Important Factors for a Successful Integration of Product Data Management and Software Configuration Management Systems

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

Abstract

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

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

Table of contents

1. Introduction

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

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

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

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

1.1 Research methods

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

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

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

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

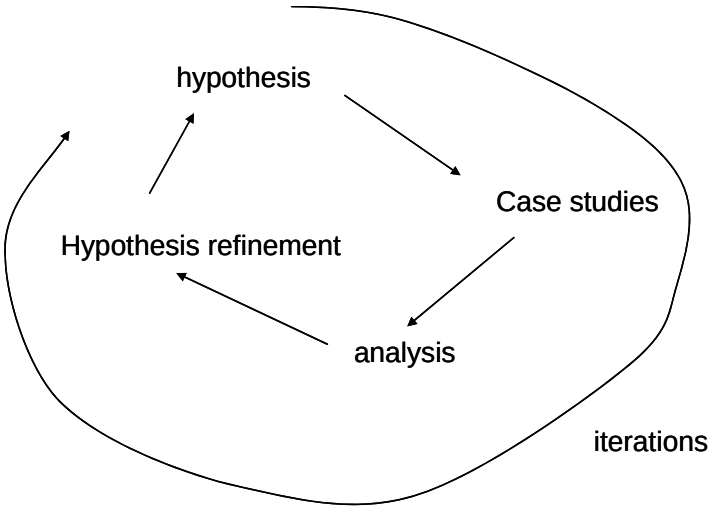


Figure 1. Data representation in PDM systems

2. Technical Principles and Key Functionality

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

2.1 Comparison of Technical Principles

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

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

2.1.1 System Architecture

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

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

Data Representation

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

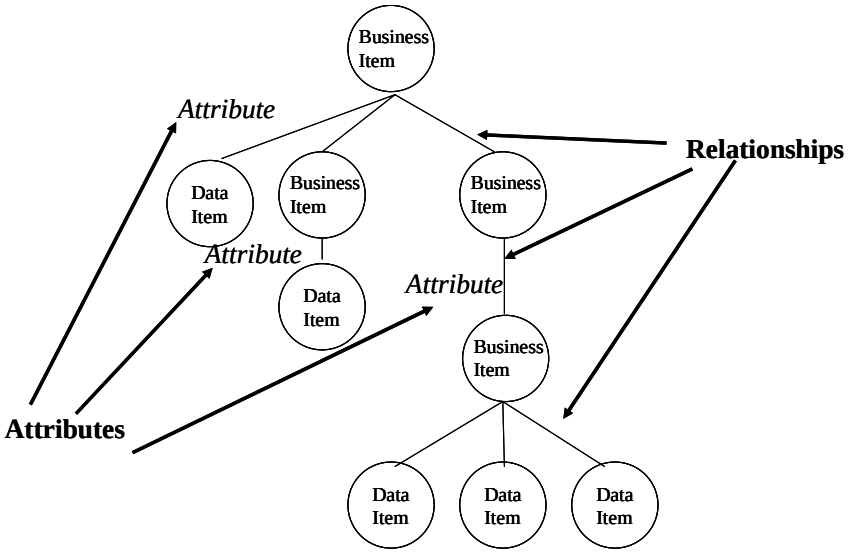


Figure 2. Data representation in PDM systems

In the SCM tools all kinds of file types and objects represented as a file or directory may be managed and stored. In most of the tools, the two types of files, source or binaries, are managed differently. For source files SCM provides additional support such as showing differences between different file versions or enabling interactive merging of two file versions. Metadata for a file is stored within the file and not in a separate database. Certain SCM systems use a similar paradigm as the PDM systems with a database containing metadata and files placed outside the database, but they do no have defined product structures.

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

Data replication

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

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

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

Application Integration

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

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

2.1.2 Evolution Model

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

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

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

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

2.1.3 Product Model

A product model is an information model used to describe the structure and behavior of a product managed by the system. A PDM system provides a support for building product models. The basic principle of product modeling in PDM is the composition relationship, used to form tree structures, referred to as product structures. The product structure is

visible and edited by the user. A hardware product has a physical existence and consists of physical parts, and thus represented by a part structure.

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

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

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

2.1.4 Process Model

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

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

The process models for PDM and SCM are conceptually similar.

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

2.2 Comparison of Key Functionality

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

We discuss the following functions:

- Version management - support for managing different versions of an object;
- Product structure management – support for describing the product in a hierarchy structure;
- Build management – mechanisms for building software (compiling and linking) and keeping generated software up to date, preferably without unnecessary rebuilding;
- Change management – keeping track of changes introduced in the product and providing support for implementing changes in the product;
- Release management – packaging the product in a form suitable for distribution and generating documentation to inform users and developers of changes included in the release;
- Workflow and process management – support made available for the developers in following a certain process with specific activities;
- Document management –support for managing documents allowing users to store, retrieve, and share them with security and version control;
- Concurrent development – support for control simultaneous access by several users (either by preventing or by providing support);
- Configuration management and selection management – providing support for creation or selection of associated versions of different objects;
- Workspace management – providing each user with a private location in which the user can work in isolation under the control of the tool.

2.2.1 Version Management

In PDM systems, versions of business items are called *revisions* and are organized in sequential series. Different revisions of a business item are connected by a relationship. *Versions* are used to manage the sequence of data items

