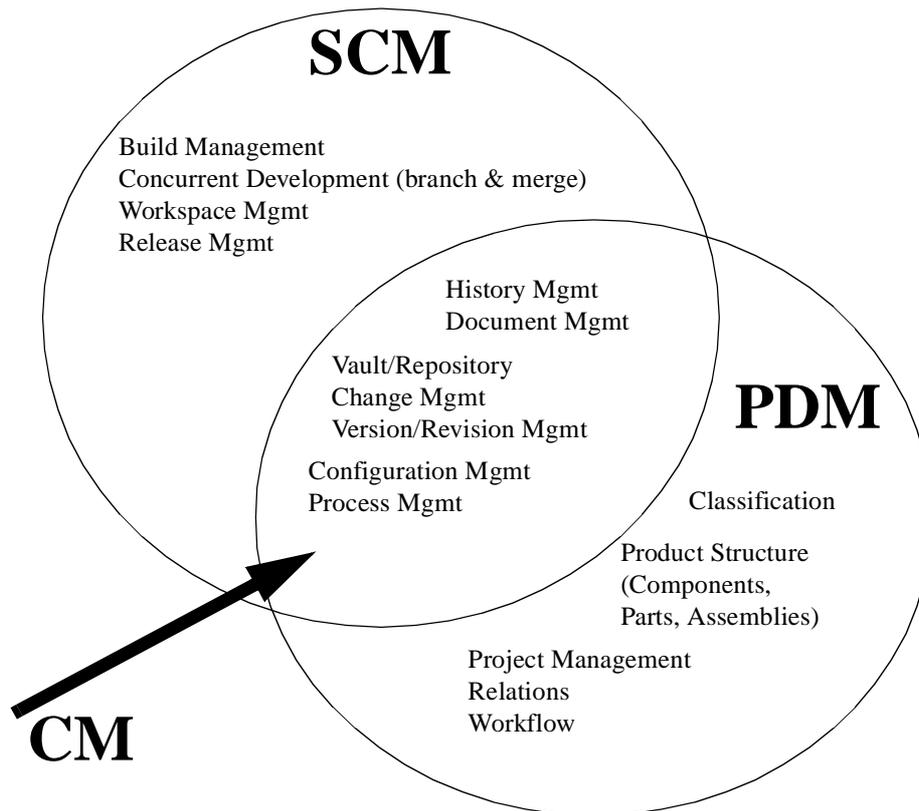
Figure 21   *The main functions and overlap of PDM and SCM*

  is binaries only, easy to put under control of a PDM system as any other component.

- They use different terminology, e.g. configuration control, which is the definition and management of product configurations for PDM, and the control of changes to a configuration item after formal establishment of its configuration documents for SCM.
- PDM is still most focused on the management level, to control and manage the product. SCM has, during the last years, also developed a strong support for the daily routines, i.e. to really support the developers during the development phase.
- There are few, if any, contacts between a software and a hardware developer.

6.2.2   The PDM and SCM dilemma

The overlapping functions are many between PDM and SCM. Figure 21 depicts the most important functions from both domains. As the figure shows, there are many functions supporting the same or similar processes. In the overlap between the two domains are common activities. However, even for these common activities, mostly different tools and different processes are used.

When developing a product consisting of both hardware and software, we are heading some more needs.

The development of hardware and software sub-projects are done concurrently, and there is a strong need to manage those together. To manage this type of product we need to have access to all product data in a collected form on system level. Roles such as Product Managers, Project Managers and CM Managers need to follow-up and manage documents, product structures, requirement baselines and status accounting during the development phase for the whole product/system. Software executables and related documents together with hardware components and their documents have to be stored or referenced (pointed) to in the PDM system. In this case there is a strong need for an interface between the PDM and SCM systems. To get access to all product data in a collected form, we need to have all metadata regarding the product and/or files collected in a single system with interpretability with other systems, e.g. with pointers to data stored in other systems. However, to work in the same system is not a solution, due to different requirements on the tools from hardware and software developers, and other roles.

In practice, not all people are working on the same parts of the system. Hardware developers use their development tools. These tools are more or less tightly integrated with the PDM system, and they send information automatically to PDM. They do not work concurrently on the same document. Software developers work in their tools and do not have any benefits from using PDM system until the product is ready for integration and verification or customer use. To be efficient, software developers must be able to work concurrently on the same file.

We cannot use the same PDM and SCM tools during the entire development process due to:

- different nature of software and hardware artifacts and different development processes;
- different tool requirements from hardware development and software development;
- similar functionality, represented by the overlapping areas in Figure 21, is not performed in the same way in the two systems;
- changes are more critical in hardware development than in software development.

Furthermore, we cannot separate the development processes since we need specification, traceability, requirement management, change management, release management, product structure, etc. on system level.

To obtain a full support for managing development and product assets (items) we cannot use just one of those systems. Neither PDM nor SCM can give the full, integrated support during the entire product life-cycle. We need the functions of booth systems.

Hence, this implies a need of an overall system integrated with the tools from respective domain.

6.3   Different Scenarios in an Integrated Environment

For having an interoperability between PDM and SCM, there is a need of making a decision of what kind of data each system should manage, and where the different data should be archived before making any decision about functionality to support.

To illustrate the roles of SCM and PDM systems in an integrated environment we passes through two scenarios giving several use cases. The use cases are good input for analysis.

Except all hardware related product information the PDM system should manage delivered software deliverables (compiled code), libraries, path to documents in SCM, and traceability to the components in the deliverables (e.g. config spec in ClearCase). SCM is the software development environment for developing software products which includes versioning, build, branch, and merge functions. SCM tools for software development is to compare with CAD/CAM tools for hardware design.

To provide a minimum integration between a PDM and SCM system at least following prerequisites has too be defined. They are:

- The PDM system is the "umbrella tool" and will manage metadata for the software information.
- The SCM system is the archive for all software information, i.e. all files will be stored in SCM, see Figure 22.
- The PDM system has to have a CLI (Command Line Interface) or API.
- The SCM system has to have a CLI or API.
- The PDM system will manage product structure and the revisions of the product.
- All items in the SCM system which are in PDM, have to be marked with an attribute.
- SCM will automatically register the specific information in PDM.
- SCM has to store the traceability (e.g config spec in ClearCase) of all included source code files in a delivery.
- To get a user-friendly interface, there must be good performance when retreiving data from SCM to the PDM system.
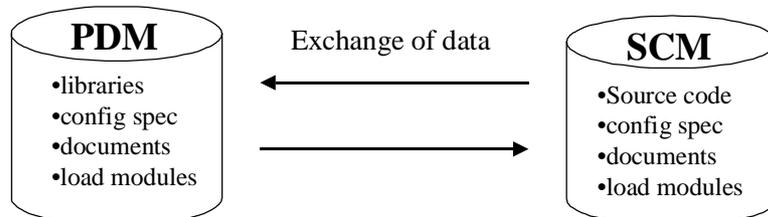
Figure 22   *An example of exchange of metadata*

Figure 22 shows an example of what kind of metadata that will be managed in the PDM system. The figure also shows that SCM will store the actual source code, config spec, and document files, i.e. the repository within SCM will be the archive.

6.3.1   Scenario A: PDM - User Interaction

Here follows certain scenarios related to users in the PDM system when all files will be stored in the SCM system. A user wants to:

- *query* for a document. The PDM system has to get a copy of the document from the SCM system and present it to the user in a read-only format.
- *check out* a document. The PDM system has to check out the document in SCM, set an attribute to signal the SCM system not to allow other users in SCM to check out the file, set an attribute with current user who updates the document, and open it up for the user. Then the user may update the document.
- *check in* a document. The PDM system has to delete the attribute signaling the SCM system not to allow other users to check out the file, delete current user attribute, check in the file in SCM again, and update the version of the file in PDM.
- *delete* a document. The PDM system checks if the user has the access rights and is allowed to delete the specified document. The PDM system has to check if the document is included in a freezed product or not. If the document is not included in a freezed product, then PDM will search for the document in SCM. Delete it in SCM and its labels and attributes, and delete all metadata and relationships in PDM.
- *get* a document, deliverable file, or library files. The PDM system has to look-up the actual file in SCM, and do a copy of it and present to the user. This is needed when a consumer has to assemble a product.
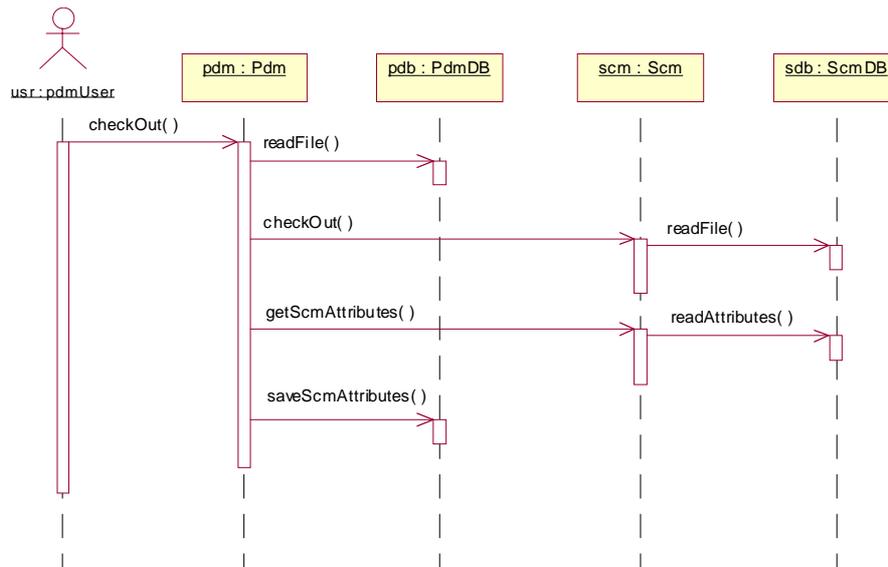
Figure 23   *Check out sequence*

Figure 23 depicts a sequence diagram of checkout [1].

6.3.2   Scenario B: SCM - User Interaction

Here follows some scenarios for a SCM user who has to provide the PDM system with information. The user wants to

- *register in PDM* one or several files. SCM has to provide PDM with the full path to the file(s), labels/attributes with relevant metadata, product (e.g. product number) the information belongs to, the product one level above in the product structure (needed if a brand new product is registered), revision, and status. When registering a library or an executable in PDM, the trace (e.g. the path, and config spec in ClearCase) to all included software source code files has to be registered as well.
- *register a new product revision*. The SCM system need to know next relevant revision of the product. SCM asks PDM for next revision for the specific product (new or old). PDM allocates the revision as work-in-progress, sets appropriate attributes, and deliver the new product revision to SCM. SCM sets the product revision label to the new received revision.
- *register a new document version.* SCM asks PDM for next revision for the specific document (new or old). PDM allocates the revision as work-in-progress, sets appropriate attributes, and deliver the new document version to SCM. SCM sets the document version label to the new received version.
- *check out.* The file is already registered in PDM and may be freezed. SCM checks if the attribute not has been set by PDM that another user already has checked out

---

1. Andreas Sjögren, Mälardalen University, has made the UML sequence diagram figures
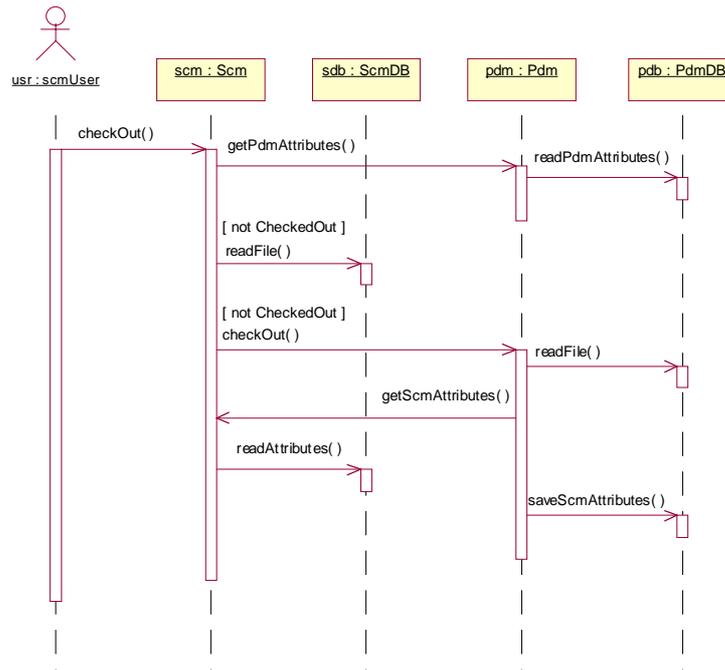
Figure 24   *Check out sequence*

the file. If the file is not checked out, then check out in SCM, check out in PDM, set an attribute in PDM with current user, and change status to work-in-progress. The user is now allowed to do the changes in the file.

- *check in.* First SCM will do an check in of the file, check-in in PDM, reset the attribute for current user, and change status in PDM.
- *uncheck out.* SCM is locking up the file lock and does an reset of the current user attribute and changes the status in PDM.
- *update product status.* SCM sets appropriate attributes in PDM (depends on the company specific development process).
- *delete document.* SCM will use the document number label and search for the document in PDM. If the document is not included in a freezed product, SCM will delete the document in PDM, and the document will be deleted in SCM including all labels and attributes. If the document is used in a freezed product, an error message is returned to the user and no document is deleted.
- *query on product revision.* SCM will use the product number label and ask PDM for current revision of this product. The result will be presented to the user.
- *query on a product structure.* SCM will use the product number label and do a query in PDM to retrieve the full product structure where the actual product is included in. The result will be presented in SCM to the user.
- *query on document.* SCM will use the document number label and do a query in PDM to retrieve the actual document. The result will be presented to the user.

The sequence diagram shown in Figure 24 shows a similar complex activity for PDM user.
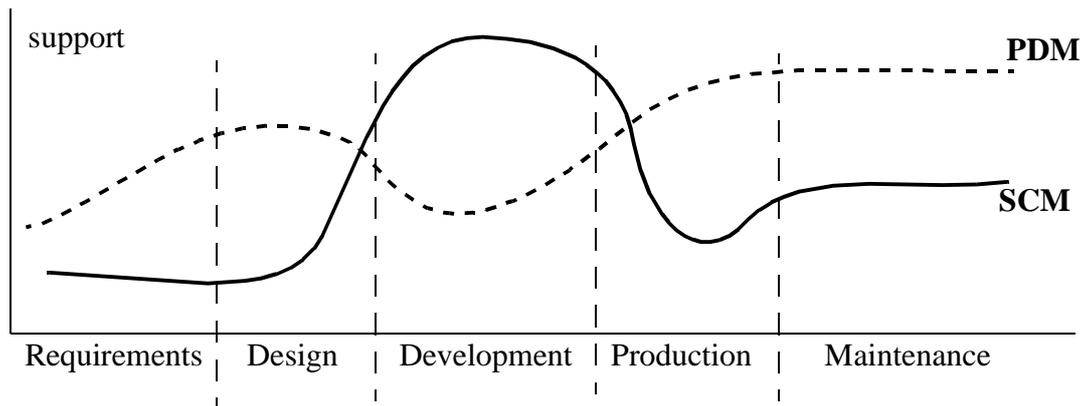
Figure 25   *PDM and SCM system support during the product life cycle*

The use cases presented here indicate that both SCM and PDM must exchange information and send a request for change of state of entities of the other tool.

6.4   Possible integrations

6.4.1   Interoperability of processes

The characteristics of the two systems emerge from the nature of the products developed. In the product life cycle, PDM is focused more on the system design phase, hardware development, and later on the production and maintenance/support phase. The software development phase is less present in PDM. Rather the final results of the software development is taken under control of PDM. On the contrary, in the software product life cycle, the development phase is usually the most intensive part (despite the intention of software engineering to move more activities to the beginning of the product life cycle). Consequently the tools bring into focus the support for the corresponding processes.

Figure 20 and Figure 25 schematically show the support provided by these tools during the product life cycle of a product. From a pure functional point of view (not implementation), the PDM and SCM tools fit together and completely cover the entire product life cycle which gives attractive predisposition for their integration. A possibility of integration is even more attractive as the trends in both systems are enlargement of the area of control already covered by the other system. Since software products are complex, SCM becomes more similar to PDM due to the structuring and configuration of the products.

6.4.2   Integration of data

As many companies are faced with a situation requiring the use of both systems, the question is which kind of integration or cooperation that can be achieved with these two systems. There are different types of integration:

- manual export-import;

- weak integration;
- full (tight) integration.

A full or tight integration can be achieved by using a common infrastructure, common interfaces and common data. This means that we need a common product model, a common evolution model and a common process model. A common support for the process model can be used with the present tools, but other models with today's functionality are too different for use as common models.

In a weak integration PDM and SCM keep their own infrastructures and data, and have well-defined interfaces.

A more simple integration is to be able to import and export data or to share data via a container.

Which parts of the tools that can be integrated depends on the tools, the requirements on shared information and the process used. The minimal integration required is on the version and configuration level. As PDM do not have branch and merge for version management, it is suitable for software file versioning to be under the control of the SCM tool. From a PDM point of view, it is more interesting to keep information about specific versions of files collected in a configuration or in a baseline, but not all possible versions of files created in the software development process. This means that a list of files (source and executables) containing the pointers to the actual files is saved, as shown on Figure 26.

In a tight integration we may want to keep all the information in PDM that is created and present in SCM. This approach (PDM-centric) gives a more complete information stored in one place. On the other hand, when an object is registered in PDM, it is also under version control in the PDM system. Now it is possible for the user to check out and check in the object, still under version control in the PDM system, and in this way there is a risk that the versions of the objects (files) in PDM and SCM become un-synchronized.

Conclusions

To be efficient and to have survey of the information/data, we need a tight integration between the two domains. To obtain a tight integration wee need a

- compatible product model;
- compatible version management;
- interaction between the process models;
- common terminology and understanding of respective domains.
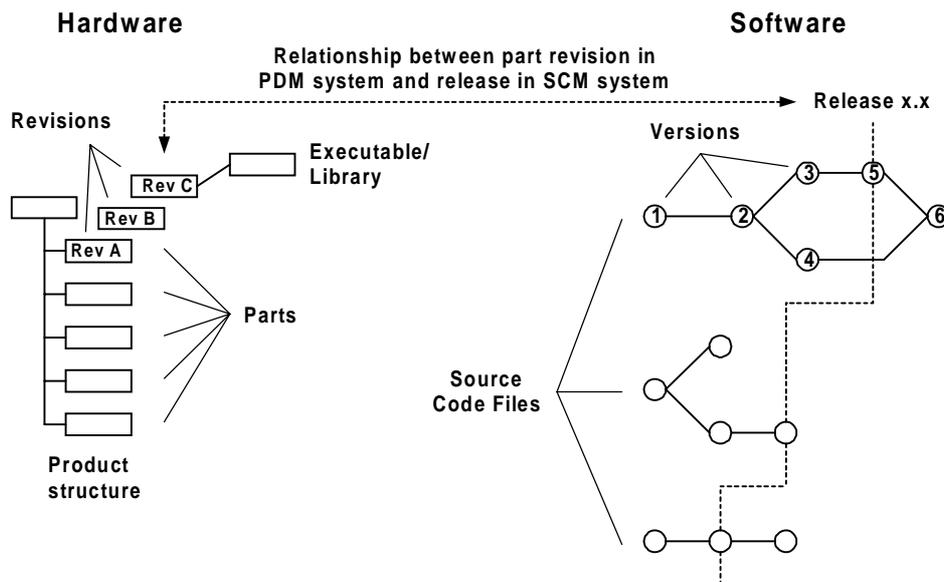
Figure 26   *Example of an Information Management Architecture for PDM and SCM*

6.4.3   Possible integrations on system level

Theoretically there are many good reasons that point in favor of integration of PDM and SCM. In practice, there exist many problems. First, the integration requires sharing or exchanging of data between different tools from the same domain but from different phases. Neither PDM nor SCM has yet solved this problem completely. A more serious problem arises when data must be shared or exchanged between the tools from these two domains. The second problem is the choice of the tools and methods from the overlapping areas. Even if a particular tool provides excellent support within one domain, it does not mean that it is suitable or well integrated within the second domain.

In general there are three possibilities to achieve the interoperability: Make a full integration of PDM and SCM, make a weak integration or make no integration at all, but use some import/export functions. A full integration can be achieved by using a common infrastructure, common interfaces and common data. This means that we need a common product model, common evolution model and common process model as [EFM98] have concluded. A common support for the process model can be used with the present tools to some extend, but other models with today's functionality are too different for use as common models. A tight integration would require enormous efforts from PDM and SCM vendors, and from users of the existing tools. It is not likely that the vendors can or want to take this step.

No integration gives a poor interoperability, requires many manual interventions and there is a high risk to introduce inconsistency in the system. This solution is unfortunately the most common solution which companies are forced to use.

In a weak integration PDM and SCM keep separate infrastructures and data, but have well-defined and efficient interfaces between them.

Figure 27 shows an ideal integration model, building a common application user interface to manage both PDM and SCM functions and present them via common interface to the user, and a common repository.

Unfortunately this model cannot work well with the tools available on the market today. Most of them have a poor APIs, which provides incomplete (if any) functionality. The repositories (if exist at all) are tightly integrated with the business layer, so it is impossible to separate a repository from one domain and integrated with another. Another challenge for both tools, independently of each other, is interoperability with other engineering tools. Interoperability requirements will lead to the emergence of better and clearer APIs.

As an API for PDM and SCM tools does not provide full functionality, and the repositories are merged with the applications, the solution shown in Figure 27 will appear in practice as shown in Figure 28. When possible, the tools and users use the common API, but there will be cases in which tools communicate directly to PDM or SCM systems.

This solution may generate problems as it duplicates the information and it may require certain manual actions, which may introduce inconsistent states for the PDM/SCM combination. For a more robust and efficient integration, PDM and SCM vendors should provide a more tight integration that manages consistency of data. Also a more powerful and usable API should be provided. This API will then be used by different engineering tools. As PDM covers a larger part of the total product life-cycle and as PDM deals with metadata (i.e. description and structuring of data), it is natural that communication with the user is via PDM, as much as possible. Still there will be tools that directly interact with SCM tools. This type of integration is shown in Figure 29. SCM tools can be integrated with PDM tools as other engineering tools (such as Inte-
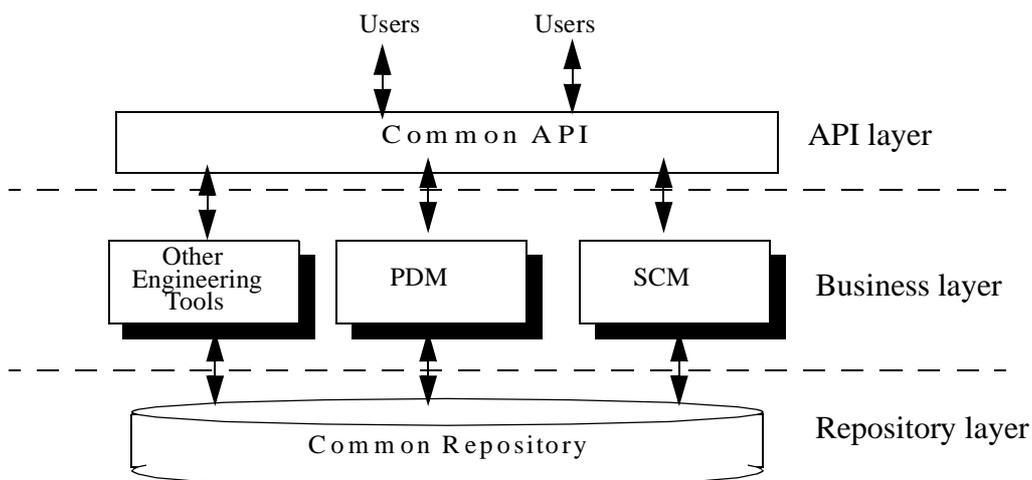
Figure 27   *PDM and SCM integration – Common API and common Repositories*
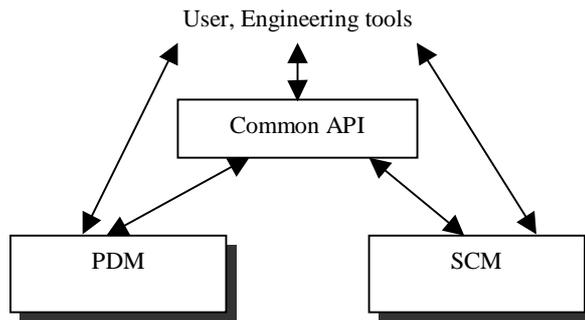
Figure 28   *Direct and indirect use of tools*

grated Development Environment tools) are integrated. PDM tools use the API from SCM. Users communicate only via PDM, which is responsible for updating information for both PDM and SCM data. This model provides better control of the consistency of duplicated data. However a similar problem remains. There is already an integration of different development tools and SCM tools and it is unrealistic to assume that such integration will not be used independently of PDM integration. This means that there will always be a possibility that data in only one of the databases is modified, thereby introducing an inconsistent state. To avoid such possible inconsistencies, a database synchronization process must be included between the databases on a periodical or an interrupt/trigger base.