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

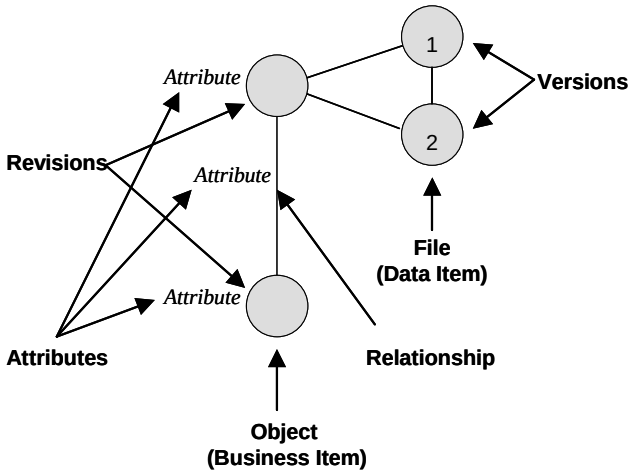


Figure 3. Version management in PDM

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

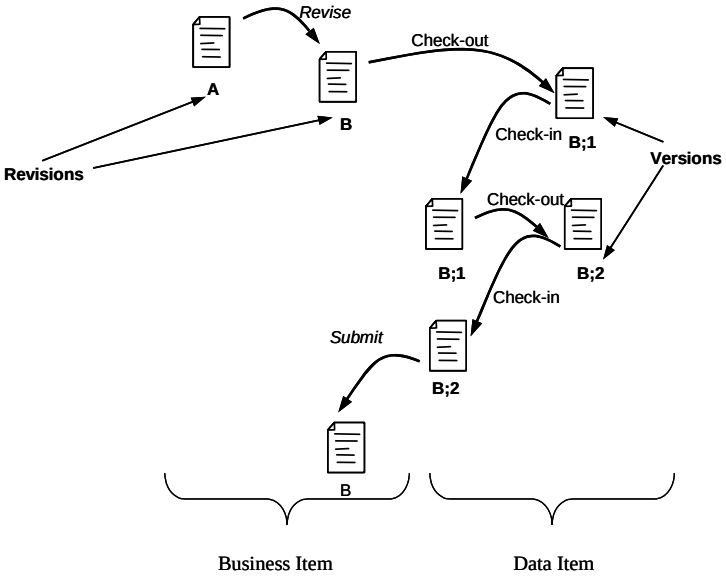


Figure 4. Check-in and Check-out of Data Items and Business Items

PDM supports concurrent development on a business item, but no support for merge, see figure 5. When one user revises the business item (revised to revision B in the figure), another user may revise the same business item but to next available revision (revision C in the figure). Both users may update the business item several times, and the business item

is frozen when it is submitted. PDM does not support merge when the two business items are submitted. Users not aware of this basic functionality may loose their updates.

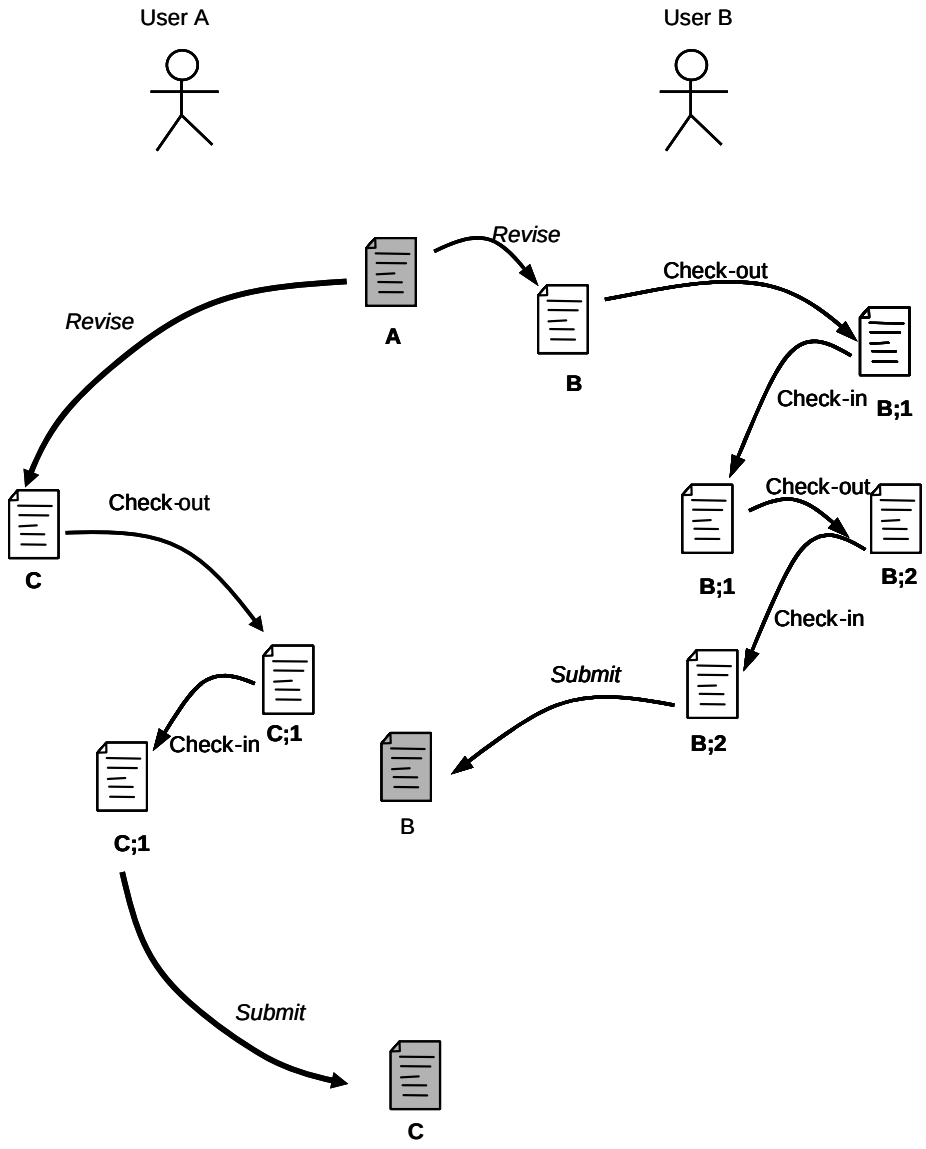


Figure 5. Concurrent development in PDM

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

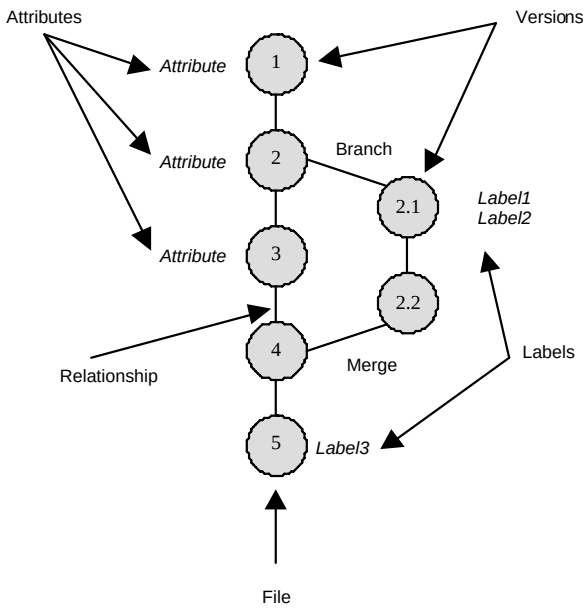
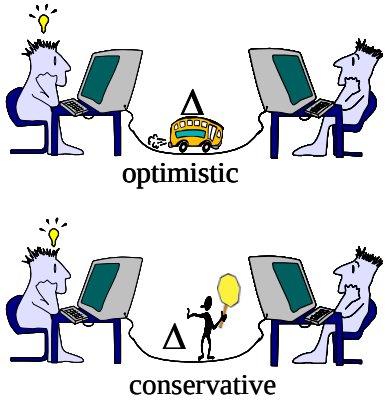


Figure 6. Version management in SCM

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

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

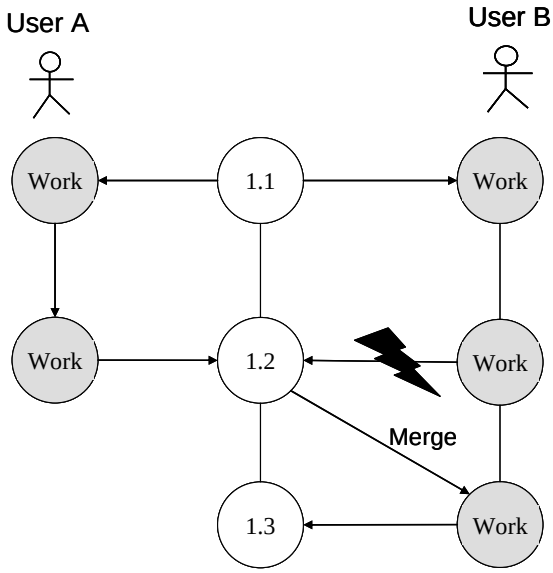


conservative strategy.

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

In figure 8 one file is updated concurrently by two users. In the reserved checkout model, the optimistic development strategy is supported by using branch and merges without locking on versions when checking out. From the branch 1.1, user A is checking out the file and updates it. At the same time user B is checking out the same version. User A updates

the file, and is ready with the changes before user B. User A decides to check in the file as 1.2. Then user B decides to check in the file, but is not allowed to check in before a merge with the changes made by user A is done. The user B can



check in the file as version 1.3.

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

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

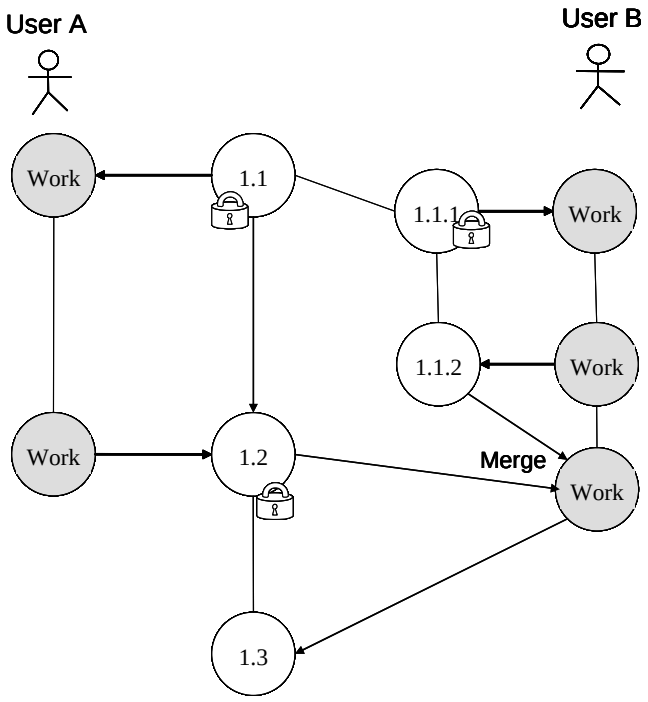


Figure 9. Concurrent development in SCM, a conservative update strategy with locking

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

- PDM manages *objects*. SCM manages *files* and *directories*;
- PDM uses *revisions* for major changes. SCM uses *versions* for all changes;
- SCM has *branches* and supports *merge* functionality. PDM does not;
- In SCM concurrent development on file level is supported. In PDM it is not;
- Both PDM and SCM tools have *attributes*. PDM support customized attributes. SCM has a special attribute called *label*, which is frequently used. General attributes, user-defined, are rarely used due to restricted visibility;
- PDM has *relationships*. SCM does not except the revision-of relationship implementing historical versioning;
- A relationship in PDM may have attributes in PDM but not in SCM.