Another problem, which already exists in both systems, becomes more acute in the integration process. Both tools are complex and as a consequence have complex and often user-unfriendly interfaces. When integrated, the system user-interface will be even more complex.

Conclusion
To get an efficient and useful integration between PDM and SCM tools we need:

• common APIs for both PDM and SCM tools;
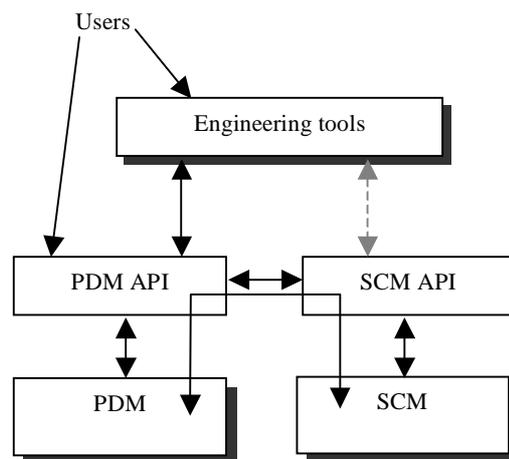
Figure 29   *Partial direct PDM/SCM integration*

- synchronization functions between the systems;
- it must be possible to directly use PDM and SCM tools;
- the interoperability functions between PDM and SCM should be easy-to-use;
- the interoperability functions between PDM and SCM must be accurate.

6.5   Examples of integrations

Today there are two known first attempts for integration between a PDM system and a SCM system. The integration is between the SCM system ClearCase [RationalCC], and the PDM systems Metaphase [*SDRC*] and eMatrix [eMatrix] respectively.

6.5.1   Integration between Metaphase and ClearCase

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 products from ClearCase into Metaphase. Those software products will be managed in the PDM system together with all hardware products within the same product structure and delivery structure to the customer.

Later releases will cover the aspect of manage information in ClearCase from Metaphase.

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

It is not easy for the end-user to understand and use the interface. First of all the end-user has to have 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 is by using the configuration specification rule files, but this function will not be available until later releases.
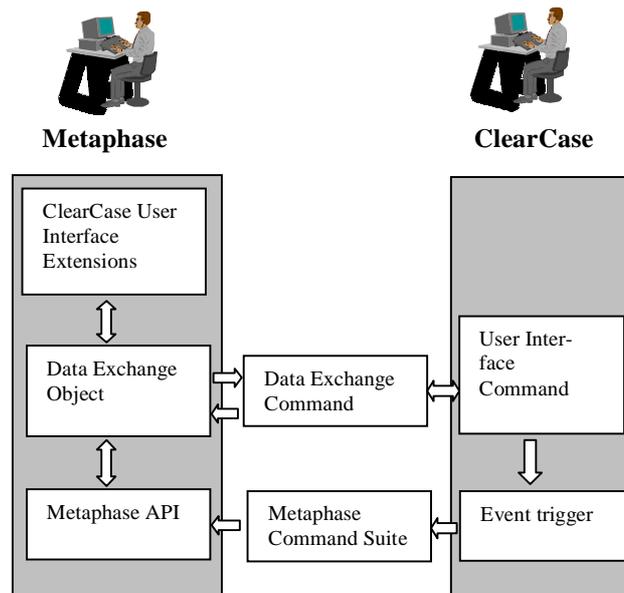
Figure 30   *Data Exchange Architecture*

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 meta data 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 meta-data are placed in both systems.

These required conditions show the complexity of the developed integration. 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. To reach a mutual understanding, the terminology has to be either the same in the two systems or there has to be a translation table. All PDM systems, including Metaphase, are designed to meet the needs for managing

product information of hardware products and has its own terminology, not the SCM terminology.

6.5.2   Integration between eMatrix and ClearCase
A short overview of the results from a project integrating eMatrix and ClearCase is presented in the interview "PrakTek" on page 175.

In Appendix B.9, PrakTek, a short overview of the results from a project integrating eMatrix and ClearCase can be read, including the motivation and log-term goal, the roles of the two tools, and some implementation details. Below is a shortened list of characteristics of the interface/integration:

- The goal is to support the development of a complex software product.
- ClearCase stores all source code.
- Currently eMatrix manages baselines and change requests. Objects (part of a baseline or referred to by a change request) are created in eMatrix representing specific versions of files in ClearCase. These objects contains 'links' to Clear-Case, which makes it possible to retrieve information about them from ClearCase. It is also possible to set labels and attributes in ClearCase from eMatrix, e.g. when the status changes due to operations in eMatrix (by e.g. the project manager).
- The interface is partly event driven (one direction only). Some operations in eMatrix also results in actions within ClearCase. The other way around is, however, entirely manual. Operations in ClearCase, which affect data in eMatrix, must be manually updated in eMatrix.

7   Conclusions

In this report we have presented the two areas of PDM and SCM side by side, which we hope will widen the understanding across the two worlds. We have also listed, explained, and compared the main functionality provided by typical tools from each domain. Finally we have made an analysis of what the similarities and differences are between the two domains (or more concrete between PDM and SCM tool functionality), and what the consequences are.

In this chapter we conclude the analysis and consequences. We also look forward and, very shortly, list some interesting topics for future studies.

7.1   Why so important?

All companies interviewed are currently working with their PDM and SCM systems. They have understood the importance to support a product's entire life cycle. To only support the development phase or only parts of the product is not good enough. They have also understood that manual transfer of information between tools is time consuming and does not cope with iterative processes. A global solution for the entire product during the entire life cycle in combination with specific support during the development is thus important. This does not necessarily mean more control in terms of picking on individual developers, but an overview of what is happening and that correct (versions of) documents are developed and released. An overview of the evolution and group awareness of what is happening right now is important for all roles, e.g. developers and managers.

Special tools have been developed either to manage large product structures with a lot of meta data, or to manage thousands of files and directories concurrently modified by distributed developers. It is now important to really work on integrated solutions.

7.2   Observations

Below is a summary of our observations grouped into cultural differences, functional similarities, functional differences, and tool integration.

7.2.1   Cultural differences

There is a cultural gap between people working in the two domains. Both believe that their methods and tools are superior and that problems in the other domain easily can be solved using their tools. A better understanding across the domains is complicated due to different terminology. Even if there are some common terms they do not have the same meaning. People from both domains rarely meet each other, and if they do, they do not understand each other due to, for example, different opinions on what a product is, terminology, when to go to production and so on. Also for vendors the culture differences are cumbersome when discussing PDM and SCM with customers.

7.2.2   Functional similarities

The PDM and SCM domains have moved towards each other in terms of functionality. One reason for this is that the data they manage is becoming more and more similar. Traditionally PDM managed product data and SCM software. This is no longer the full story. Hardware contains today a lot of embedded software. Similarly software is often packaged as libraries or components which then is managed more or less like hardware.

- *concurrent development:* Both PDM and SCM provides an infrastructure to manage large amounts of data, continuously modified by many developers. Most of the functionality provided in one domain is also supported by the other.
- *change management*: both use ad-hoc modules or integrated systems to support change management.
- *document management*: PDM has well-established document management. The trend within SCM is to manage documents in the same way as PDM does.

7.2.3   Functional differences

Despite the similarities above, there are still a lot of differences.

- *standards*: the PDM domain uses the standard STEP. The SCM domain does not follow any standard.
- *data model*: PDM uses an object oriented data model. SCM only uses files and directories.
- *evolution model vs. versioning*: PDM has revisions, variants, effectivity, and versioning of domains. SCM has revisions, variants, branches, and merge.
- *concurrent development:* PDM has shared repositories and locking to provide synchronization. SCM has workspace control where the developers can work concurrently on independent copies on the same file using branch and merge facilities. SCM also provides advanced selection facilities in order to create configurations making it possible for developers to have their own configuration in their own workspace.
- *data representation:* PDM makes a distinction between the metadata and the actual data, and focuses on managing the structured metadata. SCM does not have this distinction and has less support of managing metadata.
- *infrastructure:* Both PDM and SCM have server-server communication. PDM has a master - slave architecture, while SCM has a more symmetric communication. See Figure 16.
- *distributed development:* In PDM metadata and/or data is replicated, but in SCM the file(s) and its metadata are replicated together.
- *replication:* In PDM locking must still be used also when data is replicated to several servers. SCM uses branches to provide a more flexible (loosely coupled) replication that does not need constant connection between servers.
- *product model:* PDM is strong on the product structure which is the same as the part structure. In SCM the product model is very weak and is based on the operating system model (i.e. files and directories)
- *build management:* for SCM this is essential, but does not exist in PDM.

7.2.4   Tool integration

Integration of PDM and SCM tools is just in its beginning. To get a common process flow between the tools requires better triggers, APIs, etc. Current interfaces only supports 'weak' integration as described in "Possible integrations" on page 84. Moreover, the interfaces we have seen is unsymmetrical. Changes to data in the PDM tool can result in commands launched in the SCM tool, but not vice versa. This is probably due to previous demands on integration with other tools, see "Application Integration" on page 43, but vendors promise more symmetric interfaces in forthcoming releases.

Today there is no standard API used in the systems. This makes it hard to develop an integration between different tools. Consequently, to achieve an integration is cumbersome and costly.

7.3   Consequences

The functionality provided by tools from the both domains are to a large extent overlapping. This may lead to that people from both domains believes that their tool can cope with all the requirements from the other domain. Our conclusion is that this is not true. There is still requirements in both domains not covered by tools from the other domain. There is no one tool solution - yet.

PDM is focused on managing the metadata of a complete product. Many PDM tools are also integrated to the development environments for some specific data types, e.g. CAD/CAM files. SCM in mainly focused on the development phase of software products, managing both some meta data but especially the program source code itself. Our conclusion is that SCM tools and PDM tools thus complement each other rather than compete. The conclusion is also that there is a need for better integration of these tools to provide a solution covering both the system level and the development of system components.

Since the project and product managers normally do not want to handle more than one tool, it is necessary for one tool to be able to retrieve information needed from the other tools. The experience from one of the interviews is that such 'umbrella' tool can not be implemented using a simple file structure, even though only used for released and stable data (and not during development). Even for this simplified use, better support is required for managing different data types and to support the release process.

The differences in data models, use of standards, and management of metadata versus the documents themselves, make it harder to develop interfaces and to integrate PDM and SCM tools. The first versions of 'weak integrations' have seen the light, but tighter integrations that makes it possible to access all data from both tools need more time and thinking. In this work it is very important that industry actually state their requirements on such integration based on a process and methods they want to achieve.

7.4   Future work

There is a need for research to come up with a common model for product structure and versioning valid for both the PDM and SCM dicipline. The leading vendors seam to be to worried about their market share within their respective domain. It is to be hoped that new vendors take the opportunity (and risk) to create either a new tool supporting the requirement of both domains, or an interface supporting tight integration.

Integration of PDM and SCM tools (shuffling data between them) together with replication may be a technically hard nut to crack. In which order should data be moved and copied?

A better understanding of 'the other' domain and more collaboration between them is the first and most necessary step to take. Without agreement on the requirements we can not come up with a solution.

References

[Abr97]   Abramovici M., Gerhard D., Langenberg L., Application of PDM technology for Product Life Cycle Management, *Preprints from 4th International Seminar on Life Cycle Engineering*, Berlin, Germany, 1997.

[And92]   Andreasen, M.M., Designing on a "Designer's Workbench" DWB, In *Proceedings of the 9th WDK Workshop*, Rigi, Switzerland, 1992.

[Ansi98]   ANSI/EIA-649-1998, National Consensus Standard for Configuratin Management American National Standards Institute, 1430 Broadway, New York, NY 10018, USA, 1998.

[App01a]   Links to Software Configuration Management on the World Wide Web. http://www.enteract.com/~bradapp/links/scm-links.html Brad Appelton. 2001.

[App01b]   Collection of definitions of SCM. http://www.enteract.com/~bradapp/acme/scm-defs.html. Brad Appleton 2001.

[Ask99a]   Asklund, U. Distribuerad utveckling och Configuration Management för programvarusystem. Sveriges Verkstadsindustrier, V040073, 1999.

[Ask99b]   Asklund, U., *Configuration Management for Distributed Development - Practice and Needs*, Dissertation 10, Department of Computer Science Lund University, 1999.

[Ask99c]   Asklund, U. et. al: The Unified Extensional Versioning Model, System Configuration Management, SCM-9 Lecture Notes in Computer Science, no. 1675, 1999.

[Bab86]   Wayne A. Babich. Software configuration management : coordination for team productivity.Addison-Wesley. 1986. ISBN 0-201-10161-0.

[Beck99]   Beck, K. Extreme Programming Explained. Addison-Wesley. 1999.

[Bla98]   Blanchard, B.S., Fabrycky, W.J., *System Engineering and Analysis*, 3rd ed., Prentice-Hall International Ltd., London, UK, 1998.

[Boeing]   Lean Enterprise: Implementing Lean Practices, http://www.boeing.com/commercial/initiatives/.

[BW98]   Clive Burrows and Ian Wesley. Ovum evaluates: Configuration Management, Ovum Ltd 1998. ISBN 1-898972-24-9

[Cim98]   Product Data Management - The Definition, CIMdata Inc., Ann Arbor, MI, USA, 1998.

[Cim99]   *CIMdata Europe'99 - Conference Proceedings*, Nice, France, 1999.

[Cim01]   Product Data management and Computer-Aided SoftwareEngineering CimData
            http://pdm.project.ericsson.se/cimdata/pdm_case/pdm_case.htm

[Continuus] Continuus Software Corporation, Continuus, http://www.continuus.com/.

[Crn97]   Crnkovic I., Experience with Change-oriented SCM Tools, *Proceedings of 7th Symposium on Software Configuration Management* , Lecture notes in Computer Science, nr 1235, Springer Verlag, 1997.

[Crn99]   Crnkovic I., "Why do some mature organizations not use mature CM?", *Proceedings of 9th Symposium on Software Configuration Management,* Springer 1999

[Crn00]   Crnkovic I., Larsson M., and Lüders F., "Software Process Measurements using Software Configuration Management", In *Proceedings of 11th European Software Control and Metrics Conference* , IEEE Computer Society, 2000.

[CW98]   Conradi R. and Westfechtel B., Version Models for Software Configuration Management, *ACM Computing Surveys*, volume 30,issue 2, 1998.

[Dar00]   Dart, S. *Configuration Management - the missing link in web engineerng,* Artech House, 2000

[Dic97]   C., Dickersson, PDM Product Data Management: An Overview, Society of Manufacturing Engineers,   Dearborn, MI, USA, 1997.

[EFM98]   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. 1998.

[Elm89]   Elmasri, R. Navathe, S. B., *Fundamentals of Database Systems*, The Benjamin/Cummings Publishing Company, Inc., Redwood City, California, USA, 1989.

[Elsitech]   Elsitech, Visual Intercept, http://www.elsitech.com/.

[eMatrix]   MatrixOne, Inc., www.matrixone.com.

[ENOVIA]   ENOVIA Solutions, www.enovia.com.

[Eri97]   Eriksson, H. E., Penker, M., *"UML Toolkit"*, Wiley & Sons, Chichester, 1997.

[Est00]   Estublier J., "Software Configuration Management: A Roadmap", In *Proceedings of 22nd International Conference on Software Engineering, The Future of Software Engineering*, ACM Press, 2000.

[Fel79]   Stuart I. Feldman. Make, A program for Maintaining Computer Programs, Software - Practice and Experience, 9(4);255-265, April 1979

[Fux00]   Fuxin, F. *et al.* (2000) *"Produktmodellering i amerikansk verkstadsindustri"* (*"Product modelling in the american manufacturing industry"*), Division of Computer Aided Design, Department of Mechanical Engineering, Luleå University of Technology, Luleå, Sweden.

[Gra98]   Grabowski, H., Kunz, J., Maier, M. *"Product and Document Management using STEP AP214"*, Presented at ProSTEP Science Days 17./ 18. June 1998, Wuppertal Germany.

[Har96]   Harris, S. B. "Business strategy and the role of engineering product data management: a literature review and summary of the emerging research questions" In *Proceedings of the Institution of Mechanical Engineers vol 210, Part B: Journal of Engineering Manufacturing*, pp. 207-220, ImechE, 1996.

[Heg95]   Hegge, H. M. H. *"Intelligent Product Family Descriptions for Business Applications"*. Ph.D. thesis, Eindhoven University of Technology, the Netherlands, 1995.

[Hoek97]   Hoek A. v. d., Hall R. S., Heimbigner D., and Wolf A. L., "Software Release Management", In *Proceedings of 6th European Software Engineering Conference*, Lecture Notes on Computer Science, nr 1301, Springer, 1997.

[Hub88]   Hubka, V., Eder, W.E., *Theory of Technical Systems*, Springer-Verlag, Berlin, Germany, 1988.

[Hum97]   Distributed Configuration Management via Java and the World Wide Web, Software Configuratin Management, SCM-7, Lecture Notes in Computer Science, 1235, Springer, pp.161-173.

[INCOSE]   International Council on System Engineering (INCOSE), http://www.incose.org.

[ISO9000]   ISO 9000-3:1997, Guidelines for the Application of ISO 9001:1994 to the Development, Supply, Installatio, and Maintenance of Computer Software, Geneva, Switzerland: ISO, 1997.

[Isa00]   Isaksson, O. *et. al*, "Trends in product modelling - an ENDREA perspective", In *Proceedings Product Models 2000*, 7-8 November, 2001, Linköping, Sweden.

[ISO95]   Kvalitetsledning - Riktlinjer för konfigurationsledning. Standardiseringen i Sverige (SIS) SS-EN ISO 10 007.

[IVF00]   Lecture notes from PDM-IG meeting (The Swedish Product Data Management Interest Group), 28 November 2000, IVF, Mölndal, Sweden.

[Jan93]   Jansson, L., *"Verksamhetsorienterade Produktstrukturer"* (*"Business Oriented Product Structures"*), Ph. D. thesis, Department of Production Control, Chalmers University of Technology, Göteborg, Sweden, 1993.

[Kel96]   Marion Kelly. Configuration Management - The Changing Image. ISBN 0-07-707977-9. McGraw-Hill, 1996

[Lar99]   Larsson M. and Crnkovic I., "New Challenges for Configuration Management", In Proceedings of 9th Symposium on System Configuration Management, Lecture Notes in Computer Science, nr 1675, Springer Verlag, 1999.

[Lar00]   Larsson M. "Applying Configuration Management Techniques to Component-Based Systems" Licentiate Thesis Dissertation 2000-007, Department of Information Technology Uppsala University. 2000

[Leb97]   David B. Leblang, Managing the Software Development Process with ClearGuide, Software Configuration Management ICSE'97 Workshop, Boston, May 1997, pro-ceedings, *Springer Verlag, ISBN 3-540-63014*-7, pages 66-80

[Merant]   Merant, PVCS Tracker, http://www.merant.com/products/pvcs/tracker/.

[Mes00]   Mesihovic, S., Malmqvist, J., "Product Data Management (PDM) System Support for the Engineering Configuration Process", *14th European Conference on Artificial Intelligence*, Configuration Workshop, August 20-25, Berlin, Germany, 2000.

[Microsoft] Microsoft, Windows Installer, http://www.microsoft.com/.

[MIL92]   MIL-STD-973, Configuration Management, Washington, DC: U.S. Department of Defense, Apr. 1992.

[Mil01]   E. Miller, *"Manufacturing Industries Move Toward Engineering Collaboration"*, CIMdata, Ann Arbor, MI, USA, 2001.

[Nym96]   Ulf Nyman. Konfigurationshantering. Sveriges Verkstadsindustrier, V040047, 1996

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

[pdmic]   The PDM Information Center, www.pdmic.com.

[Pik96]   Pikosz, P., Malmström, J., Malmqvist, J., "Strategies for Introducing PDM Systems in Engineering Companies", *Advances in Concurrent Engineering-CE'97*, pp. 425-434, Rochester, MI, USA, 1996.

[Pik98]   Pikosz, P., Malmqvist, J, "A Comparative Study of Engineering Change Management in Three Swedish Engineering Companies", *Proceedings of DETC'98*, Paper No DET98/EIM-5684, Atlanta, GA, USA, 1998.

[QNN00]   Quinn, Evan and Heiman, R. *Web Object Management: Bringing Order to Opportunity,* white paper, http://www.mks.com/wom/idc_wom.htm

[RationalCC]   Rational, Clear Case, http://www.rational.com/products/clearcase.

[RationalCQ]   Rational, Clear Quest, http://www.rational.com/products/clearquest.

[Sch93]   Schwarze, S., *"The Procedure of Product Configuration and Handling the Configuration Knowledge"*, BWI Research Paper, No.3, November 1993, Zentrum für Unternehmenswissenschaft, ETH Zürich, Switzerland.

[SDRC]   Vendor of the PDM system Metaphase. http://www.sdrc.com

[SEI95]   *The Capability Maturity Model.* Software Engineering Institute, Carnegie Mellon University, Addison Wesley 1995.

[SEI00]   SEI, Capability Maturity Model, 2000, http://www.sei.cmu.edu.

[Stark]   John Stark Associates, www.johnstark.com.

[STEP91]   STEP Part 1: "Overview and fundamental principles", ISO TCI194/SC4/WG5, November 1991.

[Sve00]   D. Svensson, J. Malmqvist, "Strategies for Product Structure Management in Manufacturing Firms", In *Proceedings of DETC'00*, Paper No DET2000/CIE-14607, Baltimore, MA, USA, 2000.

[SW94]   D. Schenck, P. R. Wilson, *"Information modeling the EXPRESS way"*. Oxford University Press, New York, NY, USA, 1994.

[Tic89]   Walter F. Tichy, RCS - A System for Version Control, Software - Practice and Experience, 15(7);637-654, July 1985.

[Tic94]   Walte Tichy (Ed.). Configuration Management. ISBN 0-471-94245-6. John Wiley & Sons. 1994.

[WC98]   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

[Whi91]   David Whigift. Method and Tools for Software Configuratoin Mana-
            gement. ISBN 0-471-92940-9. John Wiley & Sons. 1991.
[Wri97]   Wright, I. C. (1997) "A review of research into engineering change
            management: implications for product design", *Design Studies*, 18(1):
            pp. 33-42, 1997.
[YP]   http://www.cmtoday.com
[Zel98]   Zeller A., "Versioning System Models Through Description Logic",
            In *Proceedings of 8th Symposium on System Configuration Manage-
            ment*, Lecture Notes in Computer Science, nr 1439, Springer Verlag,
            1998.

Appendix A: Tools

In this chapter we present a number of SCM and PDM tools. This chapter is not an evaluation or a comparison of the tools. Neither is it a comparison of functionalities between SCM and PDM tools. The purpose is to show how the suppliers present their tools, which gives a hint of the functionality of the tools and the strategy behind them.

A.1   PDM systems

Commercial PDM systems have been on the market since the middle of the 80s. There are several systems available (not all presented in this report), but not as many tools as for SCM. If we extend the scope to document management tools, the number of systems available would grow significantly. In Table 2, the systems have been divided in three types, PDM systems, ERP systems and document management systems, and a few examples are given for each type of system. PDM systems are the type of systems described in this report. ERP systems have modules that can be used for PDM tasks. Together with other data management functionalities in ERP systems, and the fact they are all integrated in the same system, makes ERP systems worth considering also for PDM tasks, especially for those companies, which already have an ERP system. Finally, two examples of document management systems are presented. The functionality is limited compared with a PDM system, but can be sufficient for many companies.

Three of the larger systems are presented in more detail, in order to give an overview of the functionality they offer. The selection is not based on any criteria. Besides those presented (eMatrix, Metaphase/TeamCenter and Windchill), Enovia, SAP and Iman are also considered to belong to the larger players on the market. Sherpa was also one the big ones, but was acquired by SDRC (Metaphase/TeamCenter) and is no longer available for purchase.