2.2.2 Product Structure Management

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

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

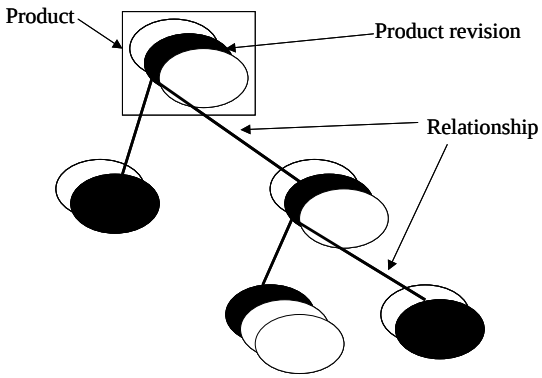


Figure 10. A configuration in a product structure in PDM

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

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

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

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

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

2.2.3 Build Management

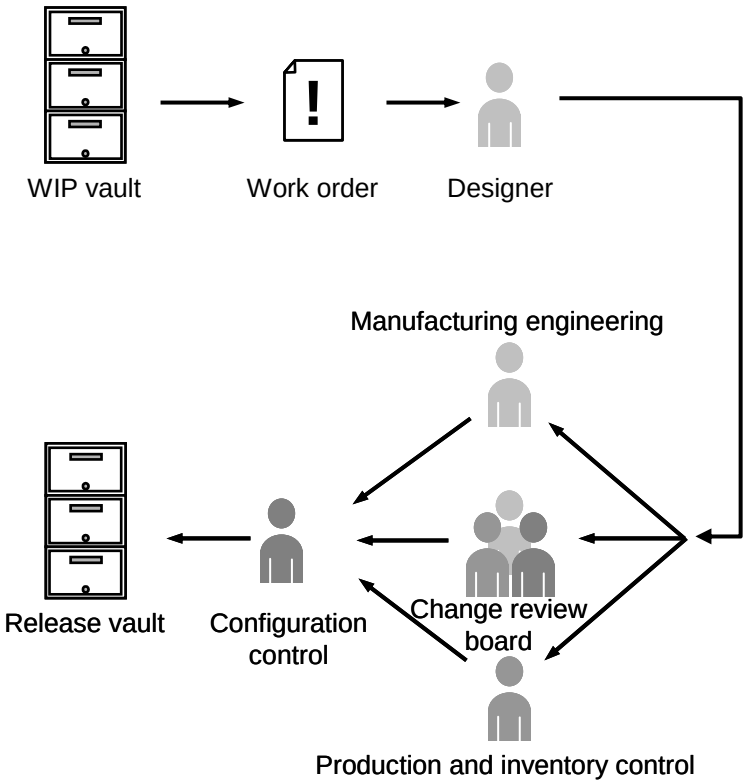
Build management in SCM supports the user in automatically building the software product and includes two central types of transformations. Source code is transformed to binary or executable form [14], and the product structure itself is changed. Typically, a new directory structure, which includes the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

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

2.2.4 Change Management

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

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



document is approved, it will be stored in the release vault.

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

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

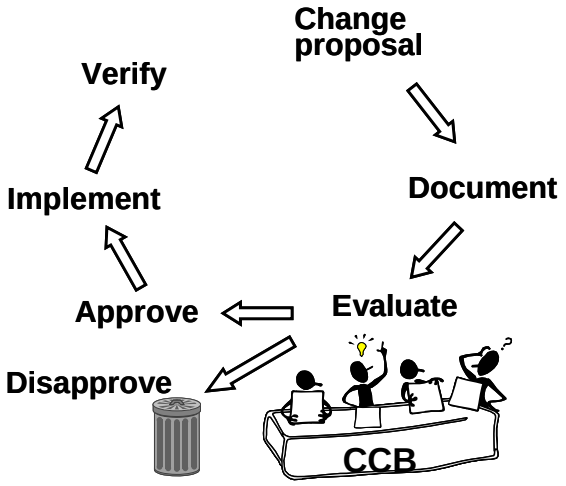


Figure 12. Example of a process for change approval in SCM

2.2.5 Release Management

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

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

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

2.2.6 Workflow and Process Management

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

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

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

2.2.7 Document Management

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

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

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

2.2.8 Concurrent Development

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

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

2.2.9 Configuration Management and Selection Management

Configuration management manages both hardware and software and originally focused on manager support. From a management perspective, configuration management directs and controls the development of a product by the identification of the product components and the control of their successive changes. The objective is to document the composition and status of a defined product and its components, ensure the correct working basis is being used, and the product is composed correctly. Examples of standards supporting this discipline are ISO 10 007 [25], and ANSI/EIA-649 [30]. Configuration management is both management discipline and a process.

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

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

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

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

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

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

2.2.10 Workspace Management

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

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

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

2.3 Conclusion

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

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

Table 1 Summary of functionality of PDM and SCM

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

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

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

3. Interoperability in Common Processes

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

3.1 Structures of Complex Products

Several development teams are involved in the development of a complex product. The teams use different technologies, different development processes, and different tools during the development and maintenance phases. The result from each team is assembled on the system level to provide a final product ready for production or delivery. Common to all product development activities is the necessity to manage data on the system level, between the teams, and within the teams. Figure 13 shows an example on system and subsystem levels of a complex product, where the system contains of mechanical, software and ASIC components. Product developed from the different subsystems will be

integrated and tested on a system level before the product will be manufactured and shipped to the customer. The required support on subsystem levels is different due to various used procedures and technologies, but common at the system level. At the system level, the differences between the subsystems are disregarded and each subsystem is treated similarly irrespective of whether it is a hardware or software component.

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

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

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

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

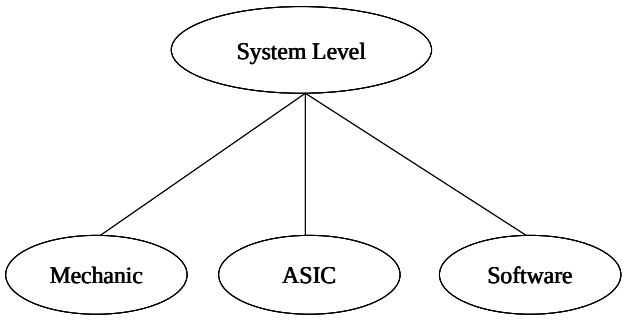


Figure 13. Example of a complex product with hardware and software components

3.2 Complex Product Lifecycle Management

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

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

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

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

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

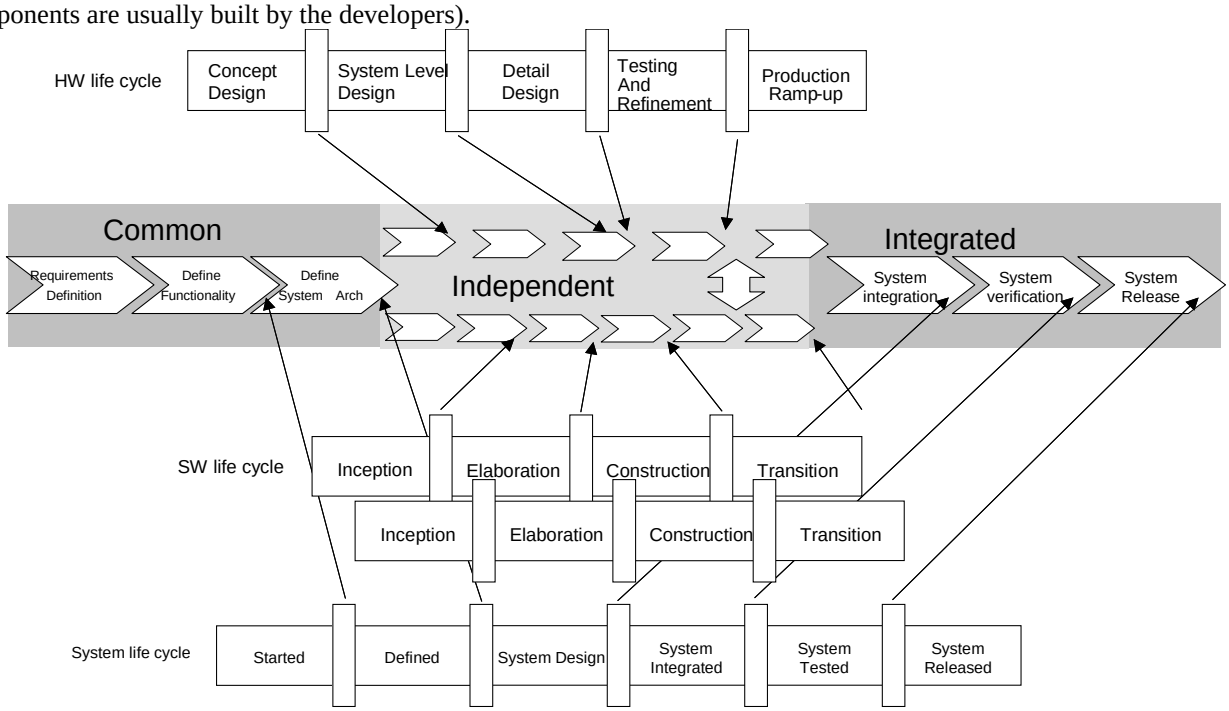


Figure 14. Complex Product Lifecycle

3.3 Conclusion

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

entire development process at all structure levels. To achieve seamless information flow on system and subsystem levels, integration of PDM and SCM tools is needed.

4. People and Culture

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

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

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

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

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

5. Different scenarios in an integrated environment

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

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

The integrated system has the following characteristics:

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

5.1 Scenario: PDM – User interaction

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

- *Query for a document.* The user states the document identity (or other search criteria) in the PDM system and the system (i) search for the document in the PDM system, if not found in the PDM system, a search is performed in the SCM system and (ii) a list is presented for the user of all found documents that match the search criteria.
- *Get a document.* The PDM system (i) look up the actual file in SCM by using the path to it, (ii) copy it, and present the document for the user.
- *Check out a document.* The PDM (i) uses the stored path and search for the document in SCM, (ii) checks out the document in SCM, (iii) copy it into the work-in-process vault in PDM, and (iv) set attributes required too synchronize the states of PDM and SCM (e.g. user identity and status). The user may now update the document.
- *Check in a document.* The PDM system (i) check in the file in SCM, (ii) set all attributes required synchronizing states of PDM and SCM, (iii) and deletes the file in PDM.
- *Delete a document.* The PDM system (i) checks if the user has the access rights and is allowed to delete the specified document in SCM, and (ii) check if the document is included in a frozen product or not. If the document is not included in a frozen product, (iii) search for the document in SCM, (iv) delete it in SCM, (v) delete attributes in SCM, and (vi) delete all metadata and relationships in PDM.
- *Import a document into PDM.* The path to a document existing in SCM is created and inserted in PDM. If metadata exist in PDM for the document, PDM (i) issues a query for the document in SCM, and (ii) saves the path to the document in the metadata. If the metadata for the document does not exist in PDM, (i) PDM must create a business item and populate it with metadata, (ii) search for the document in SCM, and (iii) save the path to the document in the metadata for it in PDM.
- *Export a document from PDM.* A PDM user wants to make a link to the document in PDM from SCM. PDM (i) checks if the document exists in SCM. If the document does not exist in SCM, PDM checks in the document version, related path to the document in PDM, and related metadata. If the document does exist in SCM, (i) PDM checks if the document version is the same. If the document version is the same, (ii) PDM makes a diff between the versions to check they are identically, and inform the user if any discrepancy. Other wise, if the document version in the PDM system is less than in the SCM system, (i) PDM inform the user that the document in PDM is older than the one in SCM, and no changes are made in SCM. If the document version in PDM is higher than in SCM, the path in SCM and related metadata is updated.

5.2 Scenario: SCM – User Interaction

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

- *Register a new product.* The SCM user registers a new product in the PDM system. SCM (i) creates a new product item, (ii) specifies all of the necessary attributes of the product (such as owner and parent product), and (iii) receives the product identity, which may be used in SCM for a different purpose such as creating a baseline, or a new working structure.
- *Register a product revision.* The SCM system must know the identity of the next revision of the product. (i) SCM sends an inquiry to PDM for the next product revision, (ii) PDM allocates the product revision, (iii) PDM sets the appropriate metadata, and (iv) PDM delivers the new product revision identity to the SCM system.
- *Register a new document.* One or several files stored in the SCM system are registered as new items in the PDM system. SCM must provide PDM with (i) the full path to the files and (ii) attributes with relevant metadata such as document status and document revision. Further, SCM must provide PDM with (iii) the identity of the product to which the information belongs. When registering a library or an executable in PDM, (iv) the reference to all included software source code files are also registered.
- *Export a document from SCM.* The path to an existing document in SCM is provided PDM. If the document does not exist, it must be registered first, and then the path is sent to the PDM system along with certain attributes.
- *Check out.* The file is already registered in PDM and may be frozen. SCM (i) checks if the file has already been checked out. If the file is not checked out, (ii) SCM checks out the file in SCM, (iii) and if metadata about the file is stored in PDM, all attribute for synchronization purposes is set. The user may now update the file.
- *Check in.* SCM will (i) perform a check in of the file, and (ii) set all attributes required to synchronize the states of PDM and SCM.
- *Uncheck out.* SCM unlock the file lock and changes the status in PDM.
- *Update product status.* SCM sets appropriate attributes in PDM, depending on the company-specific development process.
- *Delete document.* An SCM user wants to delete a document. (i) SCM checks if the user has the access right to delete the document. If the user has, (ii) SCM uses the document number label and searches for the document in PDM. (iii) If the document is not included in a frozen product, SCM will delete the document in PDM and (iv) in the SCM including all labels and attributes. (v) If the user does not have access rights for deleting the document, an error message will be returned to the user and the document is not deleted. If the document is used in a frozen product, (vi) an error message is returned to the user and no document is deleted.
- *Query regarding product revision.* SCM will use the product number label and query for current product revision in PDM. The result will be presented for the user.
- *Query regarding a product structure.* SCM will use the product number label and request PDM to retrieve the full product structure in which the actual product is included. The product structure will be presented for the user in SCM.
- *Query regarding documents.* An SCM user searches for documents placed in PDM only. SCM will (i) use a search key and perform a query in PDM to retrieve the actual document. The result (ii) will be presented for the user.
- *Import documents into SCM.* An SCM user wants to create paths to documents stored in PDM for reading purposes. SCM (i) checks if the documents exist in SCM. If they do not exist already in SCM, SCM (ii) issues a query for the documents to the PDM system, (iii) create appropriate paths to the documents, and (iv) sets the appropriate attributes in PDM and SCM.