A.1.1   eMatrix (www.matrixone.com)
eMatrix is one of the newer systems on the market. It started as an easy-to-use and easy-to-customize PDM system. MatrixOne has grown considerably the last years. The product focus is now "Intelligent collaborative commerce". This is based on three corner stones:

- *Value chain portfolio*
- *Net markets*
- *Data integration*

The eMatrix platform contains applications such as a system administration tool, a data modelling tool, web collaboration servers, and interfaces. When introduced, eMatrix was more or less a customisable toolbox. The data model can easily be redefined with a menu based tool included with the product. This feature makes it possible to adapt the product to a company's needs, the customer could even do part of this job himself. However, it often takes more time to decide what the needs are than to customise the

system. Therefore, several business oriented applications ready to use are offered together with the system in the *Value chain portfolio*.

- *Configurator Central* is a product configurator. It is integrated with the PDM tool, which makes it possible to build a configurable product definition early in the design phase.
- *Engineering Central* contains standard PDM functionality, such as product structure management, document management and change management.
- *Request Central* is a customer oriented application. It allows customers to send in request-for-quotations, checks if requests are valid and then tracks them all the way through design and manufacturing.
- *Software Central* coordinates hardware and software development. It contains software oriented features for requirements management, project management and change management. An integration to ClearCase provides a connection to the software development environment.
- *Supplier Central* is used to collaborate with suppliers within a virtual team. The suppliers get access to their customers product data, can exchange information with the customer and view change requests. The customer gets a single point of supplier information.
- *Team Central* allows the members of a project to work in a virtual team. In this virtual team the members can share data, take part in discussion groups and receive notifications of project events.

The *Net markets* is a new concept in the eMatrix product. It aims at providing new virtual markets on the net. Security is one of the important aspects for these virtual markets. Within these markets, the applications described in the *Value chain portfolio* are used.

The *Data integration* with other systems, is achieved with what MatrixOne call *Adaplet technology*. An adaplet is used to communicate directly with the database of a system integrated with eMatrix. eMatrix has packaged integrations to a large number of tools. Unlike the other suppliers, MatrixOne, has no interest in the CAD/CAM market. However, eMatrix offers integrations with most CAD tools on the market. There are also integrations to several ERP systems, ECAD tools, and other tools and systems, including an integration with ClearCase.

A.1.2 TeamCenter (www.sdrc.com)
SDRC (Structural Dynamics Research Corporation), has its roots in computer tools for mechanical engineering, such as the CAD tool IDEAS. SDRC entered the PDM market with Metaphase. Recently they released a new product, TeamCenter. TeamCenter has a new architecture based on java (J2EE). TeamCenter's component-based architecture consists of several modules.

- *TeamCenter Collaboration Foundation* is the traditional core functionality that controls and manages product information throughout a virtual enterprise. This foundation allows globally dispersed teams to author, share, and access product

information. It also includes capabilities for product definition, life cycle states, release management, and event notification.

- *TeamCenter Product Collaboration* includes part and document management, change management and advanced product configuration. It has a web based user interface.
- *TeamCenter Design Collaboration* is a CAD neutral collaboration environment. It allows users to create, share and manage virtual prototypes, independent of where they are located (in-house or supplier) and the CAD tool used.
- *TeamCenter Project Collaboration* is used for project management and collaboration. Projects can be scheduled, documents shared among team members, and discussions and notebooks support collaboration.
- *TeamCenter Requirements Collaboration* is a systems engineering tool for requirements management.
- *TeamCenter Enterprise Collaboration* draws information from dissimilar information systems and integrate it in user views. Supports integration with ERP systems, and enables supplier integration and CSM (component and supplier management).

TeamCenter will probably have the same integration possibilities as Metaphase. Metaphase offers ready-to-use integrations with several mechanical and electrical CAD tools, ERP systems, and has also announced an integration with ClearCase. Metaphase can communicate with other systems using the STEP standard. A module in Metaphase maps information from Metaphase's data model to a STEP protocol. The STEP data can then be used in another application.

A.1.3   Windchill (www.ptc.com)

Of the larger PDM systems, Windchill is the most recently introduced. Windchill was developed by PTC, which as SDRC is a well established provider of CAD systems. Windchill has had a web centric approach since it was introduced and is based on standard technology. It is java based and uses a web browser as client.

Windchill has like the other two tools an architecture consisting of a product platform, the *Windchill foundation*, and components to support collaboration in various parts of the product life cycle. The Windchill foundation contains the basic PDM functionalities.

- *Document management* includes the standard functionalities such as data vaulting, check-in/check-out, and version control, but also full text search of documents.
- *Structure management* creates hierarchical relationships between parts and associates documents with parts. The structure can be viewed form various perspectives depending on the role of the user.
- *Lifecycle management* controls the maturity of product information. A life cycle consists of a sequence of phases and gates that identify the state of an object and conditions required to let the object enter the next phase.

- *Workflow management* can be used to support processes within and between life cycle phases. Workflows can be defined, executed and monitored.

Besides Windchill foundation other components can be found.

- *Windchill PDM* provides extended functionality for product structure management and change management.
- *Windchill ProductView* allows users to view graphical information (3D models and 2D drawings), product structures, and other information through a web based interface.

Windchill foundation is used together with the other components to support the product life cycle. A number of applications are offered.

- *The Windchill ProjectLink for manufacturers/public B2B exchange* is used to share information in project teams. It lets project members store and view product information, and to attend collaborative meetings.
- *The Windchill customer collaboration* makes it possible for a company's customers to configure their own products and make queries for product information. From a manufacturers perspective, its products are presented in a searchable on-line catalog.
- *The Windchill manufacturing collaboration* aims at being an interface between design and manufacturing. It could increase knowledge capture and re-use, optimising manufacturing processes and sharing knowledge across the enterprise.
- *The Windchill product development collaboration* supports the development process, both within a company and between partners.
- *The Windchill supplier collaboration* allows a company to make its procurement process more effective. Suppliers publish their product information in a standardised way, which makes it easier to select between parts available and to re-use parts.

A technology called *Adapters* enables integration with ERP systems, PDM systems, and other kinds of information systems. Windchill has integrations with most common CAD tools on the market.

A.1.4   List of PDM tools
I Table 2, a short list with examples of PDM systems is presented. Both large and small systems are included, together with ERP systems and document management systems.

Table 2   *Product Data Management systems*

PDM systems

Enovia Solutions - Enovia	http://www.enovia.com
Matrix One - eMatrix	http://www.matrixone.com
SDRC - Metaphase/TeamCenter	http://www.sdrc.com
Eigner + Partner - CADIM/EDB	http://www.ep-ka-de

Table 2   *Product Data Management systems*

Parametric Technology Corporation Windchill	http://www.ptc.com, http://www.ptc.com/windchill
UniGraphics Solutions, Inc. -IMAN	http://www.ugs.com
Lascom - Advitium	http://www.lascom.com
Modultek - Aton	http://www.mudultek.com

ERP systems

SAP	http://www.sap.com
IFS Solutions	http://www.ifs.com

Document management systems

Documentum - Documentum	http://www.documentum.com
Cyco Software - AM Meridian	http://www.cyco.com

A.2   SCM Tools

SCM is a well established software engineering discipline, and today exist many SCM tools, probably more than 100. These tools cover a wide range of functions and they differ in complexity and prices. Although the history of SCM is almost 30 years old, the explosion of SCM tools available on the market happened in the late 90s.

The question of which tool is the best is irrelevant. There is no best tool, but there is the most appropriate tool, depending on the business goals. Many simple tools contain basic functions such as version and configuration management, which is sufficient for small development teams. Advanced tools supporting distributed development, change management, sophisticated building, etc. are used by large enterprises. In any case, a SCM tool only is not sufficient. A successful SCM utilization requires continuous process support. Even the most sophisticated tools must be integrated in the process, with other tools, and administration must exist. It is naive to believe that more sophisticated tools will require less resources for SCM. However, the overall results in productivity and quality of the product can be significantly improved.

This section gives a short list of SCM tools and some interesting references, and then some of the tools, characteristic for different type of users, are explicitly addressed.

A.2.1   References to the SCM Tools

A good starting point to explore SCM, and some PDM too, is the *CM Yellow Page* site http://www.cmtoday.com/yp/configuration_management.html. The page includes number of white papers and technical papers related to SCM, references to other SCM pages, an extensive list of commercial and non-commercial tools, upcoming conferences and seminars, consulting and education, and job opportunities.

A comprehensive SCM evaluation report is published by *Ovum*. "Ovum evaluates Configuration Management Tools", http://www.ovum.com/. (The report is not available free of charge.) The report can help very much in the SCM tool evaluation work. There

are also other evaluation reports and evaluation templates available, for example in Sweden [Nym96,Ask99a,Crn99], which can be used in the SCM evaluating process.

The *Configuration Management II Users Groups* has an excellent *CM Resource Guide On-Line*, http://www.cmiiug.com/Sites.htm. This page includes a number of references to articles, reports and proceedings, journals and newsletters, books, conferences, education/training providers, evaluation of PDM and software CM tools, copies of standards, list of organizations, user groups and research groups, CM software vendors standards, guidelines and position papers, and SCM web sites.

Since the number of SCM tools is large, it is not possible to describe them all here. We give a short overview of some of the tools, followed by a table with references to about 50 tools. The list is taken from CM Today Yellow pages http://www.cmtoday.com/yp/configuration_management.html and is divided in commercial and free tools.

A.2.2   Commercial SCM Tools

This sections gives a brief overview of some of the commercial tools.

CCC/Harvest, Computer Associates (http://www.cai.com/products/ccc_harvest.htm)

CCC/Harvest provides a comprehensive, integrated, repository-based change and configuration management solution that manages complex, enterprise-wide development activities. CCC/Harvest delivers an enterprise-wide solution for tracking software changes and managing the application development process in distributed environments.

CCC/Harvest includes the following main features:

- *Process-Driven, Integrated Change And Configuration Management*
  CCC/Harvest helps you to create and modify models of your development processes. By automating workflows, many routine tasks are also automated - such as notifications, approvals, and change migrations from one phase to another.
- *Problem Management*
  Problem Management can be automated and tracked with associated change packages and forms, which gives history information of specific changes and events that take place within a development process.
- *Application Development Tools Integration*
  This feature is enabled through the CCC/Harvest open architecture and customization option.
- *Management Reporting And Metrics Capabilities*
  Customized reports are easily created using Visual Basic or Java scripts.
- *Inventory Management*
  This feature provides authorized users with status information on the who, what, why, when, and where questions, regarding any software asset in the entire organization.

- *Automated Build Management*
  CCC/Harvest offers automated build management through the tightly integrated CCC/Openmake facility.

ClearCase, Rational (http://www.rational.com/products/clearcase/)

Rational ClearCase is one of the leading SCM tools that supports software development and maintenance life cycle. It is a tool for larger organizations, which can take full advantage of SCM, by having control of all assets in the development and maintenance processes.

The tool consists of several products that cover various phases and aspects of the software life cycle:

Rational ClearCase covers the basic SCM functions: Version management, configuration management, change management, and build support. All these functions give a wide support for sophisticated management of the development process. The basic idea is simple, a developer should not need to perform any unnecessary activities. However, through its view concept (a concept which several tools have adopted), SCM structures are flirted out, and developers work as with standard file system.

Rational ClearCase MultiSite is a product option of Rational ClearCase. A component of Rational's integrated change management solution, it enables parallel development across geographically dispersed teams.

Rational ClearQuest is a defect and change tracking system that captures and manages all types of change requests throughout the development life cycle.

ClearCase Attache extends the benefits of Rational ClearCase to developers using Microsoft Windows as their desktop development platform.

The main features of ClearCase are:

- Version control, workspace management, build management and process configurability
- Parallel development and distributed development
- Transparent workspaces for global data access
- Works together with Rational ClearQuest to integrate software configuration management and defect and change tracking
- Unified Change Management - Rational's activity-based process for change management
- Integration with Rational Suite for change management across the life cycle
- Web interface for universal data access
- Integration with leading IDEs and development tools.

Continuus, Telelogic (http://www.continuus.com/)

Continuus is an integrated task based change management solution. Any change in the system is initiated and controlled by a *task* - a description of what to do, who will do it, when is it to be done, etc. During the change process, the tasks gather additional information such as which software parts that have been changed, who did the change, the status of the change introduced, etc. In addition to change management, Continuus includes workflow-based management, which supports distributed, remote, and parallel development. Continuus' SCM tool is aimed for large organizations with a complex, possibly distributed, development process.

Continuus' SCM tool comprises several products tightly integrated, which are also possible to buy as separate options:

- *Telelogic CM Synergy* - Automated application life cycle management
  Telelogic CM Synergy is a task-based change management tool. Change traceability provides comprehensive traceability and impact analysis capabilities. It allows users to determine all the files and/or logical changes that are contained within a particular release. It also provides the ability for users to query for releases containing a particular file or task.
- *Telelogic ChangeSynergy* - Web-based change request management
  ChangeSynergy is an Web-based change request tracking and reporting system, which supports the change request process and enables organizations to respond to changes from both outside and inside sources. It offers control of development life cycles by providing, end-to-end task-based and process driven automation across teams in technical and business organizations.
- *Telelogic WebSynergy* - Enterprise web asset management
  WebSynergy is an enterprise software application designed to manage the creation, acquisition and deployment of all web-based assets.
- *Telelogic CM Synergy DCM* - Distributed Change Management
  With Continuus' DCM implementation, all development teams, regardless of their geographic location, achieve the full Continuus CM Synergy functionality.
- *Telelogic CM Synergy ObjectMake* - Object oriented Make facility
  ObjectMake is an object oriented Make facility supporting build processes, boosting productivity and increasing software quality. ObjectMake is an integrated part of the Change Management Suite.

MERANT PVCS, Merant, (http://www.merant.com/pvcs)
The MERANT PVCS product family includes integrated suites for enterprise software configuration management, as well as application specific choices for version management, change management, and build and release management.

PVCS is a family of the following products:

- *PVCS Dimensions*
  PVCS Dimensions is a comprehensive change management suite, combining tools for software configuration management with process models to automatically manage processes and workflows.
- *PVCS Content Manager*
  PVCS Content Manager is a browser-based solution for managing changes in web content. It supports enterprise-wide collaboration and facilitates timely updates of web applications and on-line content.
- *PVCS Version Manager*
  PVCS Version Manager is used for version control in team development environments. It organizes, manages and protects software assets during revision, and promotes team collaboration. PVCS Version Manager is integrated with more than 70 development environments and tools.
- *PVCS Tracke*r
  PVCS Tracker captures, manages and communicates changes, issues and tasks, providing basic process control to ensure coordination and communication within and across development teams and content teams at every step.
- *PVCS Configuration Builder*
  PVCS Configuration Builder ensures that applications can be reliability built in a reproducible manner; ensuring components from the same version are used.
- *PVCS Replicato*r
  PVCS Replicator enables distributed development, coordinating geographically diverse teams with shared requirements.

Visual Source Safe, Microsoft, (http://msdn.microsoft.com/ssafe)

Microsoft Visual SourceSafe (VSS) is a version management tool with rudimentary configuration functionality. The main feature of VSS is its relative good integration in Visual Studio (it is actually a part of Visual Studio, but the integration is not perfect). Unlike most SCM tools, VSS is not file-oriented (neither change-oriented), but project-oriented. Version management is performed within a frame of a project. Parallel development and file versioning is obtained on project level, not file level. Although VSS belongs to a low-level class of SCM tools, it is very popular as it comes together with MS Visual Studio.

The main features of VSS are:

- Practical for individual developers or small groups.
- Integration into Visual Studio, no extra costs if using Visual Studio
- Easy start for non-experienced programmer and SCM users
- There are many tools on the market which adopt VSS and build additional functions (such as change management, distributed development, etc.)

A.2.3   Freeware SCM Tools

Revision Control System (RCS)

RCS is one of the oldest SCM tool. It is very simple and include rudimentary SCM functions applied on files. The simplicity of RCS is outstanding and the reason why this tools is wide spread. RCS is also used as a basic tool for many other SCM tools. Even more interesting is the fact that the principles introduced by RCS (and previously SCCS) still exist in other more advanced SCM tools.

RCS is file oriented and the SCM repository is simply a directory, which contains versioned files. One versioned file contains one or several versions of that file. Baselines are achieved by setting a label on specific versions of the files. RCS includes difference and merge functions which can be used by users and by check-in and checkout commands.

Main features:

- Contains basic SCM functions (versioning, check-in, checkout, baselines, difference, and merge)
- SCM repository is the RCS underlying directory
- Simple and easy to use
- Practical for using on individual bases or in small groups
- Can be used as a base for building more sophisticated tools

Concurrent Versions System (CVS)

CVS uses RCS as a basic SCM tool, but in contrast to RCS, it manages directory structures. All files in a directory tree structure can be checked out or synchronized with working versions of files. CVS directly supports multi-user projects, as it uses a server client architecture. A TCP/IP connection is used for the client - server communication. There are many CVS clients, for example WinCVS (http://www.wincvs.org/), jCVS (http://www.jcvs.org/), command-line interfaces, emacs commands, etc. Similarly to RCS, CVS is a widespread tool because of its simplicity and flexibility.

Main features:

- Contains basic SCM functions (versioning, check-in, checkout, baselines, difference, and merge)
- It is a client - server application where the server manages the file repository
- Enables parallel development with concurrent changes of files
- Includes monitoring of files. Operations (check-in, checkout, etc.) on the specified files are reported to the users
- Simple and efficient
- Practical for using in small distributed groups

A.2.4   List of SCM Tools

The tables below lists some of the SCM tools available on the market.

Table 3: Process-Based Configuration Management Tools

AccuRev - AccuRev/CM AccuRev/Dispatch	http://www.accurev.com, http://www.accurev.com/i_prod.html http://www.accurev.com/i_prod.html
Intasoft - Allchange	http://www.intasoft.net, http://www.intasoft.net/products.htm
Computer Associates - CCC/Harvest	http://www.cai.com/products/ccc_harvest.htm http://www.cai.com
Serena - Change Man eChange Man	http://www.serena.com http://www.serena.com/html/changeman.ht http://www.serena.com/html/echange.htm
Rational - ClearCase	http://www.rational.com, http://www.rational.com/products/clearcase
ExpertWare - CMVision	http://www.cmvision.com
Continuus - Continuus/CM	http://www.continuus.com http://www.continuus.com/products/productsBB.html
Merant - Dimensions	http://www.merant.com http://www.merant.com/products/pvcs/dimensions/index.asp
Softlab - Enabler	http://www.softlab.com http://www.softlab.com/technology/frm_tech00.asp
Computer Associates - Endevor - OS/390	http://www.cai.com http://www.cai.com/products/alm/ccm/products.htm
Visible Systems - Razor	http://www.razor.visible.com
McCabe & Associates, Inc. -	http://www.mccabe.com
TRUEchange	http://www.mccabe.com/products/truechange.htm
Vertical Sky	http://www.verticalsky.com
Prosoft - XStream	http://www.prosoftcm.com

Table 4: Configuration Management and Version Control

+1 Software Engineering - +1CM - Solaris	http://www.plus-one.com http://www.plus-one.com/+1CM_fact_sheet.html
AccuRev - AccuRev/CM	http://www.accurev.com http://www.accurev.com/i_prod.html
Sequel UK - Alchemist	http://www.sequeluk.com http://www.sequeluk.com/alchemist/default.htm
BitMover, Inc.	http://www.bitmover.com/bitkeeper
Aldon	http://www.aldon.com
Industrial Strength Software - ChangeMaster - AS/400	http://www.industrial-strength.com
Reliable Software -	http://www.relisoft.com
Code Co-op - Windows	http://www.relisoft.com/co_op

Table 4: Configuration Management and Version Control

Agile Software Corp	http://www.agilesoft.com
Collabnet	http://www.collab.net
SourceCast	http://www.collab.net/products/sourcecast
NCI	http://www.nci-sw.com
Control	http://www.nci-sw.com/software_tools.html
ComponentSoftware	http://www.ComponentSoftware.com
CS-RCS - Windows	http://www.ComponentSoftware.com/csrcs
Data Services - ECMS	http://www.configdata.com/w002product.htm
Lockheed Martin	http://www.lockheedmartin.com http://www.lockheedmartin.com/syracuse/eaglespeed
Quality Software Components,	http://www.qsc.co.uk
GP-Version - Windows	http://www.qsc.co.uk/gpversion/gpversion.htm
JavaSoft JavaSafe - Solaris, Windows	http://www.javasoft.com http://www.javasoft.com/marketing/collateral/java_safe_ds.html
JSSL - Librarian - Windows	http://www.winlib.com
Tesseract - Lifecycle Manager	http://www.tesseract.co.za
British Aerospace - LifeSpan	http://www.lifespan.co.uk
Realcase - Multi-Platform	http://www.realcase.com
	Code Management
Perforce Software - Perforce Fast SCM System	http://www.perforce.com http://www.perforce.com/perforce/products.html
Data Design Systems - PrimeCode	http://www.datadesign.com http://www.datadesign.com/solutions
Merant - PVCS Synergex - PVCS	http://www.merant.com http://www.merant.com/pvcs/index.asp http://www.synergex.com http://www.pvcs.synergex.com
Qumasoft - QVCS - Windows	http://www.qumasoft.com
Inroads Technology - Rapid	http://www.inroadstech.com http://www.inroadstech.com/index_help.html
	Implementation Technology
Ultracomp - Red Box DuraSoft - Revision Control Engine	http://www.ultracomp.co.uk http://www.ultracomp.co.uk/products/redbox.html
Software Ever After - R-Sea-Yes - Windows	http://www.s-e-a.com.au
Lucent Technologies - Sablime - HP, NCR, SUN	http://www.bell-labs.com http://www.bell-labs.com/project/sablime
MKS - SDM-Implementer - AS/400 MKS - Source Integrity	http://www.mks.com http://www.mks.com/solution/sdm/sdm_products/imp.htm http://www.mks.com http://www.mks.com/solution/si

Table 4: Configuration Management and Version Control

SourceGear - SourceOffSite - Windows	http://www.sourceoffsite.com
SiberLogic - SourceTrack	http://www.siberlogic.com
Giant Technologies -	http://www.giant-technologies.com
Visual SourceMail - Windows	http://www.giant-technologies.com/sourcemail
Microsoft - Visual SourceSafe - Windows	http://www.microsoft.com http://msdn.microsoft.com/ssafe
Mainsoft Corporation - Visual SourceSafe - Unix	http://www.mainsoft.com http://www.mainsoft.com/products/visual/over.html
Starbase Corporation - StarTeam Windows, Solaris	http://www.starbase.com
Interwoven - TeamSite	http://www.interwoven.com
SoftLanding Systems - TurnOver - AS/400	http://www.softlanding.com http://www.softlanding.com/sourcecode.html
Burton Systems - TLIB - Windows	http://www.burtonsys.com
George James Software - VC/m	http://www.georgejames.com
UNI Software Plus	http://www.unisoft.co.at/Home.html
VOODOO - Macintosh	http://www.unisoft.co.at/products/voodoo.html
Sun - Forte TeamWare	http://www.sun.com http://www.sun.com/forte/teamware

Table 5: SCM Tool - Public Domain Free Software

Aegis	http://www.canb.auug.org.au/~millerp/aegis/aegis.html
CERN - CMZ	http://wwwinfo.cern.ch/cmz
CVS	http://www.cvshome.org
DVS	http://www.cs.colorado.edu/serl/cm/dvs.html
Inversion	http://inversion.tigris.org
jCVS	http://www.jcvs.org
Keep-It	http://www.keep-it.com
ODE	http://www.accurev.com/ode/index.html
PRCS	http://www.XCF.Berkeley.EDU/~jmacd/prcs.html
RCS	http://www.gnu.org/software/rcs/rcs.html
SCCS (free implementations)	http://www.cvshome.org/cyclic/cyclic-pages/sccs.html
TCCS	http://www.oreilly.com/homepages/tccs

Table 5: SCM Tool - Public Domain Free Software

tkCVS	http://www.twobarleycorns.net/tkcvs.html

Appendix B: Interviews

B.1   Introduction

As in most reports from The Association of Swedish Engineering Industries also this one contains interviews made with Swedish companies. The purpose of these interviews is:

- to get an up-to-date input to the project which, together with literature, build the base to the analysis and conclusions.
- to summarize some them in the report in order to provide some examples of current situations within Swedish companies.

We have no intention to give all interviews the same format in order to compare the interviewed companies with each other. The interviews are not comprehensive, but each focuses on some important issues from that interview, ranging from technical descriptions of tool interfaces to concrete descriptions of the current situation at that company.

All persons interviewed have had the possibility to review the final text. The interviews are not some official statements from the respectively organization. In each part the terminology have been used for respectively organization, which in some cases differ from the rest of the report.

Because both areas are under continues development and change in the companies, all the interviews are on the spots account.

B.2   Summary of the interviews

During the period October 2000 to August 2001 seven interviews have been carried out. Short outlines of the interviews are:

ABB Automation Products
- An example of a complete solution with many different tools.
- Discusses the management of hardware and firmware:
  - How the product volume affects the product structure. At large volumes the manufacturer installs the firmware, but at small volumes even the customer can install the firmware. This is reflected by the place of the "Firmware-node" in the product structure.
  - The version of the firmware is readable only from the control panel, and not as previously on a physical sign which gave problems.
- All software, even though located at many circuit boards, is treated as one product.

SaabTech Electronics AB
- Today it is a significant need to replace the current PDM-system, CAS, with a new generation facility for product data management;
  - One reason is that, within the near future, it will become hard to provide resources for operation and maintenance of current CAS. Digital will no longer support the platform;
  - Another is the wide interest for additional functionality plus a more user-friendly web gui.
- They have bought a new PDM-tool, Metaphase Foundation, which first will be implemented to fulfill the same functionality as the old tool.
- The solution is an "umbrella-tool" with Electronic's interfaces to the needed local IT-systems.
- Want an apparent support for the life cycle model.

C Technologies AB (publ.)
- Discusses the need for a better overview of the overall development and releases on the system level.
- The release area, implemented in an ordinary file system, does not cope with the requirements. They are in the middle of a pre-study gathering requirements from all development groups for a new solution.
- The "one tool solution" is probably not possible, but an "umbrella-tool" with interfaces to the local tools will be needed.

Ericsson general
- An example of a complete solution with many different tools.
- Corporate basic standards.

Ericsson Mobile Communication
- Discusses the importance of engineering support systems that not only benefit the business but also are understood and utilized.
- The development of a new, more user friendly web-interface on top of one of their PDM tools.
- An example of a step by step implementation:
  - phase one; manage non product documents only, e.g. meeting protocols.
  - phase two; also support BOM management including product documents.

Ericsson Radio Systems AB, Division Mobile System
- Operational PDM/SCM concept from a concrete project and how it works today.
- They have a good life cycle, good distribution of the work, good theoretical base, and "CM in place", but much is done manually depending on bad integration between the tools. E.g. manual transfer of requirements for the product to requirements for SW and HW parts.
- Lack of PDM-tools that work for all phases of the life cycle. Especially the early phases have bad support.

PrakTek

- The goal was to implement a CM framework for software development (i.e. building a software product). They concluded that they needed both a PDM tool and an SCM tool - integrated with each other.
- An example of an interface and integration between a PDM tool and an SCM tool (eMatrix and ClearCase), including a short technical description about the implementation.

B.3   ABB Automation Products

B.3.1   Interview, Malmö
Peter Nolte, Technical Product Manager

This interview gives a short description of the tools used to manage PDM and SCM. It also gives some experiences about product structure, level of traceability, when to update revision numbers, and how to manage firmware.

B.3.2   Company and product
ABB develops and delivers products and IT solutions for force measurement, control and protection in industrial processes and in power application. Products such as Controler 800M, Controler 800C, I/O 800, I/O S200, HSI Operate-IT etc.

B.3.3   Tools used
ABB in Malmö (former Alfa Laval Automation) previously used a system called "K-bas" (construction base) which was aimed at support for the developers during the production phase. For example, the tool stored information about how to build the system. This tool was old and is now replaced with standard ABB tools such as Documentum and Capri. Currently used tools are (list not complete), see also Figure 31:

- Documentum - central structured archives. Stores product information, e.g. document structure and part lists. Also stores project documents. Server in Västerås.
- Capri - (graphical) user interface to Documentum. Used by technical product managers.
- SCOT - provides views of parts of Documentum plus additional information. Used by, among others, product management and other ABB companies, accessing manuals, data sheets, product guides, etc.
- SAP R/3 - stores product and production data, such as subcontractors, stockroom, and prizes. Information about a product is transferred from Documentum when the product is to be built.
- Visual SourceSafe (VSS). Used for software. Also used for function specifications, test specifications, implementation proposals (IP), etc.
- Tracker. Database for error handling, proposal of changes etc. Different 'areas' are used for HW and SW.

One common problem with most of the tools is their bad user interface. This makes them hard to understand and it is almost impossible to really know what you can use them for (their functionality). As the result they are not used to the extent they should both in terms of number of developers using them and number of times used per developer.

Today most of the information is, more or less, manually transferred between tools. A more automatic transfer had better supported short development cycles and the overall process.

Documentum

Partlists of all compo-
nents included in the
product
Project documents

SAP R/3

Product and
production data

all source infor-
mation needed
for production

tools,
FM Templates
etc.

VSS

binaries

SW
specs.
IP

Tracker

CR

detailed descrip-
tion of product
being built

complete
product

TietoEnator produces docu-
ments, technical writing
Work in house or "at home"
using ABB's VSS directly

SCOT

easy access to part
of the information
stored i Documen-
tum

outsourcing.
E.g. Flextronics
use their own tool
"Baan".

customer

Figure 31   *Some of the tools used at ABB Automation Products*

B.3.4   Product structure

One important reason to use a PDM system is to manage the product structure. Figure 32 depicts a graphical view of a small example of such structure. Included are all parts in the entire product, independent of realized in hardware (HW) or software (SW), outsourced or developed in house.

A product structure may contain optional nodes. One reason may be that the production volume determines when and by whom firmware is loaded. Another may be whether a specific functionality should be realized in hardware or software.

Firmware (FW) is low-level software loaded on a circuit board. An example is software implementing the protocol on an i/o board. The same HW can be loaded with different FW implementing different protocols.

At ABB the volume of a product varies from >10000/year down to 100/year. The volume affects the optimal product structure. A CPU part for several system families may have a generic HW. The production volume determines when the firmware is loaded into the CPU HW. For a low volume the firmware is loaded late when packaged or even by customer. For a high volume product the firmware is loaded early, often by the manufacturer (e.g. Flextronics). I.e. it is important to have a flexible product structure.

When a product is sent to production a tree of concrete nodes, is specified, called a part list. It is important to unambiguously define what the product consists of. Each node, article/part, can be released in different ways: developed in house, outsourced, or bought 'off the shelf'. Irrespective of realization it always has an article number.

Figure 32 *A small example of a product structure. Contains optional nodes realized depending of high or low production volume.*

Figure 33 depicts an example of an article and what it consists of (in this case four parts, articles). Each part can, in turn, be complex and consist of articles.

Each article has a article number of its own - down to a certain level. ABB has full traceability down to the manufacturer of a circuit (only tested manufactures are used and specified in the SAP/R3). Some companies have traceability down to batch number of the used circuits. The 'same' HW has the same part number independent of manufacturer. Thus, it is impossible, in the product structure and in the BOM (bill of material), to see who manufactured e.g. a resistance (often more than one are specified in SAP/R3) on a CPU card.

B.3.5   Versioning and traceability

An article number is unambiguous and includes the revision number. E.g. 478.127.001 is an the first revision of an article. Next revision will be named 478.127.002. It is important to define what type of change to the article should require a new revision number. The main rules are:

- A change of functionality/interface implies a new article number.
- A change not modifying the functionality (but e.g. performance) may require a new (increased) revision number.

```
<article no> Product
    <article no>   PCB (bought from FlexTronic)
    <article no>   label
    <article no>   packaging
    <article no>   Firmware version x.x
```

Figure 33 *An example of an article and what it consists of. A part in an article can, itself, consist of other parts.*

Another rule is that the product revision is increased when anything within the product has been changed. This is not, however, always true. Examples are:

- Change of manufacturer of a resistance, still with the same characteristics does not require a new revision number.
- New layout without changing the functionality does not need a new revision number.
- A new revision of a software implementing the card logic requires a new revision. Note that even if the new software implements the same functionality a new revision is required.

These rules are based on experience. Changes that more likely change the behaviour (or often cause problems) requires a new revision. A conclusion could be that it is hard to guarantee exactly the same functionality when software is changed, and therefore SW changes always requires new revisions.

If changes are made in parts produced by other companies, ABB can always require exact information about changes and revisions.

B.3.6   Other versioning experiences

- Each circuit board has a bar code identifying the hardware *including* the software needed for the logic (not firmware). I.e. some SW is loaded into HW and together treated as HW only. This bar code is automatically read by all tools working on the board, e.g. adding new circuits.
- One SW code base is called ATLAS. An example of revision of ATLAS is 0.43-38-xxx, where xxx is increased for each build (daily build). All soft wares (generated from the same code base) is treated as one "product", see Figure 34. This implies that the firmware for one CPU may get an increased revision number due to changes in some other code not affecting this CPU at all. Sometimes this leads to confusion from the customers who ask what the difference is between their revision and the latest (and the answer may be "none").
- 10 years ago they treated circuit boards and their firmware together as an article and marked each board (visible from outside) with its article number, i.e. if the firmware was changed the label also had to be changed. This however, did not work out in practice. Customers (and service) forgot to change label when new SW were loaded which resulted in loss of traceability and confusion. Today the
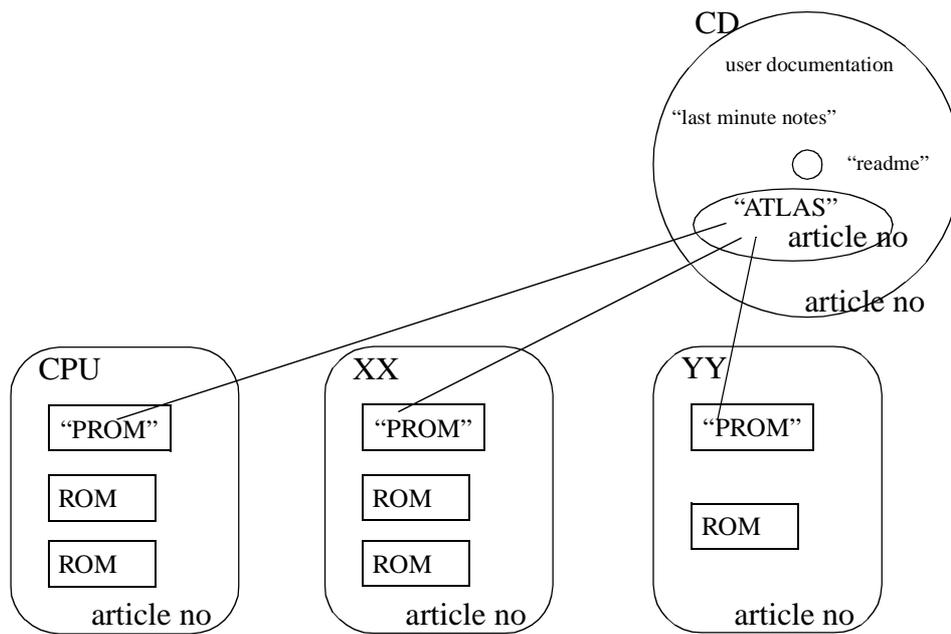
Figure 34 *All software is included in the same article, called ATLAS. Software is release on a cd, together with user documentation, etc. An image of the cd is stored in SourceSafe and in SAP R/3.*

revisions of all software is read by the system only (e.g. from a service terminal), accessing the software directly. In this way the correct revision is always retrieved. For example is all revisions checked when new software is installed.

B.4 SaabTech Electronics AB

B.4.1 Interviews Järfälla November 2000 and May 2001

This section is the results of the interviews with Pekka Lundström November 2000 and Hans Willebrand May 2001. Since the first interview some parts of the company have been moved inside the concern Saab or been sold to other partners.

B.4.2 The company

SaabTech Electronics AB develops and manufactures advanced electronics within the areas Electronic Warfare, Optronics, and Sensors. SaabTech Electronics (formerly CelsiusTech Electronics) is a business unit within Saab, northern Europe's leading high-technology company and employs 550 people.

SaabTech Electronics' main fields of expertise are systems know-how, optronics, microwave technology, precision mechanics and signal- and image processing. Within these fields our technological level is high relative to our international competition thus making us an attractive partner for cooperations.

The company consists of 3 divisions:

Optronics

The Optronics Division is the Nordic region's leading supplier of advanced optical and Optronics sights, chiefly for various types of missile and weapon sight. Among others, most Bofors weapon systems employ our sights.

Our areas of expertise are advanced optics, precision mechanics, electronics and system integration. We have our own manufacturing plant and can grind and polish lenses and apply advanced surface coatings.We currently employ 90 people in this division.

Electronic warfare

We are a leading supplier of Electronic Warfare equipment to the Swedish Defence Forces, with advanced radar warning systems and self-protection countermeasures in our product portfolio. The strategic technology base is concentrated around systems know-how, advanced electronics, signal processing and ECM (Electronic Countermeasures).

Within the Development department of the division, there are sections for Mechanical design, Electrical design, Software design and Test and Simulation. The Integration & Test section is also organized within this department, having the responsibility for final assembly and test of all deliverable items. We are currently 260 people in this division.

Sensors

The Sensors Division of SaabTech Electronics is engaged in radars, microwaves, antennas, signal- and image processing and stealth technology. We develop microwave sensors, microwave subsystems and products where these are important parts. Our customers are various business units within SAAB as well as external system houses and end customers.We are now around 200 people.

B.4.3   PDM product

General
Below is an overall description of CAS (Centralt Artikel- och Strukturregister), its role among the companies for technical administration, and its primary functions to fulfil this role. CAS is an in-house developed PDM system. CAS has links for all articles to theirs documents, changes, e.t.c. CAS originated in the middle 80s, and is run on a Vax/VMS-environment.
CAS is the company's main register for articles and structures and contains data of articles, articles structures, type lists (a list with approved articles which are allowed to be used under current requirements) plus a manufacturer and customer register. All articles and theirs structures "are born" in CAS and thereafter the information are distributed to adjacent tools, e.g. the company's MPS-system.

In CAS most articles have references to all documents, which are describing the articles. Furthermore the documents part-list and blank-lists are generated out of the system. Apart from these, CAS can also generate a number of reports/schedules.

Environment, system and authority
CAS is implemented on a VAX-computer and all functions can manages from a VT100-terminals. Information retrieval can be made by HTML-browsers on the company's intranet. The system is available for many concurrent users and is based on the database DBMS and program language COBOL.

CAS has an own access system that makes it possible to control the access to the different functions for each user.

To register data in the system the user must be an user on the Vax-computer with the right to enter/update the information in CAS.

In order to read the information in the database only requires access to a HTML-browser. The same function also exists with a VT100-terminal but the user must be a user on the vax-computer.
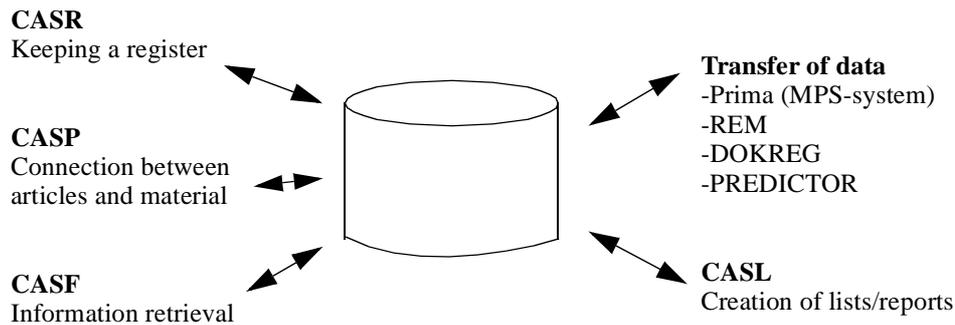
Figure 35 *System functions*

Functions
CAS is divided in the following different parts: CASR, CASF, CASL, CASP plus functions for transfer of data to other system.

CASR/CASP.   Update/registration of data in CAS is primarily performed in CASR. Certain information exclusively used by production is registered in CASP. It was cheaper to implement this function in CAS instead of in the MPS-system. CASR contains functions for maintenance of articles, part lists, type lists, customers, and manufacturers. The registration and update of data in CASR is carried out by operators within current design organization. They undergoes education before they receive the authority to maintain these records.

CASF   have two GUI, either HTML-browser or VT100. For users that only read information in CAS it is highly recommended they use the HTML interface on the company's Intranet. CASF contains functions for search of information. One block is used in order to search information as e.g. metadata. The other block enables searching for free text. In order not to overload the company database, a separate database has been created to be used to for free text searching. This database is updated once a week through transfer of data from the main CAS-database.

CASL   contains functions for printing of different types of lists/reports. The authority to print is regulated by needs to minimize the load of the system. The main operators normally carry out the order of lists. Which information that shall be enclosed to the order of a list is described in the guiding document for respective list.

Transfer of data   CAS is the company's master system for metadata of the articles, i.e data "is born" in CAS and then transfers to other systems. Transfer is made to the following systems:

- Prima - The company MPS-system
- VeriBest - CAD/El
- REM - The company's system for delivered systems and spare-parts.
- PREDICTOR - Program for reliability calculations.
- (DOKREG - The company´s register for documents.)

B.4.4   SCM product

For software development the CVS (Concurrent Versions System) client/server is selected both for embedded and not embedded software systems.

B.4.5   Structuring of product data - the principles

All parts (system, subsystem, product, HW- and SW-parts) in a structure are named articles. Each project creates its own structure and are reusing so many articles from finished and ongoing projects as possible.

Where the SW is placed in the structure depends of if it is an embedded or not embedded SW.

An article, not on lowest level, is described of its structure. The structure describes the parts of the article. The articles are identified with an article number. For each article number the information that describes and/or defines the article (information that is needed for its procurement, production, test, service, servicing, continues development, etc.) are linked.

The information that are linked to the article can consist of:

- occasional data
- document references
- decided changes (consists of document)
- and for composite articles its parts one level down (consists of).

Of the informations above only the document is an object of its own.

Also certain files linked to articles or document is counted to be an object. Files exists of several types:

- included as an article at delivery (e.g. SW)
- shall/can/should be used at production of an article (e.g. CAD-files or files that is used at processing printed circuit cards)
- ”the original” for a document.
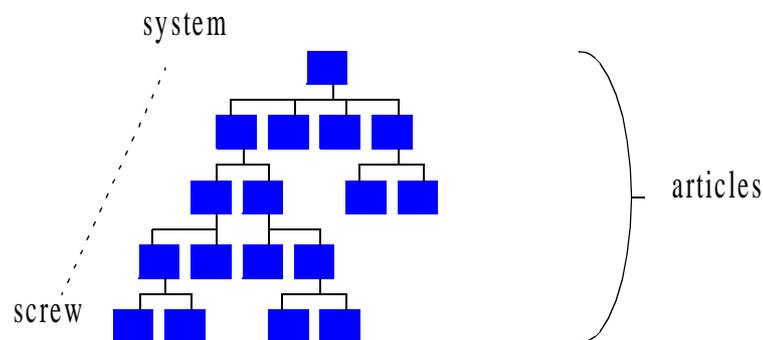
Figure 36   *Articles*

In order to simplify the management of articles they are grouped to a package named "unit". These "units" consist of a collection of articles for which it is decided to handle them as one unit with reference on follow-up and status accounting of carried out changes/updates.

By spliting the product into "units", it is possible to control the configuration of the product in an easy way.

Within a project a number of objects have been decided to be under Configuration Management. These are named CI (Configuration Item).

In many cases the "unit" is a CI.

The article is the object where other information is linked to. An article can be everything as from a complete system down to simple screws. In a structure articles can consist of hardware , software and manuals that are needed for its operation and maintenance. Depending of the type of the article different information for the article are defined and specified.

System and product are two special cases of the object 'article'.

- System equals the top article in the hierarchical structure within a project that we are responsible for or deliver to the customer.
- A product is a system component part, i.e. an article on the level closest under system.

Document
A document is a package of information that has a identity and version plus a defined validity (document status).

Document manages on the same ways regardless which type of article it describes (hardware , software or manuals). Also document that manages as stand-alone object, don't linked to article, e.g. report, protocol, etc. manages in the same way.

File
A file is a package of information, which is storage on a media under an identity. A file is connected to a development environment and a current tools.

"Unit"
"Unit" is specified on the levels in the structure there decision has been taken to account made changes/updates. The account is made only if the change decision (change order) tells it. This means normally to the levels where requirements have been allocated. In daily speaking these are called main units, embedded units, subunits, spare parts, etc. An unit has always a sign and is the lowest level where it is possible to track individualls.

CI
An article that is define as a CI is not manage in a different way from the rules that counts for current article.

B.4.6   New PDM product

General
Today there is a significant need to replace current CAS with a new generation facilities for product data management - CAS II. One reason is that, within the near future, it will become hard to provide resources for operation and maintenance of current CAS. E.g. does not Digital support the plattform any longer. There is also a wide interest for additional functionality plus a more user-friendly web-gui. To provide the extended functionality plus improved availability, it will be more effective to completely re-develop the system.

The development of CAS II will be carried out by the project Prokon. Prokon is an R&D-project ordered by FMV and is carried out by Saab Dynamics, Kockums Industrier, and SaabTech Electronics.

CAS II will be based on the commercial tool Metaphase Foundation from SDRC. Several tools have been evaluated but TeamCenter has been selected.

CAS II will be implemented stepwise. Ultimately the tool will be used by all users as a common information system for our products and projects.

Concept PDM/CM-system
The vision with a central PDM/CM-system is to create an integrated system (routines, rules and tools) in order to establish and maintain the configuration for the article during its entire life cycle (development, manufactoring, maintenance and administration). Initially CAS II will implement the CAS functionality. The configuration of an article is necessary information needed in order to describe an article complete enough to manufacture, use, administer, and further develop after the end of the initial development. The configuration and the data can be managed in different development environments with accompanying IT-support, for example electronics, mechanics, software, and user (customer) documentation.

Based on experiences within the PDM/CM-area at the company during 1980-2000 and trends on the market, the overall system architecture will be as depicted in Figure 37.

The main PDM/CM-system is the overall system that provides support for management of the total configuration of the article, which also includes management of links between the different divides of the article under its entire life cycle. Cooperation with local IT-systems takes place on reasonable level for different activities. Superior control is required both on identity and version for articles (static CM) and thereto linked documents normally dynamic CM is applied.

Storage and transfer of data in "neutral" formats are important aspects that must be fulfilled. The need to store data for a long time (20-30 years) on well established and stable standards is a major requirement.

In the local IT-system all the articles and related data needed during the product development process are created and managed. When an article version is released the necessary result of development should be controlled in the main PDM/CM-system either directly or by references. This is often a subset of all the information that is managed in the local IT-system.

The development of the main PDM/CM-system shall enable application for the company's generic life cycle model, see Figure 38.

The requirement for the implementation of the PDM/CM-process is the life cycle model that has been decided at the company. Coupled to the model the project management decides which functional and physical articles that shall be under CM, how overall structure shall be designed, and which document shall be the basis to respective results for an edition/version of the article (baseline). The PDM/CM-system supports processes that enforce configuration through maintaining relations between versions of articles, documentation and changes on these during the entire life cycle.

In order to support the PDM/CM-concept the main PDM/CM-system should at least provide support for:

- *Document management*; support to e.g. create, identify, register, file, present, search, reuse, etc. Also workflow for e.g. reviewing/approval and release should be supported
- *Article management*; meaning support to create, identify, register, present, search, reuse, connect information plus structure management,
- *Configuration management*; support for identification, control, reviewing plus status accounting of object that manages of the PDM/CM-system.
- *Change control*; including:
  - support in order to create and manage basis/document from change proposal to approved change order and the incorporation of the descision
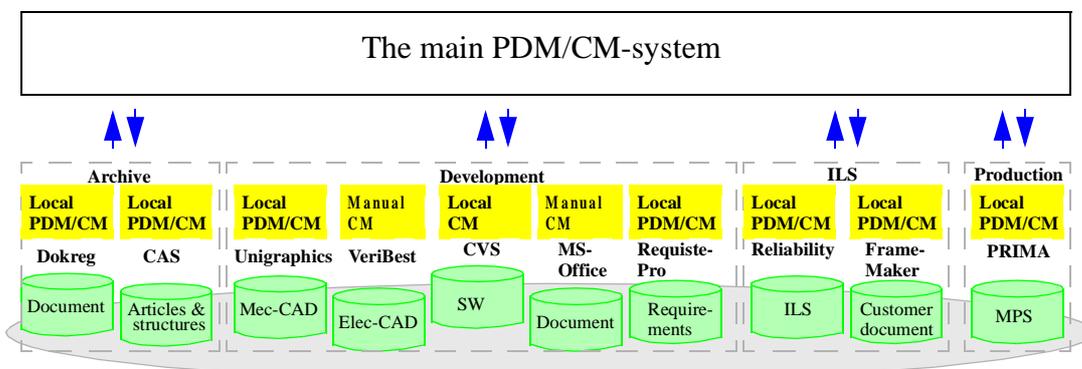
Figure 37 *Overall system architecture*

- workflow for review/approve/release
- status accounting, e.g. change proposal created, approved, implemented during development/production/delivery/customer.
- *Work flow*; support for to create processes (work flow) as e.g. a release process for documents.
- *Program Management (WBS);* "Program Management" enables e.g. for a user to relate a WBS-activity to an object that manages by the PDM/CM-system. This gives ability to status accounting of document, which is ready, started, waiting for review etc.