5.3 Conclusion

We have set up integration prerequisites for a hypothetical integration between PDM and SCM. This resulted in two scenarios of user interaction, one for PDM and one for SCM users, including several use cases. The example shows requirements on intensive communication and interchange of information with difficulties in automatic synchronization and update of data. This can lead to a conclusion that either a tight integration of the tools should be achieved, or the processes should be isolated but at given and well defined points import/export of data should be achieved. The first case

would lead to a technically better solution, but it is a question if it can be achieved – since there are too many vendors of PDM and SCM tools, too many different domains which these tools cover, to be feasible to define a common information model. The second approach is more feasible from today's perspective, although it provides significantly lower quality of services.

6. Case Studies

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

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

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

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

6.1 Ericsson Radio Systems AB

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

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

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

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

6.1.1 Processes and Information Flow

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

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

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

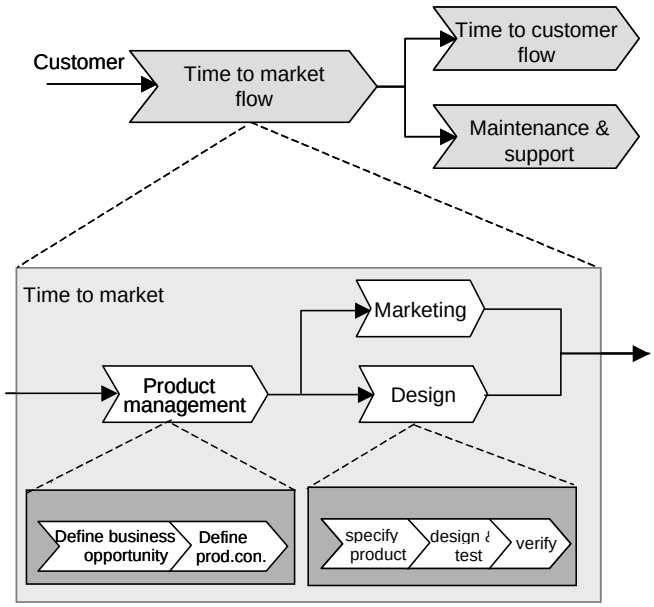


Figure 15. Time to market flow

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

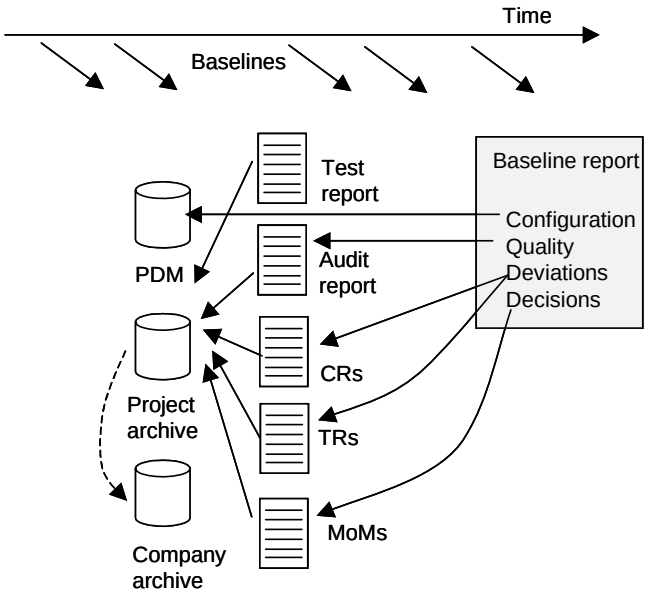


Figure 16. The baseline content

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

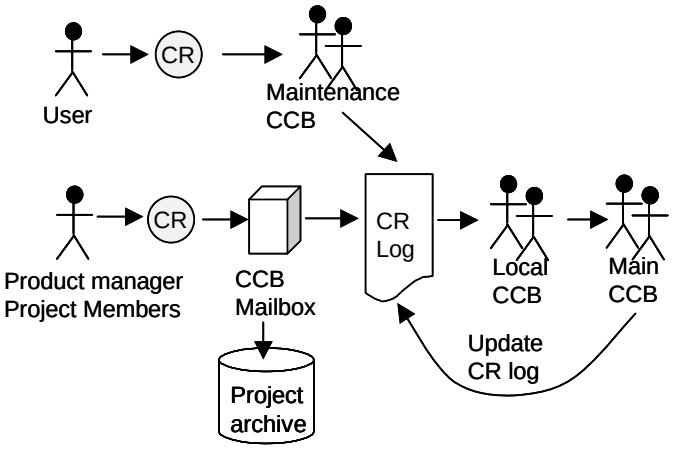


Figure 17. The CR flow

The product information generated in the project is managed in several tools. The different subprojects are managing their generated product data in their specific tools, e.g. hardware product data is managed in their tools, and software product data is managed in software specific tools. In figure 18 the project information flow, involving needed tools are shown. Customer requirements are registered in the customer requirement tool. New requirements generated either by product management or project team members, are stored and managed in the specific requirement tool DOORS® [37]. A main requirement specification (MRS) is written and updated from DOORS® and stored in the project archive. The MRS will be used for design for new or changed functionality. The design documentation is stored in the common company archive. Deliveries are stored in the software archive. PDM contains the product structure and is updated with the latest

information. When the design is ready for a function test, it will be fetched from the different archives. Builds will be temporarily stored on local file servers. When a system test is performed successfully, the product is ready for deployment. Customer product information will also be produced. This information is fetched from the common company archive, refined, and stored in the customer product information archive before delivery to the customer.