Apart from the above support it should also be possible to integrate the PDM/CM-system with other IT-system by efficient interfaces.

For integration to local IT-system three types of Local IT-system can be identified:

- embedded PDM/CM-system e.g. UG-manager for management of MCAD-data produced by Unigraphics
- local CM-support e.g. the archives system Dokreg and CVS for SW.
- manually CM-support, for instance manual naming of directory structures and files in the NT-environment e.g. for document created by MS-Office.

Export/import of information between internal and external PDM/CM-system/IT-system shall normally be distributed through the main PDM/CM-system. Format of the information can both be application dependent or application independent but should in most cases be application independent.
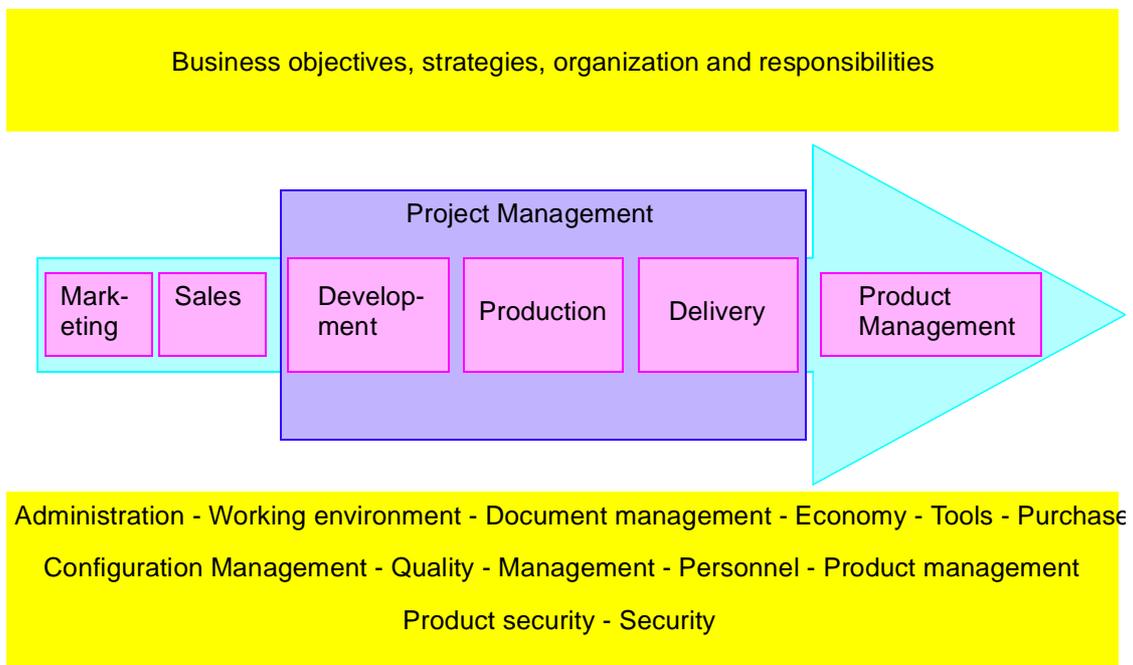
Figure 38   *Life cycle model*
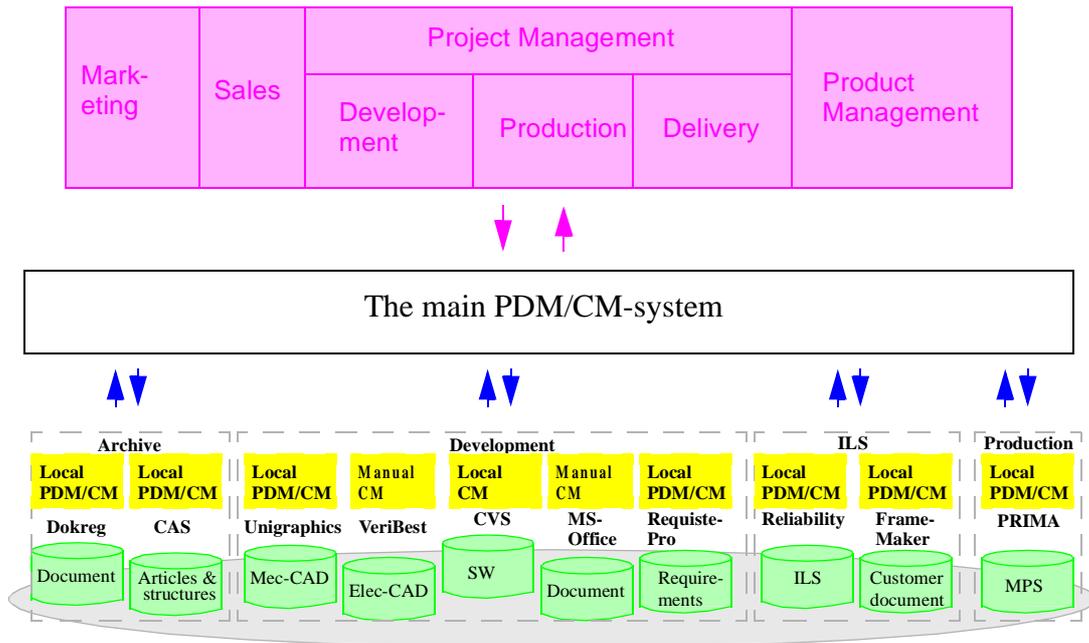
Figure 39 *Overall system architecture and its relationship to the life cycle model*

Implementing the company's life cycle model in the main PDM/CM-system the support for configuration and change management should be used for all phases. For the initial phases, Marketing and Sales/ Projecting the primary needs of support is document management. In succeeding phases Project management, Development, Production, Delivery and Maintenance we will found the main needs of support for e.g. article and structure management.

The initial phase of the article development process is primarily the creation/management of documents. In the succeeding phases, beginning with the Project Management, the WBS-activities are created and related to objects managed by the main PDM/CM-system. This includes the creation of the development plan as an activity and system requirements analysis/construction as another. Analysis/construction happens normally at the lowest implementation level after which the system construction is realized in the main PDM/CM-system in a function structure.

The system design and thereto linked document/information are reviewed/approved/ established by using the workflow support in the main PDM/CM-system.

The realization of the system design is carried out in respective development environment (locally IT-system) and "is released" to its parent PDM/CM-system at least at establishment of a formal version.in the local IT-system. Examining/approval/establishment are carried out with the workflow in the main PDM/CM-system.
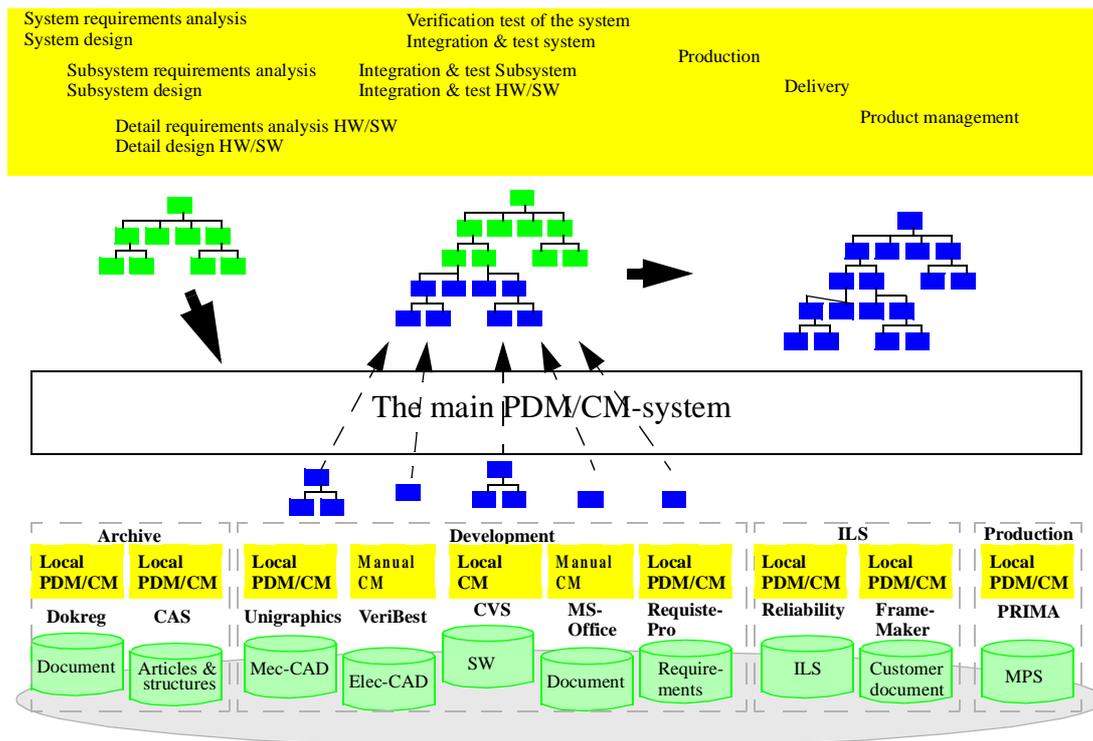
Figure 40 *Overall systemarcitecture and its relationship to the development phase during the article development process.*

The system design structure is released from respective development environment to the main PDM/CM system, gradually building up the realisation structure (dark or blue boxes in Figure 40).

The release of an article is made both for use in the line organization and in the project. The release in the line organization controls by setting the metadata "**pr**o**d**uct **r**elease" (PDR) on the article and in the project by status marking in the WBS.

For the production, all required information is collected in the main PDM/CM-system and transfered to current MPS-system. It is during this and succeeding phases the configuration on manufactured article individual begins to be registered and/or updated in the main PDM/CM-system.

During the maintenance phase of the article the total information that is produced for the system/the product plus for article individuals with s/n on LRU-level are stored in the main PDM/CM-system.

At bigger modifications/upgrades of existing articles a new project is normally started and proceeds through the development process. Old system are phased out after a formal decision.

B.5 C Technologies AB (publ.)

B.5.1 Interviews summer 2001, Lund

Per Beremark, quality manager
Bengt Löfstedt, product and project manager
Cecilia Broberg, manager document systems

B.5.2 The Company

C Technologies AB (publ.) is an established, high-tech Swedish company with cutting-edge competence in digital camera technology, image processing and digital pens. The company's main product, C-Pen®, is available in three different models and has received a number of international honours for best IT product. C Technologies' technology is an attractive addition in other products through license or OEM agreements. OEM sales are primarily aimed towards manufacturers of mobile phones and PDAs and secondarily towards the computer hardware industry as a whole.

C Technologies AB has approximately 130 employees with just over 60 working in Research & Development.

The C Technologies Group includes subsidiaries Anoto and Wespot and has a total of approximately 340 employees.

B.5.3 Organization, routines, and tools

R&D (Research & Development) at C Technologies is divided into several groups, see Figure 41. Most of the groups consists of 5-8 persons with the notable exception of Software with 25 employees. Besides these, there is 'Operations' that actually produce and deliver the products, but this is outsourced to other companies. During the development phase, each group manage their own data using their own tools. A project manager is responsible for the entire development, i.e. the result from all groups together.

This interview is about how C Technologies manage release and change management in a heterogeneous development environment with no common PDM tool. We also discuss the requirements on a future 'umbrella' tool to better support the process.
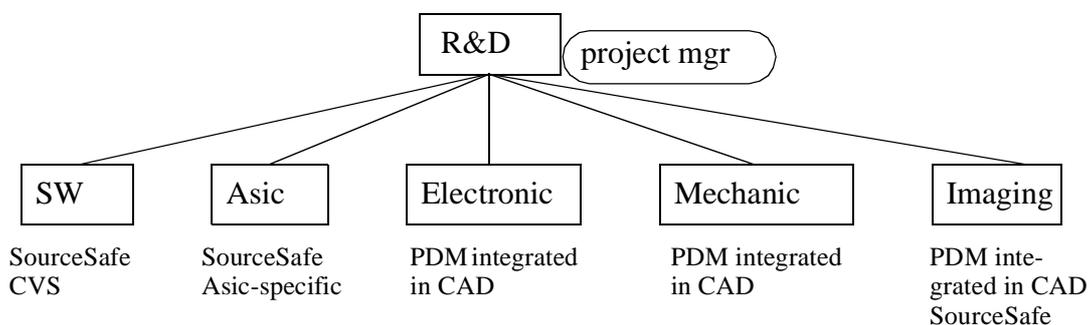
Figure 41 *Organization. During development each group manage their own data using their own tools for SCM and PDM.*

The company was founded in 1996 and has lately grown dramatically facing all problems related to that. E.g. are most of the routines followed more like "traditions" rather than a documented process. Each developer has a lot of knowledge and is skilled within his/her domain, but there is no documentation of how, in which order, things should be done (processes, methods, routines). Especially, each group have their own routines (often on an individual level), and there are no routines common for all groups.

Release management
There is no system/tool managing the release information. Instead an ordinary file system is used, i.e. a directory is created containing a structure in which the groups can put their release documentation, see Figure 42. The name of the directory defines which release it is. All file names includes the release number, i.e. they are renamed when copied from their development storage to the release area. Prototypes, alfa, and beta releases are managed similarly.

Consequently there is no tool support for the release process, or any integration with the tools used within each group. Most documents stored in the release area are pdf-files or other common (simple) formats, but also (sw) binaries are stored. The BOM (bill of material), for example, is stored as an Excel document.

No CAD-files are copied to the release area. Instead these documents are "printed" to pdf and delivered (copied) to the release area. Similarly no source code files are copied, but labeled in SourceSafe.

Within each SCM and PDM tool the release is labeled to enable traceability to the sources. A release has one label used (globally) in all tools.

SW	Asic	Electronic	Mechanic	Imaging
SourceSafe CVS	SourceSafe Asic-specific	PDM integrated in CAD	PDM integrated in CAD	PDM integrated in CAD SourceSafe

Release area
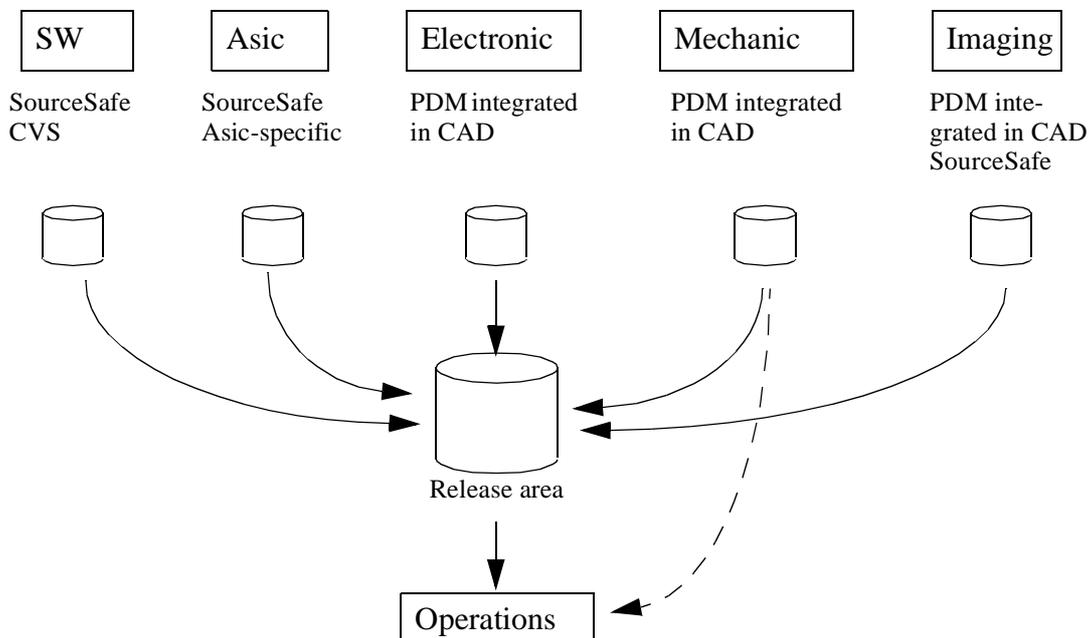
Operations

Figure 42 *Each group delivers (in standardized formats) to a release area. Production is managed by Operations.*

Different routines in all groups

The Mechanic group has the best established routines. Within their group they use a Configuration Identification similar to Ericsson, using revisions, states, etc. From this internal control they have no problem to deliver correct pdf-files to the release area. Only pdf-files are stored in the release area. CAD-files (original format) are also sent directly to Operations for production.

The Electronic group uses their own CAD-system. All documents are stored in Source-Safe, but they lack a history description (until recently). Discipline (rather than tool support) is required to be able to track differences between releases. Drawings (designs) often have revision history written (manually) on the drawing. Pdf-files are created and delivered to the release area.

The Software group uses SourceSafe from which they deliver for release. Sometimes, however, the miss to deliver which makes it harder for the project manager to follow the progress of the project (for the software part). The delivery process is: (1) build the correct binaries, (2) copy them to the release area and (4) rename them to a name including the correct release number. All sources and binaries are labeled in SourceSafe.

One recent project uses CVS instead of SourceSafe due to performance problem with SourceSafe. This specific project follows the XP development model [Beck99] which advocate short cycles and frequent releases to improve awareness within the project and to the market.

Change management

Another important process that often suffers from common tool support is change management. Today the software group uses a special tool called Wreq in which the developers themselves register new requirements. The support group, however, do not, which results in a delay for these requests to be registered. It is thus important for the project and product leaders to have 'big ears' to, in time, know what has to be done.

Summary

In the current solution the project managers lack an overview of the entire product. It is especially hard to follow the development between releases. Only the release area is understandable for the project manager who is not familiar with the different, specific, PDM and SCM tools used by each group.

Mechanics and Operations really want better control, not for their local development but for the release process. Their complains actually started a pre-study. Right now it is hard to know what is the latest version, is it tested, is it released (or is the latest release out of date?). C Technologies also have many subcontractors, and the communication with them has to be more formalized.

B.5.4   Ongoing pre-study

Cecilia has recently started a pre-study of the requirements for a common PDM system, including both process and tool. All groups are now supposed to list all their require-

ments on such a system. It will be very interesting to see what this leads to, but, unfortunately, it will be after the print of this report.

Future?
Since C Technologies is in the middle of gathering requirements for the new processes and tools we do not know yet their future plans, we can only guess what will happen. Right now they consider it impossible to only have one common PDM/SCM tool for all groups. Each development tool has their specific requirements and the developers are used to the 'local' tools used. Possible, however, is to have a common tool also, i.e. in addition to the local tools. Such 'umbrella tool' could provide the global view to the managers, some control functionality, and support for a common release process. It could also be possible to integrate it with other existing tool such as 'Axapta' which is C Technologies business system, managing components, subcontractors, information about purchase, etc.

B.6   Ericsson – Corporate Basic Standards

B.6.1   General

This document states some of the general concepts (Corporate Basic Standards) used within Ericsson regarding product and document administration. It gives a background to the other Ericsson interviews included in the report. Corporate Basic Standards (CBS) defines the rules for the handling of products and documents within Ericsson. This document describes specifically the handling of:

- Definition and identification
- Versioning
- Variants
- Product structuring.

This appendix also describes the main PDM tools, PRIM and GASK. They are used for product and document administration within Ericsson. Both are based upon CBS and are in-house developed systems. The first versions of the two tools were available in the early 80's.

This text gives a brief overview for the situation spring 2001.

In order to understand Ericsson Corporate product and documentation systems, you must understand the underlying concepts. These are much older than any existing computer system. These concepts have been used since early 1930's.

All pictures within this document have been provided from CM training material from Ericsson Mölndal (038 13-FEA 202 262 Uen Rev A).

B.6.2   Product

Definition and Identification
Within Ericsson all objects handled within the corporation or delivered to customers are named products. These can be physical or non-physical. Thus projects, processes, software, binders, components, cables, rubber boots, and exchange sites are all identified as products.

Since more than 60 years Ericsson identifies all products in the same way. All products are classified based upon a central corporate classification schema named ABC-classes. The classes (about 1500) are identified with three letters, e.g. AXE. I.e. AXE is actually the three first letters of the product identification, see further example in Figure 43.

```
┌─────────────────────────────────────────────────────────────────────┐
│                          ABC-main groups                            │
│  A    Systems for Telephone exchanges    M    Raw and bulk materials │
│  B    Packaging concepts                 N    Network systems,       │
│  C    Function and program products           elements and plants    │
│  D    Telephones                         P    Micro electronic       │
│  F    Immaterial objects                      components             │
│  H    Telesignalling systems             R    Electrical components, │
│  I    Plants and not regular products         complex                │
│       and documents                      S    Engineering            │
│  K    Telesignal devices                      components, details    │
│  L    Production, installation and       T    Cables, cords,         │
│       information aids                         conducting wires,     │
│                                               winding wires,         │
│                                               printed wires          │
│                                          U    Military and           │
│                                               industrial systems     │
│                                          Z    Transmission systems,  │
│                                               installation and       │
│                                               equipment              │
│                                                                      │
│              Figure 43    Product Classification System             │
└─────────────────────────────────────────────────────────────────────┘
```

Figure 43   Product Classification System

The 17 main classes from the product classification system are broken down to form three letter classes. Some examples of ABC classes are:

- AXE   Public telephone exchanges systems
- BKB   Battery units, e.g. used for batteries on Ericsson mobile phones
- FCP   Project activities
- LZT   Printed Matters, Leaflets, Catalogues
- CAH   Software Unit
- ROA   Printed board assemblies with miscellaneous spacing

The three first letters are combined with three digits to form the product type, e.g. AXE 106. To the product type we add the sequence number. This is the full product number.
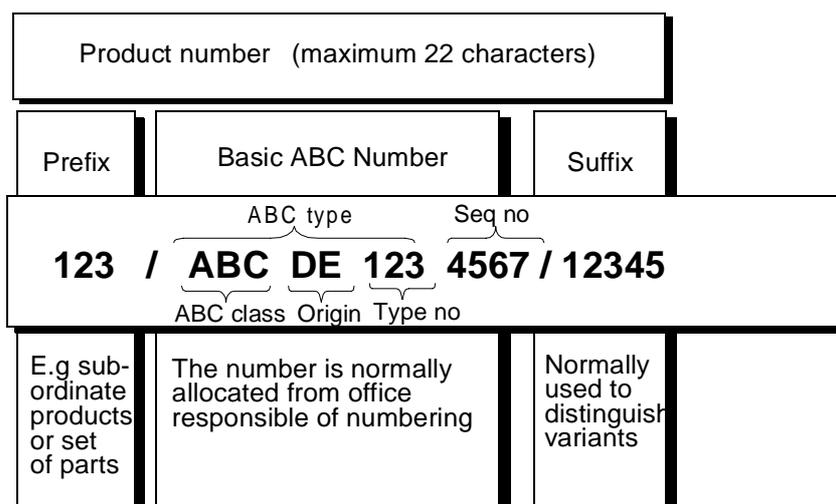
Figure 44   *Product Number*

Prefix, suffix and country origin can be added to this basic product number. In the Figure 44 you will find the complete description of a product number within Ericsson.

Examples of product numbers are ASB 501 04, ROF 137 5054/2, CAA 111 1305 or EN/LZB 103 04.

Hardware and software are handled alike from this perspective. Software source code is today handled as a document tied to a product. This implies you will get a lot of documents (e.g. thousands of files will result in thousands of documents and a lot of manual work to be done in PRIM) related to the specific product. Today's trend is to treat the software source code as information object (not a document) related to a product. This implies you may under certain circumstances archive the source code in the SCM tool without managing documents in PRIM.

Ericsson does not use the term component, part or article.