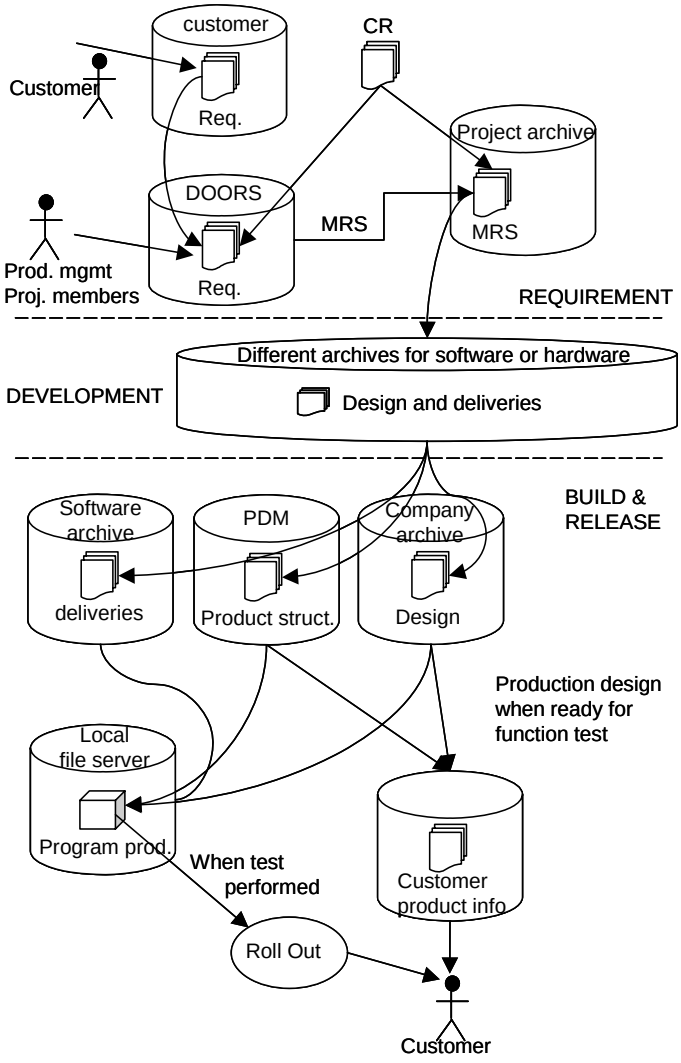


Figure 18. Overview of the project information flow

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

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

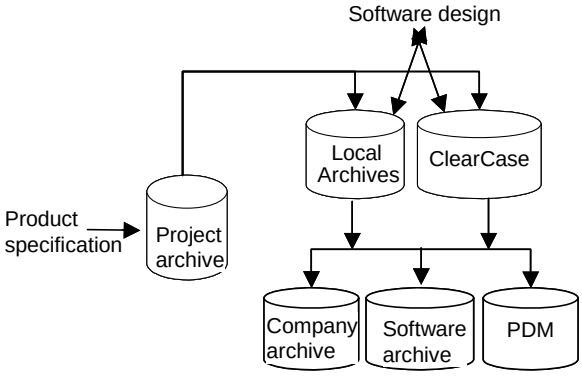


Figure 19. Information flow for software design

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

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

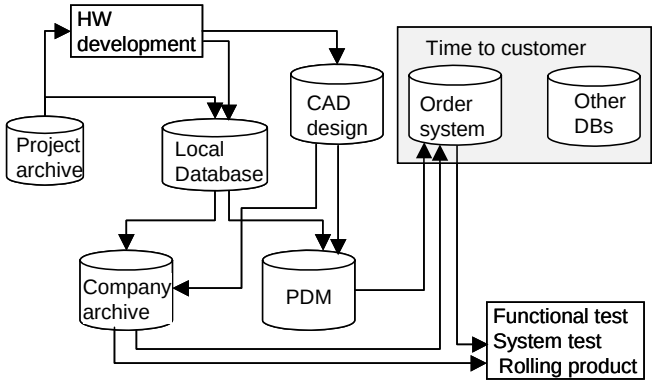


Figure 20. Information flow for hardware design

6.1.2 Tools and Technology

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

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

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

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

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

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

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

6.1.3 Integration Requirements

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

Process integration

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

Tool integration

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

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

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

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

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

Culture Aspects and Organization

The hardware and software development teams are geographically dispersed. Hardware developers are organized separately from the software developers, which result in low communication between the development groups, and low knowledge bout the other domain. In the project, separate subprojects are managing the development of hardware and

software components. In the company, a common terminology is used. This terminology describes the products and its describing documents. Both domains use their specific tools managing the product data. To achieve a more efficient development of the product managed in the project, the people must be integrated too.