Versioning
Ericsson product versions are named product release states. This is shortened to R-state for product versions. There are three R-states:

- R1   Interchangeability is stated in the documents, usually in the Product
            Revision Information, PRI.
            (limited interchangeability)
- R1A   Interchangeability is indicated by the R-state.
            (regulated interchangeability)
- RA   The latest version replaces all earlier ones.
            (simplified R-state)

Ericsson differentiate between *administrable* (e.g.R1A, R1, RA) and *preliminary* (e.g. P1A) product versions. Administrable versions are meant for delivery to customers. Preliminary version is used during the development process to identify different versions of a product. Interchangeability does not exist between preliminary R-states. The type of version is identified in the R-state of the product. Ericsson has products of standard nature, e.g. screws. These do not have R-states at all.
Within Ericsson an unspoken rule is to document every change.

The Product Number and R-state together form the Ericsson Corporate Product Identity, see Figure 45.

Variants
When no relationship between two products in terms of compatibility can be identified but the products are related in terms of function then a new variant is designed. Variants use the suffix at the end of the product number. number An example is that the same type of trunk line is delivered to different countries, where differences lays at low functional level. These different trunk line boards are then variants from the original trunk line. E.g. ROF 137 5452/1, 137 5452/2, 137 5452/3 etc.

| Product number   (maximum 22 characters) | | | | Product version |

| Prefix | Basic ABC Number | Suffix | R-state |

ABC type   Seq no

123 / ABC DE 123 4567 / 12345   R1A

ABC class Origin Type no

| E.g sub-ordinate products or set of parts | The number is normally allocated from office responsible of numbering | Normally used to distinguish variants | Indicates version of a product |
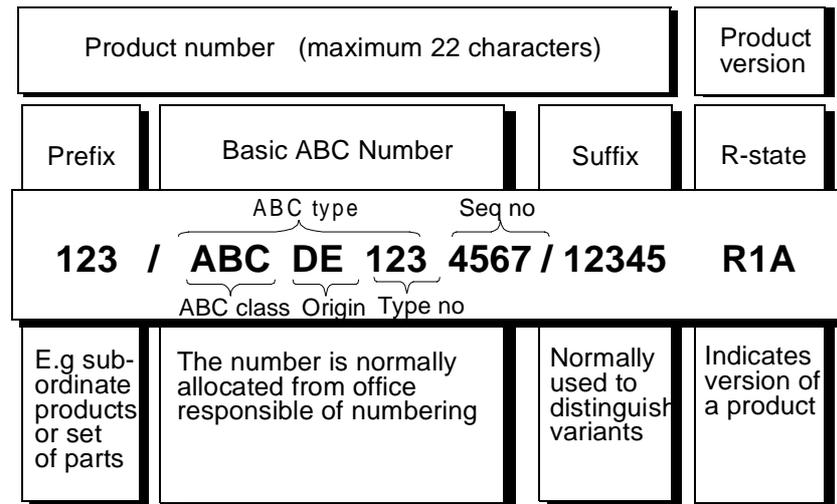
Figure 45 *Product Identity*

Product Structure

Ericsson product structures exists in four variants. All structures are hierarchical. The different structure variants are;

1. Functional structure, "consist of" specification of a functional product. Points to the functional products at the level below. It can also point to implementation products, e.g. hardware or software. Usually mechanical items are tied into high levels in the functional structure.
2. Realization structure. An implementation product, e.g. Printed Board assembly (PBA), magazine, cables, always have a manufacturing structure. The normal BOM is included here, also other information needed to produce the product is included
3. Project Structure. This structure can be linked to top level of the system structure.
4. Sales structure or Commercial Structure. A sales object structure defining products that are sold

For some products no breakdown is possible e.g. due to the fact that the product is purchased. Then a product specification is written to state the product data needed for purchase and maintenance agreements.

For larger product structures e.g. AXE you use one system with a product structure (here also called source system) to create applications. This is done with the source system as a base but with the product configured (e.g. program volume).

The result is configured products for specific applications (could be one or more countries, one or more customers). The latter applications are called application system. This way of handling larger system and their configuration is named source system - application system. The advantage is that you can create several applications from the same

source (hence the name) without a complete redesign. The disadvantage is that these source systems are very complex.

An application system specifies the implementation products included. But the final configuration to specific products to be used, e.g. exchanges, is still not done.

Site management is generally not included within Ericsson. This is due to the fact that the operators keep their own structure and systems for their sites. There are situations where Ericsson does handle sites, but this is an exemption.

In the Time To Customer flow the Sales Structure identifies products to be delivered. These uses implementation structures for breaking down products selected.

The handling of structures varies of course within a corporation handling everything from larger turnkey system to consumer products. The product life cycles varies from 10-30 down to months.

B.6.3   Document

Definition

A document is defined as information stored on some kind of media. Thus all kind of information is included here. E.g. descriptions, software code, machine code for test machines and product structures in PRIM. In short, everything that is not a product is a document.

As with the product identification all documents are numbered according to a central corporate classification schema named Decimal-classes. Since more than 60 years Ericsson identifies all documents in the same way. The classes are identified with three to

Document number (maximum 36 characters)					Language edition	Document version
Registering office	Class prefix	Decimal class	Individual number	Variant code	Language code	Rev-state
XYZ	**123**	**/123 45 -**	**ABC 123 45**	**-A1**	**Uen**	**A**
Only at documents made for a limited use	Identifies sub-documents	Indicates type of information	Product that the information refers to (product docs) *or* sequence number from registering office (general docs)	Indicates document variant	Indicates language edition	Indicates version of a document

Figure 46 *Document Identity*

Example of Information Structure

TELEFONAKTIEBOLAGET L M ERICSSON				PIPBO2A V03.42 1998-09-11					
A CODE	DOCNAME/TITLE	DOC NUMBER	PRODUCTNUMBER	R-STATE OF THE PRODUCT					
				R1A	R2A	R3A	R4A	R5A
0	PRODUCT REV. INFO	109 21-			/	-1,A	-2,A	-3,A	-4,A
0	DOCUMENT REV. INFO	109 24-			/	/	-1,A	/	/
1/2	MOUNTING DRAWING	1078-			A	B	D	E	E
1	DESCRIPTION	1551-ROF 654 2918			A	A	A	A	A
1/2	CIRCUIT DIAGRAM	1911-		A	B	D	E	F	
1/2	TRUTH TABLE	1/191 12-			A	B	B	B	B
3	CODE GEN INFO ERE 190 93-1003		---- ---- ---- ---- ----						

A-CODES

			1997-06-18 S	1095-ROF 654 2918/1 UEN
PRODUCT NAME	FUNCTION DESIGNA	DOCUMENT SURVEY	DESIGN RESPONSIBLE
PRINTED BOARD ASSEMB	DATOR		EMW/FY/K
EMW/FY/K			

Figure 47 *Example of an Information Structure*

five letters in front of the product number. E.g. 1551–ASB 501 04 is a general   description.

Examples of document numbers could be
- 109 21- ASB 501 04, Product Revision Information;
- 131 32 -ROF 137 5054/2, Manufacturing Specification;
- 102 62-CAA 111 1305, Design Specification;
- 1050-EN/LZB 103 04, Description.

Versioning

Ericsson differentiate between *administrative* and *preliminary* document versions by means of the Rev-state. Administrative versions are meant to be registered, filed, stored and distributed to subscribers. Preliminary documents are used during updates.

This is shortened to Rev-state for document versions. Rev-states comes in one variant only, letters. Examples of a version for a document are A, B, C, D, E, F etc., see Figure 46.

Variants

Variants keep traceability to original documents. The versioning rules give flexibility for document variants and for revising published documents with traceability. For a published document that needs correction, the following Rev-state of the document might be published as well. To keep traceability, a digit is added, e.g. A1, A2. Thus versions of documents can form more complex trees of information for those products that have been maintained for a longer time.

Structure

A document is always tied to a product in an Information Structure. This gives a document overview for each product, where you can see which document Rev-states that are valid for each product R-state, see Figure 47. The 1095- is available in the product register, PRIM.

A document for one product can be tied to several products information structure.

B.6.4   Legacy PDM Systems

The two main PDM tools within Ericsson for the handling of products and documents are PRIM and GASK. PRIM is the central product register (all metadata and product structures) and GASK is the central archive (file and document archive). Both PRIM and GASK have been in operation for many years. Development for new PDM environment is ongoing. In the near future a change is more than possible. One example is that some ClearCase vobs today are defined as approved archives in PRIM.

Clients for these tools exist for PC-windows, Unix and the web. PRIM is the database with all of the information.

PRIM

PRIM is an abbreviation for Product Information Management System. It is based upon the Ericsson basic concepts.

PRIM consists of four different parts: Product data, Document Data, Information Structure, and Product Structure.

Product data
Here a product gets its product identity and product category (determines the use of R-states). You will have to choose your ABC-class and type. Then the product sequence number is assigned and the complete identity is registered.

Within Product basic data the product responsibility is also assigned, e.g. design responsible and release responsible. After product data is registered in PRIM the product is available for other systems, e.g. fault reporting in the central modification handling system (MHS). Functional designation and product name is also registered.

Foreign product numbers are normally re-numbered with an Ericsson number, but it is possible to register foreign product numbers. Ericsson has several other systems to manage external product numbers, e.g. ELIZA for suppliers.
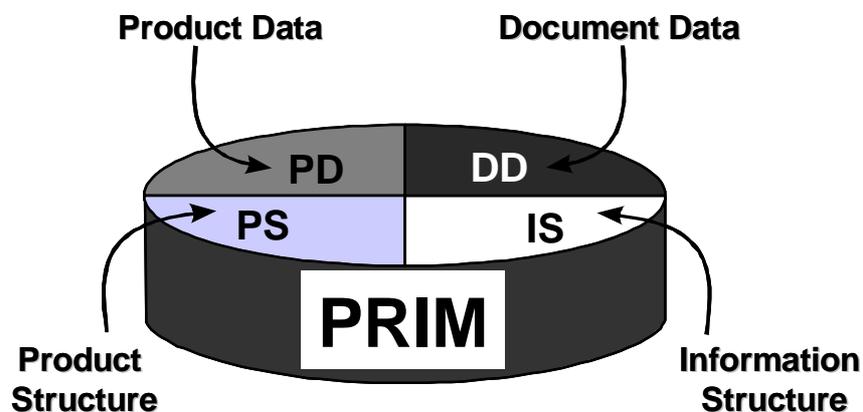
Figure 48   *PRIM overview*

After the product data is registered the product is available to be specified into other products.

Product status gets registered here. There exist four main tracks:

1. Design Status, DS
2. Production Status, PS
3. Product Release Status, PR
4. Restriction Code, RE

Restriction and exemptions codes are not described here. A picture describing the later parts of a product total life cycle shows the use of product status:

The codes in Figure 49 are used for all products functional product, software product and hardware products. These codes are further explained below.

Design Status, DS Design status is assigned from '–' to '4'. The 'DS –' gets registered automatically when the product number is. The following rules apply for progress:

• DS1 Design started.
• DS2 The design is complete and the preliminary documentation is available for restricted application.
• DS3 The product documentation, except test documentation, has been reviewed, approved, registered and archived
• DS4 All the product documentation has been reviewed, approved, registered and archived

In the status DS4 the product can be released, see PR codes below.
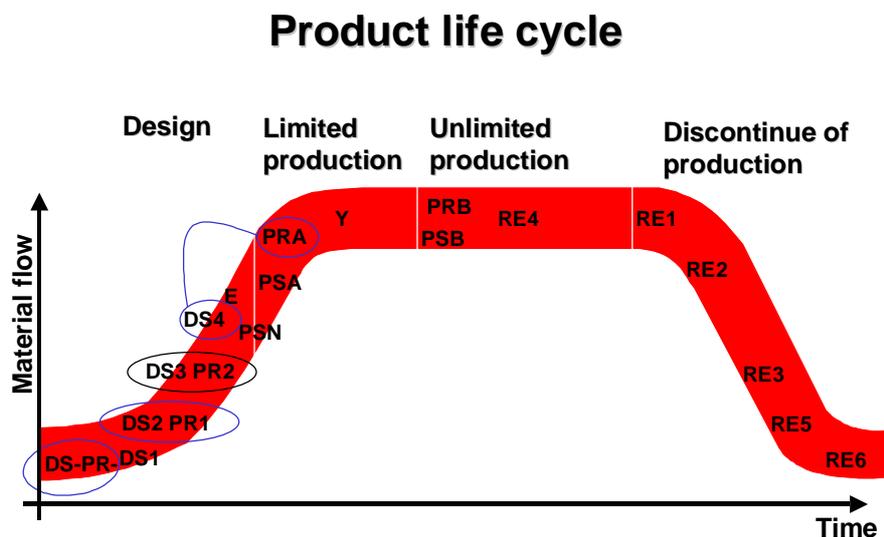
Product life cycle

Figure 49 *Product Life Cycle*

Production Status, PS   If used this status indicates whether the product has a production approval. The concern is here that the product can be produced with the correct quality and cost. Without the PS A code the product cannot be released.

PS shows that he manufacturing has tested the new product in the production lines.

Product Release Status, PR   Product Release status is assigned from – to B. The 'PS -' gets registered automatically when the product is registered. The following rules apply for progress:

- PRA   Product is released for a limited number of markets.
- PRB   Product is released for global use

There are some other codes, not explained in this document. To release a product you must have the product at DS4 and PSA (if used) levels.

Document data
Here the document gets registered with its identity (including Rev-state). Author is also assigned as well as archive and if the document is stored and approved.

During development preliminary product documents may be stored in a project archive. Then the documents will be stored in the corporate archive, GASK. When a document is stored in GASK, automatically document information will be stored within PRIM.

The management of preliminary product documents differ within different organizations within Ericsson. One example is that when the project is getting closer to release the document gets transferred into the official archive GASK. This is then valid both for textual documents and code/machine information for production plants.

After the registration here the document can be included in the information structure basic data.

Information structure
Here is the heart of the whole PRIM system. Here you link all document versions to product versions. Amongst these documents is the structure specification pointing out the lower subordinate products.

Here your register your document number and versions as the projects progress. When the product release is performed (PRA) a freeze is made in PRIM and all documents to a product version get frozen. The only way to change a version is to make another release with a new product R-state. In the information structure you can also follow the progress of a product, you can see how documents and versions change over time.

Product structure
Functional and manufacturing structures within Ericsson are created by an application within PRIM. You specify the structure tied to a product one level at a time. You can

then break down on a number of levels to get a complete structure generated for all product levels.

Manufacturing structures (BOM) are designed according to central rules. Here more information than in functional structures is added. The production information is identified here, e.g. mounting instructions for components are included. Boards are structured with the information needed to identify components, their location and the mounting case applicable on a PCB. The information structure will identify the other needed drawings.

All structures are identified as documents in the information structure overview. A functional structure is identified as decimal class 131 61-, a manufacturing specification as 131 32- and product specification as 1301-. They are all treated as documents and do for example have document rev-states.

In the structures you can choose if the system should pick a product according to the compatibility rules or be specific identifying the exact version.

External interfaces
PRIM is linked to several other systems. It is more or less the heart of many Ericsson product information systems that forms a huge network all linked together.

Most important is the link to the central document archive GASK.

Another important link is to the modification handling system where all fault reports are registered and tracked.

GASK
The central document archive GASK has been an archive for all released documents since more than ten years. A replacement is planned although nothing final has been presented yet.

This archive is used by all organizations across the world.

GASK contains all types of files, e.g. textual documents, CAD files, software code information (e.g. PROM information). A number of security codes are used to get the right user authority so that the product information is secure.

With an organization like Ericsson it is valuable to have information accessible across the world, on-line.

All organizations normally have a project archive where development information is kept. It is later transferred over to the GASK archive for final archiving

B.6.5   ClearCase
In the recent years, the Rational SCM tool ClearCase has become more or less a standard within software development within Ericsson. Today more and more handling of

SW is done in ClearCase. Semi or fully automatic links are usually built into ClearCase to make a connection to PRIM and GASK. Especially since software within Ericsson more and more is developed incrementally the features from a SCM tool is needed.

There are also a corporate initiative to build a common platform for ClearCase within Ericsson. The goal is to support implementations of local ClearCase installations by offering a common platform on top of Clearcase. This platform is to include a variety of needed adaptations to get Clearcase started. Examples are connections to PRIM. This common platform is named Ericsson ClearCase (ECC).

One conclusion that can be made is that within the SCM tool the software code is treated as an object itself. But traditionally, as described above, Ericsson treats the code as a document or in other words as an attribute to an object. To archive a lot of source code files, the files are stored within a container treated as a document.

B.6.6 PDM
In Ericsson efforts are today made to establish Metaphase as a PDM system covering the full product life cycle.

Since this is not in place the existing product register and document archive, PRIM/ GASK, have the role as a central PDM system. Regarding SCM systems they are not involved to PRIM/GASK other than delivering product data information.

B.7   Ericsson Mobile Communications

B.7.1   Interview, Lund 2000-10-12
Magnus Miegel, manager engineering (2000-10-12),
Mats Ohlsson, (2001-08-10)
TTM Engineering Platform at Ericsson Mobile Communication (ECS) in Lund.

This interview discusses the importance to have engineering support systems that not only benefit the business but also are understood and utilized. We also discuss some current work to improve the situation by developing new, more efficient, tools.

The development of mobile phones (so called terminals) is geographically distributed at several sites in the world. Engineering functions like software, mechanical and hardware engineering are mirrored at all sites, e.g. in Lund.

B.7.2   PDM tools
Different groups use their own specialized PDM tools to manage their local development, often integrated with the development environment. They are responsible to get the needed support in terms of processes and tools. Sometimes this is achieved by global solutions in common for many groups, sometimes it requires 'home made' solutions. Some examples are: The Electronic group (EDA) uses systems managing their components in terms of CAD data and data about purchase and contractors; The Mechanic group (MDA) has a system to manage revisions of CAD files, etc.; The Software group uses ClearCase to manage revisions and releases of source code.

Within ECS several global solutions exist for many domains and phases in the product life cycle, e.g. supporting requirements management. The support systems for overall BOM (Bill Of Material) control towards factory are PRIM, GASK, ESTER, and others, which are described more in detail in appendix B.6, "Ericsson – Corporate Basic Standards". The rest of this interview is mainly about BOM management.

Current situation
PRIM and GASK are used after production release. PRIM is used mainly for the view 'As designed' and SAP R/3 is used for 'As manufactured'. None of these systems are used during the early development phase, at least not as much as intended. According to the PDM group at ECS this is unfortunate. A better use of these tools, also during development, had been beneficial for all parts. Instead local tools are used and registration in global tools is done late in the development process.

Ad hoc uses of PRIM and GASK results in limited control of versions and their contents. When the product is released all production documents are stored in RPIM/GASK, but during early phases, before production, and to some extent also for work done between releases, this information is often only known locally. Other design data is not stored in PRIM/GASK, but managed ad hoc. The project manager thus has a big responsibility to retrieve and send the correct information to the producers and contractors, especially for early prototypes, before correct revisions are registered in PRIM.

PRIM was primarily designed to manage the production of large and complex products like AXE switches, rather than mobile phones. Complex and rigid rules define the process, e.g. how to release the product, which requires a lot of manual administration and control. These rules also put high demands on the project leader and makes the process too slow to cope with the short development cycles used. It can also be hard for the developers to see what is in it for them, and therefore they do not use them, even though the truth may be that they have gained using the tools. Another reason not to use the tools is an unfriendly user interface.

Of course, there are also reasons not related to the tools or processes used. An example is the general fear to give away unfinished work, even for reading only. One main functionality of a PDM tool is to provide awareness to all developers and managers, which is best achieved if documents are checked-in frequently and made accessible. It is thus important to be able to 'check-in' work under progress and make it clear that it is in an unfinished state.

Improvements in progress
TTM Engineering Platforms at ECS is (right now) developing a new graphical (web) user interface on top of Metaphase, called 'metaDoc'. Actually metaDoc is built on top of CDM (Collaborative Data Management), which is a Ericsson specific layer on top of Metaphase supporting 'corporate basic standards' e.g. the rules mentioned above and all identification rules explained in B.6, "Ericsson – Corporate Basic Standards". Also CDM is under development and only parts of the final functionality is currently provided.

The introduction of metaDoc is made in two phases. In phase one metaDoc manages non product documents only, e.g. meeting protocols, general specifications, etc. Today there are 900 users and about 1200 documents registered. The users comes from across the organization. Next phase will also provide support for BOM management including product documents. To be able to launch phase two, both CDM and metaDoc must first implement support for structure which is not yet fully provided.

It is important that everybody gain business value when a new system is introduced. It is almost impossible to get acceptance if some users get it worse (e.g. more to do) in order to make it easier for other. All must get benefits from a new system/solution.

other interfaces	Unigraphics	metaDoc
CDM - Collaborative Data Management		
Metaphase		
Oracle		

Figure 50 *Tool architecture*
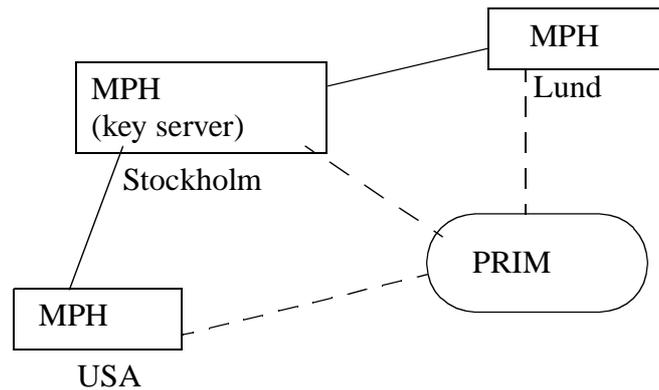
Figure 51 *Architecture of many distributed Metaphase servers and one central PRIM server.*

Future plans
The phase 2 of the PDM program has been initiated and is supposed to support BOM control. It also includes a re-implementation of CDM in order to make the rules less complex and more easy to understand and use, i.e. adapt the BOM management to the short development cycles used.

TTM Engineering Platforms actually does not want to develop tools by themselves. Initiatives such as metaDoc is taken only because they could not find something similar at the market. If a vendor provides an interface good enough, they will probably use that instead.

It would still be beneficial to integrate the overall PDM tool with the local tools used within each group. There are ongoing tests of an interface between MetaPhase and Unigraphics, and, as described in 6.4, "Possible integrations", there is also work on interfaces between ClearCase and both Metaphase and eMatrix.

A possible global architecture for a future tool could be as depicted in Figure 51, with PRIM still existing as a backbone for distributed MetaPhase servers.

B.7.3   Product modeling
ECS uses two different ways to model their products: (1) complete product structure, and (2) product (rule) engine. Often a lot of different products and product variants have to be managed, even though only some of them finally reach the market. To create one product tree for each product should be ineffective so instead a 'product engine' pro-

duces the products, as depicted in Figure 52. A 'super BOM' contains all components. The 'product engine' then defines a lot of rules of how to combine these components to build legal products. Some rules defines which components technically can work together, while some rules are more market driven (a color may be reserved for a specific product). New products can be tested by adding rules to the engine. However, it is more complex to manage rules compared to manage relations.

B.7.4 Traceability

All released products are registered in PRIM, i.e. the entire product structure is registered. Strict bound configurations are not used, since this had resulted in too many versions, especially on the top level (i.e. the release number of the product does not exactly specify the revision number of all of its components). Full traceability is instead achieved through a generic configuration plus a 5-letter code that really specifies its contents.

The main reason not to use bound configurations is the large amount of manual administrative work in PRIM bound configurations require. Another reason is the need to be able to modify the product without changing its revision number. Customers are sensitive for new revisions and it is hard to sell a product with an old revision number, independent of what the difference to the latest revision really is.

The drawback of using generic configurations instead of bound configurations is lack of full traceability. Sometimes the only way to find the correct revision, e.g. to fix a bug, is to 'follow PRIs' (Product Revision Information).