6.2 Case Study: Industrial and Financial Systems

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

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

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

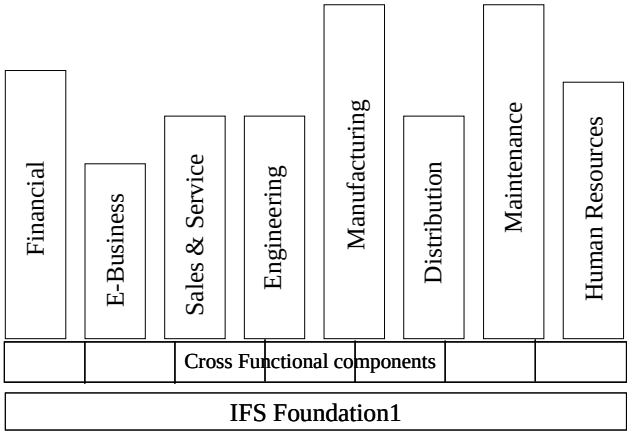


Figure 21. IFS Application functional architecture

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

6.2.1 Development Process

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

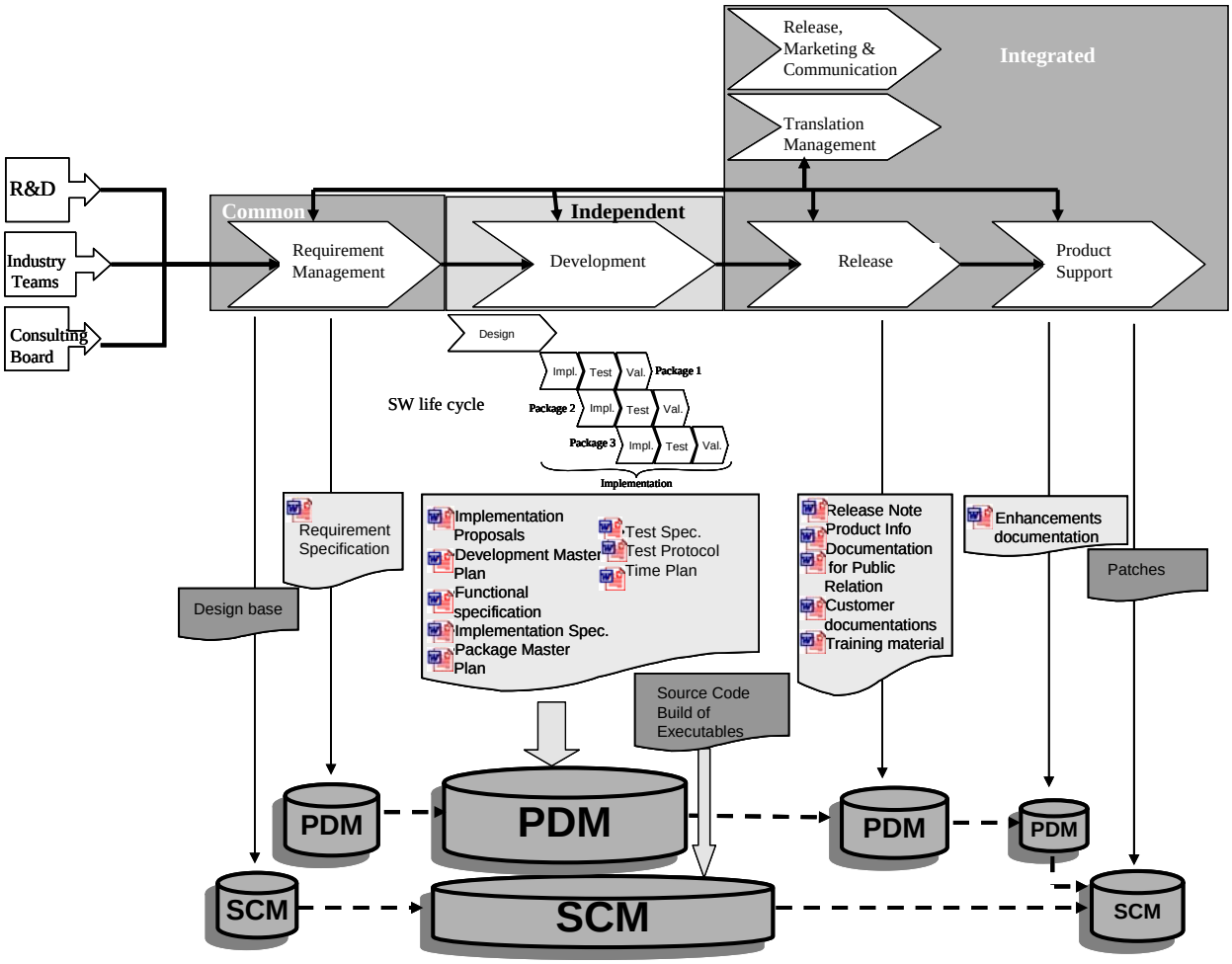


Figure 22. Product Life Cycle Process and connections to PDM and SCM systems

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

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

development phase, the incremental software life cycle, is depicted in figure 20 in the independent part of the product life cycle. All documents provided during the development phase are managed in the PDM system.

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

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

6.2.2 Technologies and Tools

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

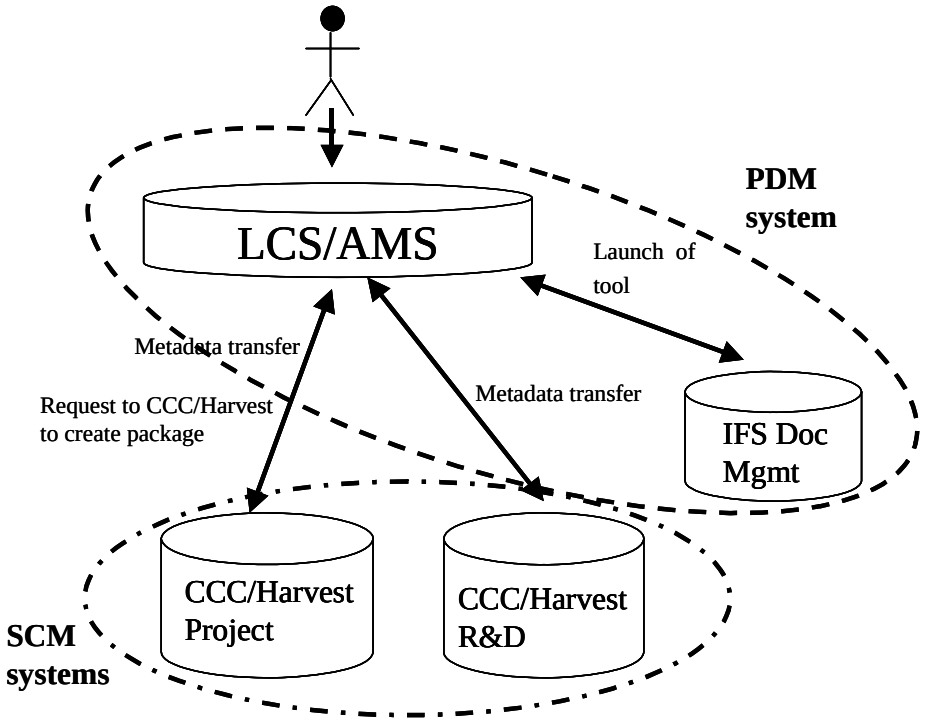


Figure 23. PDM and SCM tools and their interoperability

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

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

In the PDM system data represents of metadata, e.g. the patches and modules delivered to a specific customer, component description, part numbers, package id, and versions of delivered parts, and in which SCM repository the software is stored. In the SCM system data is represented as software files packed into product packages or patches. A package is a container with the files.

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

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

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

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

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

6.2.3 Integration Initiative

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

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

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

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

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

6.2.4 Culture Aspects and Organization

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

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

General update meetings are held for increasing knowledge of the total product portfolio. These meetings are held for all staff.

6.3 Findings for the case studies

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

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

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

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

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

7 Conclusions

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

8 References

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

[36] G. Hofstede, *"Cultures and Organizations Software of the Mind Intercultural Cooperation and its Importance for Survival"*, ISBN 0-07-029307-4, McGraw-Hill, 1997.
[37] Telelogic vendor of DOORS, www.telelogic.com, November 2005.
[38] J2EE, java.sun.com/j2ee/, November 2005.
[39] SOAP, www.w3.org, November 2005.
[40] OPC, www.opcfoundation.org, November 2005.
[41] Business Service Optimization vendor of CCC/Harvest, www3.ca.com, November 2005.