B.7.5 Change management

Today there is no tool support for global change management. The software group has developed a tool called 'Fido', which supports the entire change request process and bug-tracking. The tool makes it easier to find all products affected by a change request. A bug reported in one product may also exist in other products (e.g. a bug in a (revision of a) platform common for many products), and the change request created due to the bug therefore lists all affected products. Additionally the responsible for these products are informed through the tool.
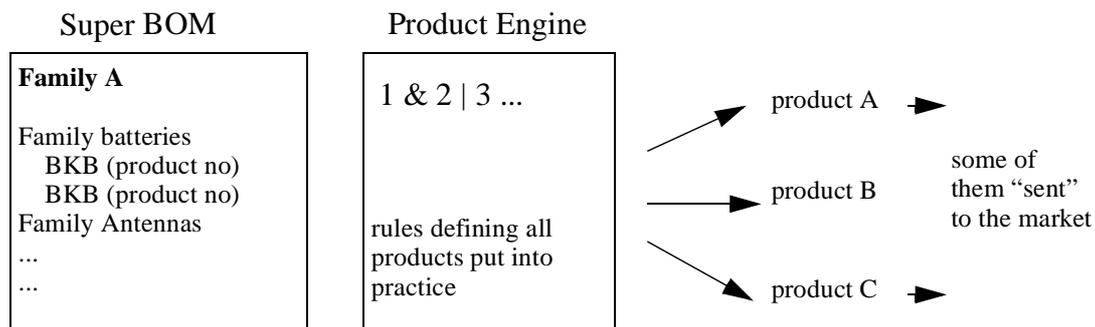
Figure 52 *Modeling the products by rules describing the products. The rules do not define all possible realizations, only those interesting.*

Fido is also integrated with the SCM-tool used (ClearCase) and is part of the freezing process of releases.

The management of changes to a BOM, ECN/PCN (Engineering Change Notice/Production Change Notice), is performed by writing, so called, PRIs (ECN), or to issue exceptions (PCN).

Figure 53 depicts the change management process. Fido manage the requests during the entire process and makes it easier for HVM to find the persons and products affected of a change.
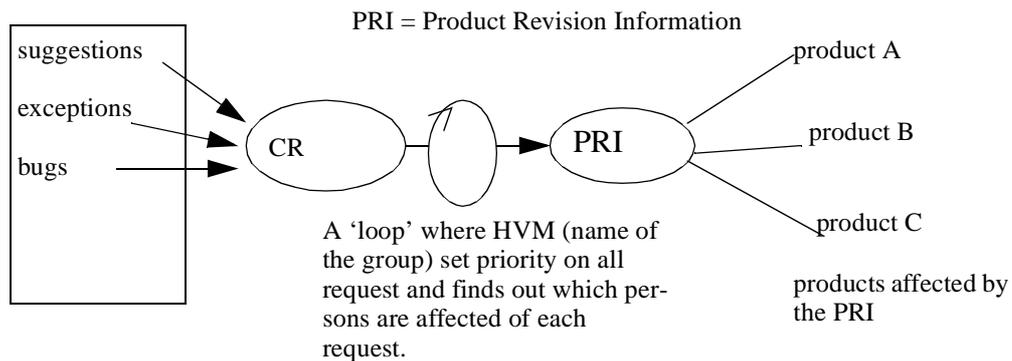
Figure 53 *change management process for both permanent changes and exceptions. HVM is the CCB with people from e.g. global customer service, production, and designers.*

B.8   Ericsson Radio Systems AB

B.8.1   General

This interview has been done with an existing project at Ericsson Radio systems AB, division mobile system PDC systems. The project chosen was the phase 10 project for PDC mobile telephone systems. The main goal of the phase 10 main project is to deliver a packet data solution to the Japanese market (this functionality is often shortened PPDC, packet data PDC) during 2001. The project is being developed within the Mobile Systems Division in Ericsson Radio Systems AB. The interview was done with the CM Manager Marita Berg, and it took place during several meetings from October 2000 until May 2001. Editorial changes have been made until August 2001.

Due to reorganisations within Ericsson many of the units mentioned may have been altered.

The purpose was to describe how the project utilises support systems for operative functions. The focus is upon the support from PDM and SCM systems. The interview covers the product life cycle from customer requirements until delivery to customer, here named Time to Market, TTM.

The project is a large development project. Organisations throughout the world are involved, e.g. Sweden, Germany, Japan. The main project group consists of 32 persons! The numbers of involved products are huge.The development includes both hardware and software often using incrementally design methods. The integration issues are very complex.

The interview will describe how the project has a good understanding and descriptions of the product life cycle. The configuration methods used are as well up to date and
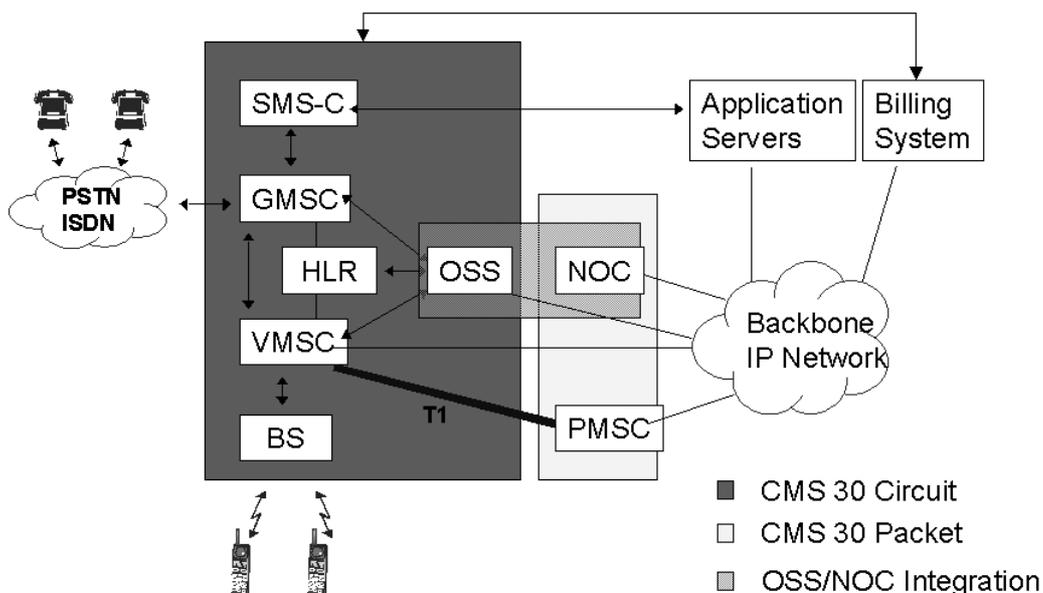
Figure 54   *PPDC Network Architecture*

competitive. Ericsson has a traditionally stronghold with the Corporate Basic Standards. These are a firm base for product and document handling (see appendix B6).

The interview also concludes that the total number of tools in combination with few interfaces and standards used for communication in between, shows how the work today still is dependent upon manual procedures. One example is the handling of change requests.

B.8.2   The Product
The PDC mobile systems with the PPDC functionality is described by Figure 55:

The picture above shows how the public telephone network (PSTN) and data networks communicates with the gateway exchange. Two more exchanges VMSC (voice handling) and PMSC (Packet handling) make up the main products in the PDC mobile systems. Please note the mobile phones below connected to the base stations (BS).

The product consists of both hardware (HW) and software (SW). Software handling is different for old traditional exchanges (e.g. GMSC) and newer operating support systems (OSS/NOC). Regarding documentation (DW) the phase 10 project will delivered DW in elelctronic form only.

Product Structure
Two product structures exist for the PDC phase 10 products, a functional structure and a sales structure. The functional structure will be linked directly to the implementation products. Since this interview deals only with development we will focus upon the functional structure which is drawn below.

CMS30	
SSB	
MSC	Mobile Switching Center
OSS	Operation and Support System
RNM	Radio Network Manager
PMSC	Packet Mobile Switching Center
CHSe	Charging Server
HLR	Home Locating Register
SCP	Service Control Point
SCE/SMS	Short Message Center
INSA	Intelligent Network Service Application
NOC	Network Operational Centre
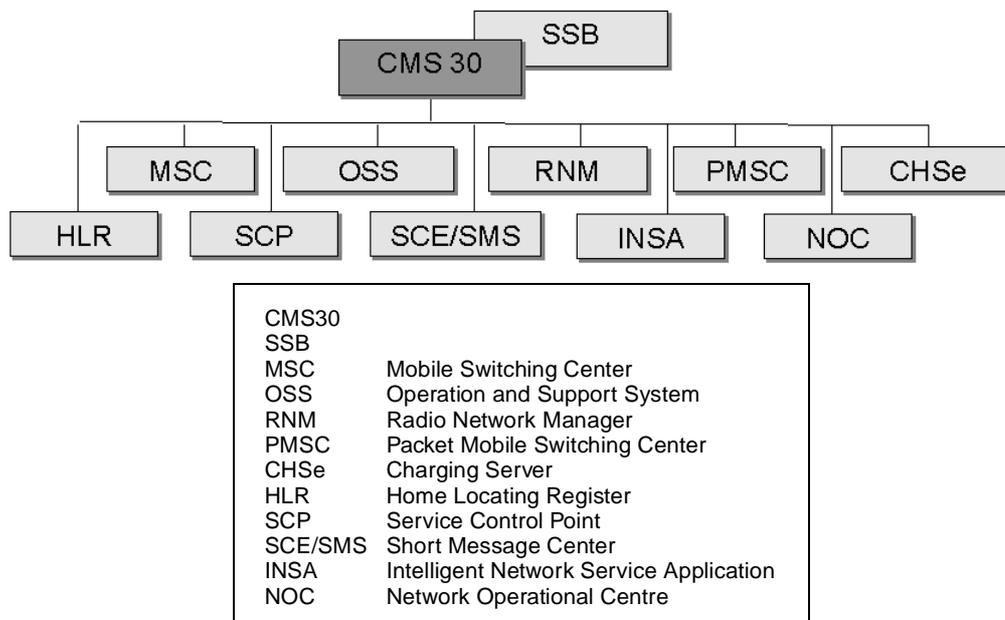
Figure 55   *Functional structure for PDC systems.*

Taking the Mobile Switching Centre, MSC, as identified above, for further structure breakdown of the functional structure gives us the Figure 56:

The structure described above can be broken down further, at least 5 more levels. The functional levels will be broken into the implementation levels of the product structure (the lower levels). The implementation levels contains all the realisation products. These realisation products will implement the functions described in the functional part of the structure. The realisation products are Software and hardware.Sometimes functions are realised with only SW, sometimes with hardware and software. Mechanical assemblies will be included in the structure as separate from the basic functions modules (e.g. processor, telephone and packet data).This since the mechanical parts does not have a direct connection to system functions. Parts of the final customer sites are not included in this structure (e.g. batteries, special cables, installation products (houses, antennas etc).

The product structure will be according to the Ericsson Corporate Standards.Please refer to appendix B6.

B.8.3   Organization

This chapter will describe the different organizations and subprojects within the PDC phase 10 project.

Requirement Management
The customer talks to the local Ericsson Company, with the short designation NRJ. Here the local product management will be responsible for the discussions and agreements upon contract issues..

NRJ then communicates to Sweden and the PDC main project level requirement coordinator. The coordinator is a representative at main project level for the product management in Sweden. The requirements will be assigned to one or several of the three major product lines within PDC. These three product lines are PDC (voice functionality), PPDC (packet data) and other systems (e.g. OSS).
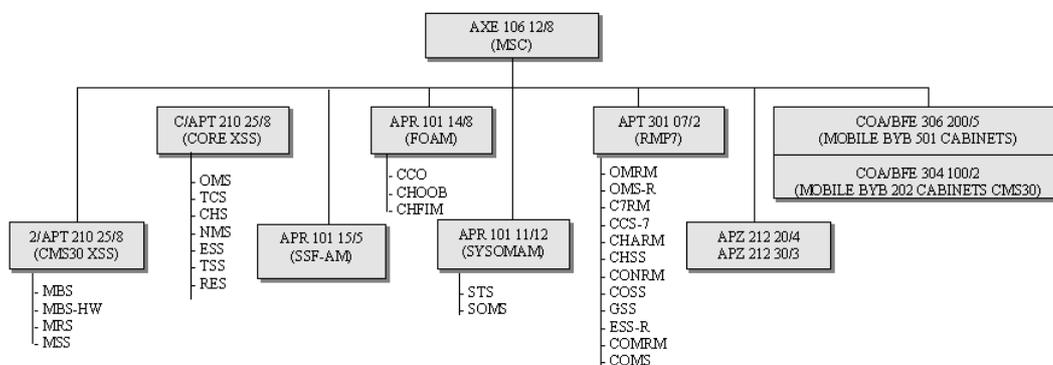
Figure 56 *Functional Structure for MSC.*

Figure 57 *Project Organisation PPDC*
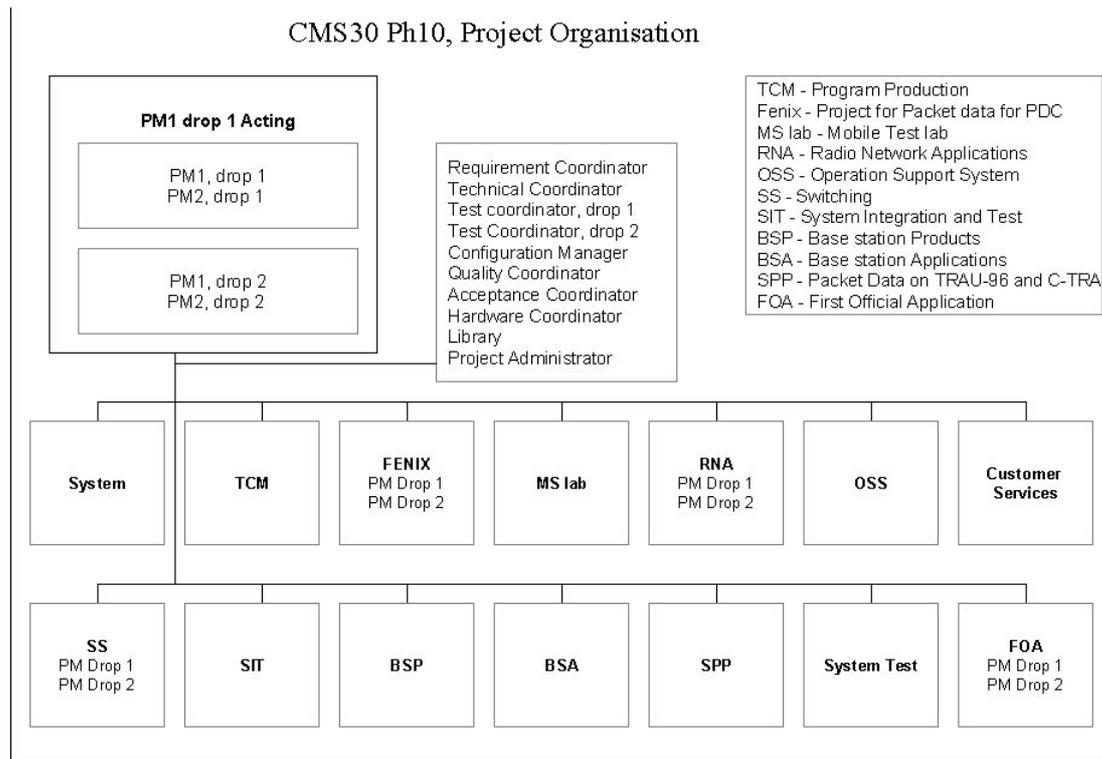
As an indication of the number of requirements there are 57 original main requirements in the phase 10 main project. One of these is broken down to 89 in the next level below. The lower level requirements are specifying more detailed criteria needed, requirements for e.g. the packet data handling in order to achieve the correct product. As the development continues more requirements will be identified.

The project life cycle phase is about 18 Months. The goal is 12 Months, but the customer wish is to get a project delivered within 6 Months.

Project Organization

The phase 10 main project is organized in a hierarchical manner. The project is organized as describedll in Figure 57:

The development units involved are not within PDC product line, but within other organizational parts of Ericsson. The project exists thus across organizations within the whole Ericsson corporation. This is common within larger Ericsson projects. The project is developed in Sweden, Japan, Germany and Finland.

Main project managers and coordinators take responsibility for the project in whole. The different co-coordinators has the responsibility to co-ordinate the different parts.

Especially interesting in the project organization are the subprojects:

- TCM, test configuration management that is responsible for supplying. The other design subprojects with AXE Test beds for Function Test.
- SIT (System Integration and Test) subproject that is responsible for integration of the Radio Network products and Packet Data.

These two subprojects have a very difficult task to gather all information and distribute the correct version to test. Especially since system test is done in Japan and function test in Europe.

Apart from the 14 subprojects there are 9 coordinators and 3 more project managers.These number shows the importance of integrated information systems giving accurate up to date information. And we have not even counted all other involved persons in the project.

The first exchange delivered from the project to the customer is handled through the development project, it is called first office application (FOA). The other supply services are handled through the roll out project (not included in the development project). The project ends with the decision for product release, within Ericsson named general availability (GA, for more explanation about GA please refer to separate appendix B6).

A project with a incremental approach within each subproject, mixing hardware and software, both consumer terminal and larger systems and being distributed across the world will represent one of the most complex configuration management situations possible.

Maintenance
Although not in focus, a brief description over the maintenance phase is included here.

The product is handed over to the local company (NRJ) after product release (GA). Service, 1st and 2nd line support, are normally handled within the design organizations involved in the subprojects described earlier. Thereby development, maintenance 1st and 2nd line support are placed within the same units.

Product Corrections are all delivered through ongoing development ("phase") projects. Thus no separate fault corrections releases are handled.

B.8.4   Time To Market Flow
From a high level viewpoint the TTM flow can be described as in Figure 58;

As seen in the picture, the TTM flow spans from customer input (several other input exist in reality) to the handover to TTC (Time to Customer) and maintenance & Support flow. The handling of Market activities in parallel to design is today common in the industry. But in this special case this situation is not at hand since this is a single customer with one single contract. This interview will cover system from product management over to design and finish at the handover to TTC. Marketing tools are not covered
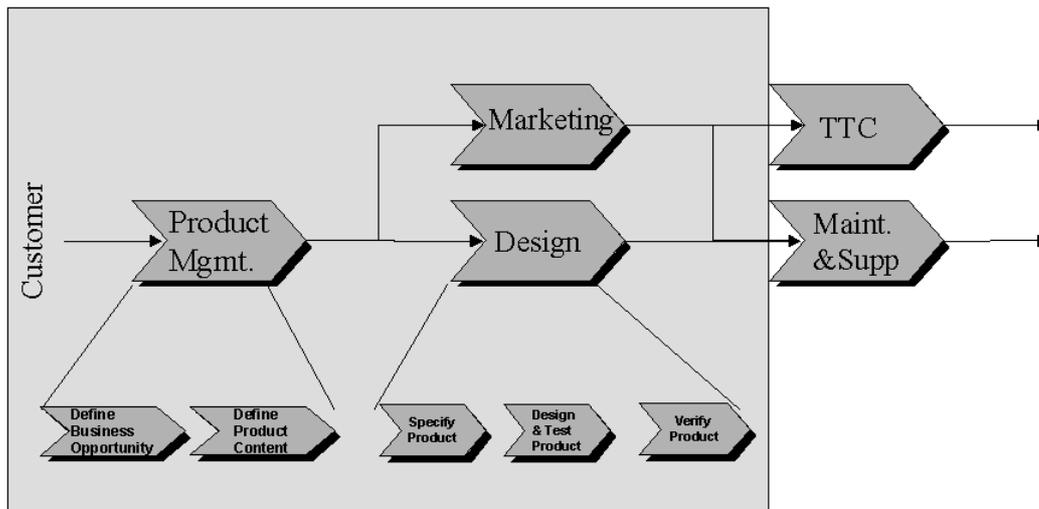
Figure 58 *Time To Market Flow for PDC systems*

in this interview. The product management and the design processes have been broken down to illustrate the next level of processes to be supported.

This product life cycle is well understood within the project and is an advantage for implementing system support.

B.8.5 CM Methods

The CM methods within the phase 10 project are well defined and understood. The local tools supporting the methods are not state of the art or integrated. This will leave much work up to manual handling and a risk for more faults.

The configuration management methods used within PDC can be summarized with the following three pictures (Figure 59-61). The baseline concept always documents the four corner stones, i.e. configuration, quality, deviations and decision. Thus the baseline is not a mere freeze of a configuration but much more.

A further explanation of baselines below in Figure 59 shows how these are documented. PRIM is the central Ericsson product catalogue where all products and linked
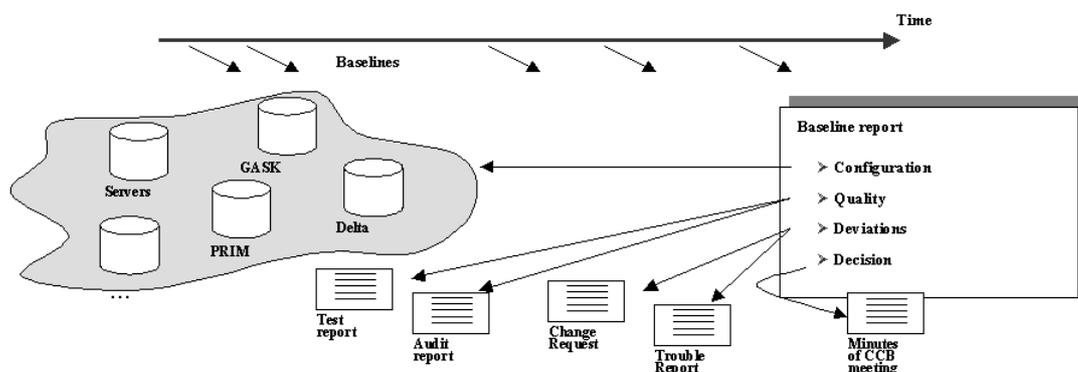
Figure 59 *The baseline concept*
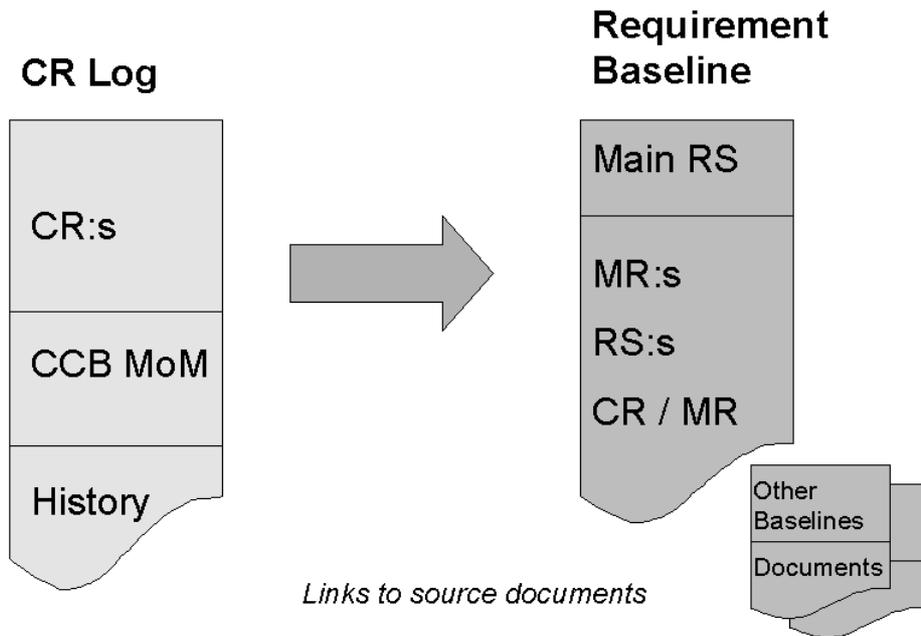
Figure 60 *Change request and how they are linked to baselines*

documents are identified and released. GASK it the central document archive (which for Ericsson includes software). For further information about the Ericsson Corporate Standards for product and document handling, PRIM and GASK please refer to appendix B6. Delta is a project archive tool used during development, especially for the development of AXE.

After the establishment of a baseline formal control shall be applied (please refer to ISO 10007). Figure 60 illustrates how the change request concept is used to track changes to baseline changes (in this case the requirement baseline).

The CR (Change Request) log is linked to the referred baseline, thus always keeping formal control up to date and complete.
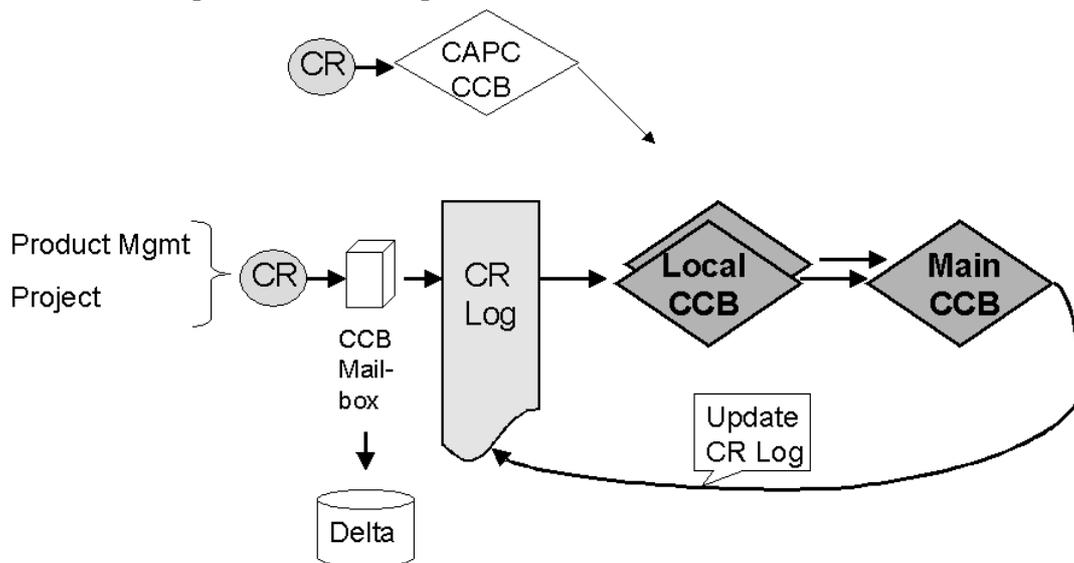
Figure 61 *The Change Request Flow*

The Change Request handling flow is shown below, in Figure 61. All denominations are normal Configuration Management terms.

With these pictures it can be summarised that all necessary methods seems to be available for how to keep the product information accurate and complete for the phase 10 project.

B.8.6   Information Flow

The picture below is the overall illustration of the project information flow involving the tools needed. This a generic picture. More tools exist to make the flow complete, but int his interview this picture is sufficient to describe the situation;

The picture indicates the problem to establish formal control and trace ability. With many tools and many interfaces the possibility to get fast and accurate information from involved systems is low. The project handling of today indicates a lack of an overall PDM system, although a good basic order is at hand as described earlier.

Below are descriptions for three major phases; Requirement Management, SW/HW development and Build management. The purpose is to give a background to the tools and why they are used. For each section a vision paragraph has been written. As mentioned earlier we want to describe the situation with information from a generic viewpoint. Much more can be described regarding details.
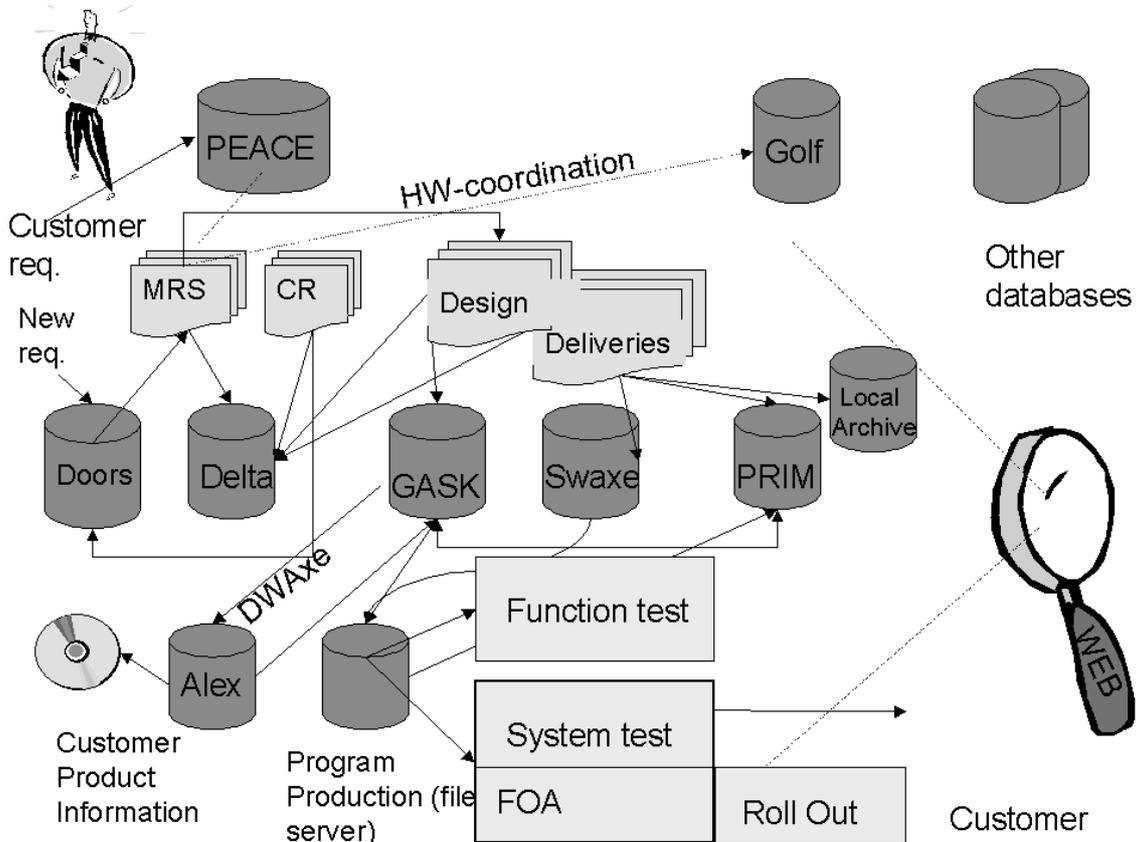
Figure 62 *Overview PPDC information flow*

Requirements Management
The picture below describes briefly the flow of requirements from customer to start of development:

The requirement from customer is stored in the peace database at Ericsson, Shin-Yokohama, Japan. The requirements are also stored in a Doors database at Ericsson, Kista where all (the total amount of requirements for the whole PDC systems) requirements are stored.

All applicable requirements for the project Phase 10 are then collected in a report from Doors, a document that is stored in the project library Delta.

The requirements are controlled in a baseline and changes are handled with Change Requests. However since the different requirements are described in an ordinary document and stored in the project library it is possible to update the document without issuing a Change Request. Thus this change will not be visible in the requirement baseline. The requirement baseline and the change requests are handled manual without any tool.

The Requirement baseline has no connection with the peace database in Shin-Yokohama, Japan.

Vision
That you can control the requirements so if they are updated you are forced to write a Change Request. The Change request should also be visible together with the requirement. Requirements should then be linked or tagged down to test specifications so that trace ability of each requirement is can be done through the whole system. The requirement baseline should be the only source of requirement information (no peace database).
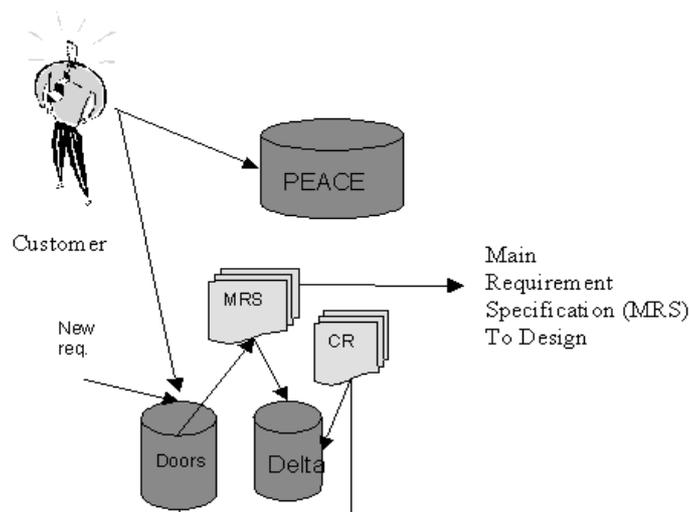
Figure 63   *Requirement flow*

SW Development/ HW-Design

SW Development
In the interview no persons with proper software knowledge could assist, therefore the text below is on generic level for Ericsson with the PDC adaptations known.

The different design organisations (systems engineers) are writing Requirement Specifications and functional descriptions in order to specify how to implement the original requirements. The product is then developed based upon this input from the specify product process as seen to the left in the picture above. The documents forming the input are normally stored in Delta (as described earlier). The software is stored in different places depending of the design organisation or product (Clearcase or local archives). The development flow is thus depending on manual tracking for changes occuring.

The software programming languages used are different. AXE exchanges use the internal Ericsson language PLEX. The Plex language has been used since the 70s and the supporting tools have been modernised during the years. Traditionally all source code have been stored in a central archive (GASK, for further information please see appendix B6).

Modern languages are also used within SW development in Ericsson today C++, Java and more. There are also attempts on generating code directly from UML.

In recent years the use of Clearcase for software design configuration control have accelerated. When the design is ready the code is archived in an approved archive, i.e. GASK (for non-Plex)/ SWAXE (for PLEX).

All documents written during SW development are archived in local archives. All product documents are stored in GASK.

Thus a mixture of archives exist with metadata and files in different places. Although CBS should control this, the need for an integrated environment exists.

HW-design
Hardware follows on a high level the corporate policies within Ericsson(please see appendix B6 for further information). Hardware development have matured much more
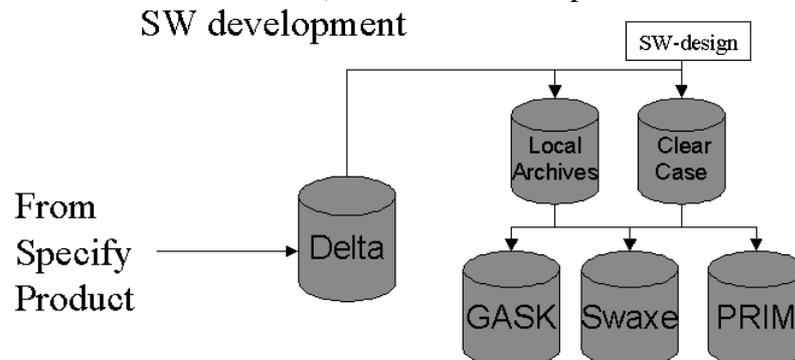
Figure 64 *Software information flow*

within Ericsson regarding common methodology and tools. However here as software differencies exist.

A normal flow is that system studies on block level (2-4 levels from the top functional level) identify the need of a hardware product. This is then specified to detailed needs, e.g. signalling.

A designer draws the prototype in a CAD environment. After a positive test of the functions of the hardware, the industrialisation phase starts. A hardware implementation support environment helps to adapt layouts to standardised boards acceptable to the producing organisation requirements. In a development project production people participates from the early start. If needed the PBA (printed board assembly) is taken to a "zero" series production to test property and function test.

A last production serie of the product serie is built as "common production" to measure that cost and time is correct to the requirements stipulated. The production unit then approves the product for general availability within Ericsson. In Figure 65 a overall flow for hardware design have been drawn:

Mechanics and cables have for a long time in Ericsson been judged as "non system dependant products". Most often the PBA board (and SW) are structured into function blocks defining the needed SW/HW modules for a certain function. Mechanics, cables, batteries, power supply, cross connectors are structured in a separate structure or separate branches of the functional structure.

Vision for Software and Hardware;   One common tool to be able to trace requirements from top level. Check in and check out functionality. SCM system linked for transparent linkage of information to a PDM system.

Within Ericsson projects to further support the development environment are focused upon the PDM system Metaphase. A Project to create an interface between the Metaphase and Clearcase is ongoing, this in order to get the full product information control involving both PDM and SCM.

Figure 65   *Hardware information flow.*

Build Management

The build process is centralised. All information is gathered at one point and distributed to all subprojects. At least for the later stages as described below. Indetail this process is of course much more complex.

After development the software is sent to TCM subproject. They put together a test bed (dump) for the other subprojects to test on. Deliveries goes through Swaxe (SW for AXE) is a tool that is used for delivery of AXE software products (PLEX code). The software for the base station (c-code) is delivered through GASK.

With the testbed available function test starts. The software is also delivered to SIT subproject where the AXE system is integrated with the packet data part. From this stage all faults are reported as Trouble reports in MHS (Modification Handling System). MHS is linked to PRIM. Products not registered in PRIM can not get fault reports in MHS. MHS is spread throughout Ericsson all over the world.

Customer Product Information documents is fetched by DWAXE (tool for Docware AXE) in Delta and GASK. They are put together in a Library called Alex. When products are released and documents are approved in solid revisions all customer product information documents (for operation& maintenance) are translated into Japanese.

The customer orders hardware directly via the Golf system.

Vision
Automatically delivery on demand of changed products (daily build). Possibility to create reports of the delivery (content, which Trouble reports are corrected, which Change

Figure 66   *Software Program and document library production*

Requests are included, which requirement is the delivery connected to.) Possibility to create reports over applicable TRs for the whole system

A fully automatic SW/DW production system that can start upon pressing a single button and then using the existing configuration information.

Test
After finalization of Function Test the System Test starts. After finalization of System test the system is ready to be sent to the FOA and Rollout project. The most important system here are once more the fault report system MHS.

Vision
The Test environment is still striving after the automated test. Thereby lowering the resources and competence to write the large amount of test documentation.

B.8.7 Conclusions

The basic CM method, the needed "know how" and "know why" are all in place in the PDC project phase 10. If you add the existence of Corporate Basic Standards (CBS see appendix B6) there is a good start for implementing tools. Also integrating tools should have a good chanse with all routines present.

The lifecycle model is clearly identified, this gives a good possibility to implement a defined PDM-SCM integration.

But it must be stated early on that one of the major remarks from Marita were the lack of tool support. The environment described in this report is not satisfying from the user point of view. there is no control over the total information flow. Ericsson is making efforts to change this environment.

Thus the analysis of connecting PDM and SCM is somewhat not valid today for this project. There are larger problems at hand. There seems to be a lack of a PDM system for the whole flow described in this interview.

Ericsson has projects ongoing to introduce the PDM system Metaphase. For this project however a variety of tools have been used. They will all have different functions but four of them do form the PDM system available today:

- GASK   Document archive released documents
- DELTA Document archive during development
- PRIM   Product register
- MHS   Fault reporting

B.8.8 Requirements upon the 4 most central tools

This chapter tries to state the reasons why some tools are used within the Project environment. Four examples have been described.

PRIM

The central product register (further described in appendix B6) is the only alternative for a product register inside Ericsson today Although other local systems are used at other situations, it is only PRIM that can give you the product number globally accessible. Sometimes data will not be registered in PRIM until the point of release. This gives the situation that during development many other tools are used.

There is no other global system within Ericsson with the needed interfaces to other systems existing today.

Alternatives to PRIM could be SAP or other local PDM systems.

Ericsson Corporate Standards demands the use of PRIM.

GASK

The central document archive GASK (further described in appendix B6) is often debated within Ericsson. There are other tools/archives used during the projects lifecycle. The security and global access ability gives this tool the advantage of being used for such a widely spread project as this. But this tool is used as PRIM, you will store the information here at the release point or other project milestones. Other local tools/ archives will handle the information during development.

Ericsson Corporate Standards demands the use of GASK.

MHS

Another old Ericsson tool is the modification handling system, MHS. This fault reporting system contains both methodology and user functions to register and track fault reports. Many other functions exist as well. The global availability and existing infrastructure are the main reasons for using MHS. Although old and often being mentioned as "soon replaced" it is still used, especially during and after system test. Within development other tools for fault reporting are used, e.g. Clear DDTS, ClearQuest.

The handling of change request is today manual within this project.

DOORS

Requirement handling and trace ability demands more and more of tools. Although still partly manual (se the text in the Information Flow chapter) some tools are getting choosed more often. DOORS have within Ericsson proved its possibilities to handle a larger volume and high complexity of requirements. DOORS is used within this project. Despite this, the verification of requirements within the project are still done manually.

Other Other tool alternatives for requirement handling within Ericsson are MATRIX and Requisite Pro.

B.8.9   Overall comments

Below are given eight comments about the situation for PDM/ SCM for the PDC phase 10 project. It should be noted that these comments apply to this project situation only.

These comments shall not be seen as general comments on Ericsson as a corporation. The comments have the whole spectrum from early requirements to maintenance & Support.

+ Ericsson Corporate Standards gives a good base for PDM
+ The global systems available (PRIM/ GASK and MHS) give Ericsson an advantage for this distributed project. Especially during and after system test.
+ SCM is well implemented.
+ The lifecycle model is clearly identified
- There are far to many systems involved without interfaces.
- Too much manual work, "the absence of PDM", connected to the fact that PRIM and GASK are old and soon are on their way to be phased out.
- Weak CM tool support for the phases up to system test in general
- Generic CM support; the lack of tools linking change requests to baselines.

SCM is today handled as all other products. This could be binders, tapes, screws telephone exchange or software products. This way of handling software is tied to earlier traditions and not to the full context of software. This could lead to a situation where software as described in this interview is handled quite separately from system, hardware or Docware support. Thereby a PDM/SCM integration is not focused upon.

With the new projects ongoing this situation might come to a solution.

B.9   PrakTek

B.9.1   Interview, Stockholm 2001-06-20
Lars Walterson, PrakTek

This interview presents some of the results from a project called iFame (Integration of cm FrAmework Matrix and Ecc), which PrakTek were involved in. Ericsson Process & Application Consulting was responsible for the project, and they have also authorized this interview. The technical orderer was GSM-platform development in Linköping.

PrakTek is a consulting company working within the information technology industry. Their field of competence covers both CM and PDM.

B.9.2   Project goal
The project started in September 2000 and ended in June 2001, involving approximately 15 persons, of which 5-6 developers. There have not yet been any official evaluation of the project. The goal was to implement parts of the CM Framework developed at Ericsson. To cope with the requirements they needed both a PDM and a SCM tool. Since ECC (Ericsson ClearCase) and eMatrix already were in use they decided to integrate these, and the project goal was thus to implement the first version of this integration. If this first version turns out to be successful the more long-term goal is to improve this integration and to also add functionality to eMatrix in order to get a system managing all relevant data during their product development, thus implementing the entire framework.

B.9.3   Background
Also for pure software development (i.e. no hardware involved) there is a need to manage a lot of meta data, or additional information, connected to the software items. In many SCM tools, including ClearCase, this is provided through attributes connected to the versioned items. These attributes are user defined and can be used to store any meta data. PrakTek's experience, however, is that attributes are seldom used for this purpose due to bad usability. Even for a CM group building PDM-like support, attributes only is not enough for a complete solution.

The iFame project was started to investigate the possibility to use a PDM-tool to handle meta data and implement a process, while still being able to store and retrieve the source code using the SCM-tool, i.e. to use each tool for what it is best at.

B.9.4   Tool architecture
Two types of objects are stored in Matrix: baseline (BL) and change request (CR). All data connected to these objects, including their processes, are also stored within Matrix (which is the main idea of the entire project). I.e. the PDM tool is used as intended. Important data referred to from these objects are the source code files included in the baseline and implementing the CR respectively. These files are also objects in Matrix (called CI, configuration item), but only implemented as 'links' to their 'real' storage in ClearCase (using 'foreign vault' in eMatrix).

Matrix   **ClearCase**

Interface

Baseline object BL1

Configuration Item objects

CI

CI

CI

Relation attribute, false or true: Sets to true when CR has been implemented.

main   **Attributes:**
        sum_CR_list
        sum_CR_finished
        sum_merged_CR_list
        sum_merged_CR_finished

0

1

2

bug-fix **Attributes:**
         sum_CR_list = 1:CR1
         sum_CR_finished
         sum_merged_CR_list
         sum_merged_CR_finished

0

1

Change Request object CR1

Figure 67   *In Matrix baseline and change request objects are stored with all their meta data and processes. Source code files are stored in ClearCase, but versions included in a baseline or implementing a CR are also represented as 'link'-objects in Matrix.*

Figure 67 depicts the architecture of the two tools and their integration. In Matrix a baseline object, BL1, is stored and refers to three CI objects (configuration items). A CI object is actually a link to a specific version of a file stored in CC. Also a CR object is stored in Matrix, referring to the two CI objects implementing the CR.

The interface used to integrate the tools is called MXCC and is developed by ITI. Much of the work within this project was to put new requirements on the MXCC interface. The interface is not only links, but attributes are mapped between the tools and CC commands can be executed from Matrix. For example can a user working within Matrix retrieve all information about a file stored in CC, e.g. attributes and all information retrieved from the CC command 'cleartool describe'. Metadata stored in Matrix can not be retrieved from within CC.

The interface is partly (one direction only) event driven. Some operations in Matrix also results in actions within CC. The other way around is, however, entirely manual. All modifications to data stored in Matrix is made through Matrix 'normal' interface and never due to operations in CC.

B.9.5   User functionality
The idea of using both a PDM and a SCM tool is to provide better support for managing meta data to the source being developed. This will give additional support especially to the project managers but also to the developers.

Baseline
A baseline process may vary in different projects but some core functionality is provided by the integration to cope with the most common. The project management gets

most of the advantages, by being able to work entirely within the PDM tool. Meta data from the source code can now be stored and managed together with the other project data.

Also the developers get some new functionality. A BL object has a status flag which initially is set to 'preliminary'. When transferred to 'approved', due to some management action, this results in a BL-label being set in CC. All CI objects referred to by the BL object are labeled, i.e. the specific version referred to is labeled. The developers can thus be aware of baselines being set by the managers.

Change requests
Also for the CR objects the project management gets most of the benefits being able to manage data uniformly, but the integration also provides a communication link between the management and the developers used during the process.

Also a CR object has a status flag. When set to 'approved' this information is propagated to CC by updating attributes for all files referred to by the CR object. This enables some support in terms of warnings and process guidance:

- When a developer checks-out a file he/she is informed of all CRs registered on that branch, including CR 'merged' from other branches. This makes it possible to check if the CR he/she will work on is registered, if not, something is wrong.
- When the file is checked-in, the system prompts the developer to acknowledge the CR implemented, if finished. When a CR is implemented this is automatically registered in CC and, after some manual communication, in Matrix.
- According to the CR process all CI objects referred to from a CR object must also be included in a baseline, i.e. referred to from a BL object. This can be checked both in Matrix where the same physical CI object is referred to, but also in CC where the developer gets a warning if a version has a BL-label name but the CR is note registered in the attributes.

Roles
The developers work (as normal) in ClearCase. Some updates in Matrix also results in CC being updated, which can be used by the developer. When a developer reaches a phase where meta data stored in Matrix should be updated, this is not done automatically. The project management is notified using some other tool, e.g email.

The project management (project leader, CM, etc.) is supposed to work only in Matrix. CM creates CRs in Matrix. A group of both project leader, CM, and developers do impact analysis for new CRs, creating the structure in Matrix and the links to the correct versions in CC.

B.9.6   Implementation details
In ClearCase each branch has its own set of attributes. CRs are registered in CC using four attributes: 'sum_CR_list', 'sum_CR_finished', 'sum_merged_CR_list', and 'sum_merged_CR_finished'. Each attribute has a value consisting of a list of CRs fol-

lowing the syntax <version in CC>:<name of CR>, e.g. "2:CR4 2:CR7 4:CR8". When a CR is registered it is appended to the attribute 'sum_CR_list'. Note that attributes to all files referred to from the CR object are affected. When the CR is implemented the same procedure is followed for the attribute 'sum_CR_finished'.

Attributes in CC are connected to a branch. This means that when a new branch is created also a new set of attributes are created. CRs are not being copied or moved when a new branch is created, but the new attributes are empty. When a branch (e.g. a branch called bug-fix) is merged (e.g. to the branch main) all CRs registered in the bug-fix branch are copied to the 'merged'-lists in the main branch.

When a developers checks-out or checks-in a file the system presents all CRs registered for that branch. I.e. all CRs that fulfills CR and/or CRmerged below:

$$CR \in sum\_CR\_list - (sum\_CR\_finished \cup sum\_merged\_CR\_finished)$$

$$CRmerged \in sum\_merged\_CR\_list - (sum\_CR\_finished \cup sum\_merged\_CR\_finished)$$

B.9.7   Conclusion

We do not know yet if this integration of ClearCase and Matrix is successful. Both tools are used for what they are best at, so the sum should be better than each of the tool itself. There are, however, some solutions that could be improved:

- Manual update from CC to Matrix. Apart from slowing down the process it may lead to data being inconsistent in Matrix and CC, when a change in CC not yet have been registered in Matrix. An event driven interface also from CC to MAtrix should have been possible to implement, but instead an already existing interface was used. 'Automatic' update of Matrix through polling has been tested but were to performance consuming.
- The developers are noticed/warned by the system at check-out and check-in, which seems to be late. The information stored could have been used give the developers more awareness, e.g. by letting them browse through and overview the information instead of just being picked on when doing wrong or prompted when the system thinks it is necessary.
- When a CI object is created in Matrix it is referred to a specific version of a file stored in CC. There is support that makes it more easy to find this correct version than just presenting the version graph of a selected file. There is no configuration specification (or similar) available from Matrix or any other search facility.

It will be interesting to follow the succeeding projects continuing the implementation of the CM Framework.

Index