Quality Improvements by Integrating Development Processes

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

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

Abstract

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

1. Introduction

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

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

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

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

During 2001 the Association of Swedish Engineering Industries sponsored a project about PDM and SCM. As part of this project several case studies were performed, e.g. ABB and Ericsson AB, in order to analyse concrete current requirements and solutions. The project resulted

in a report [1]. The work went further, more case studies were performed, Sun Microsystems, and Mentor Graphics, which resulted in a book published by Artech House [2].

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

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

2. Development Processes and Infra-structure Support

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

2.1. Processes and Underlying Principles of PDM and SCM

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

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

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

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

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

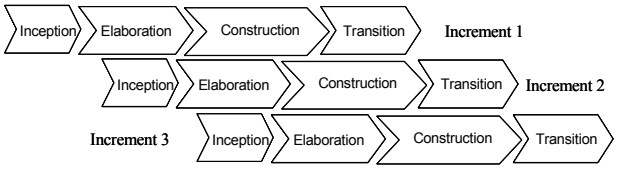

Figure 2. A generic incremental model commonly used for software development

The most fundamental differences in the development processes are the following: hardware development, supported by PDM, follows a sequential process with a clear separation between the phases. The software development process, supported by SCM, is flexible, with unclear borders between the phases. While outcomes from different phases of hardware development differ

significantly in form and even physical shape (a technical drawing of a product is very different from the product itself), the outcomes from software development phases are very similar and often only transformations of each other (for example, from a UML design code can be partially generated, and the final production is a transformation of source code into binary code). Such facts make these processes incompatible.

2.2 Information Management and Data Flow – A Case Study

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

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

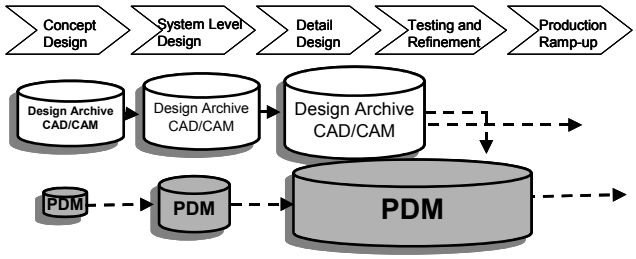

Figure 3. Processes and information storage for a hardware product

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

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

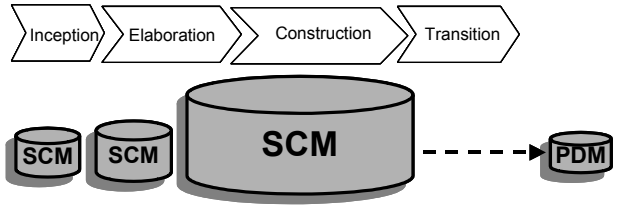

Figure 4. Processes and information storage for a software product in one increment

2.3 Standards

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

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

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

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

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

There are different standards and models for different Product Life Cycle Management (PLCs). Some standards addresses the life cycles of systems closely related to PDM and SCM, e.g. ISO/IEC FDIS 15288 Systems Engineering – System Life Cycle Processes [8].

For integration purposes no standards exist today.

2.5 Conclusion

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

3. Tools and Technology View

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

3.1 Data Representation

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

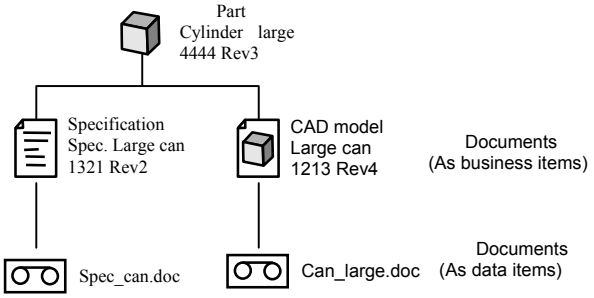

Figure 5. Data Representation of documents

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

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

3.2 Version Management

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

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

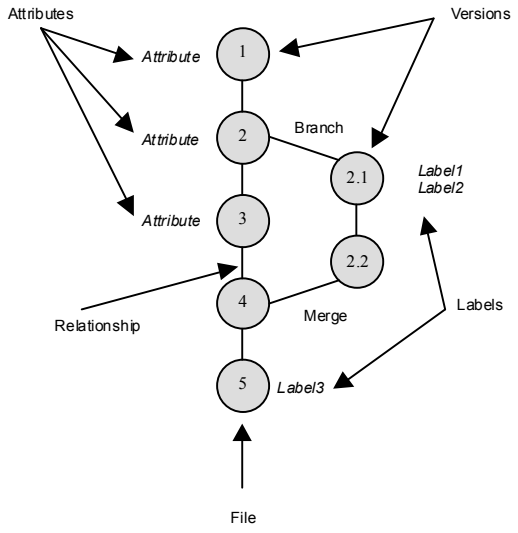

Figure 6. Version management in SCM

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

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

3.3 Management of Distributed Data

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

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

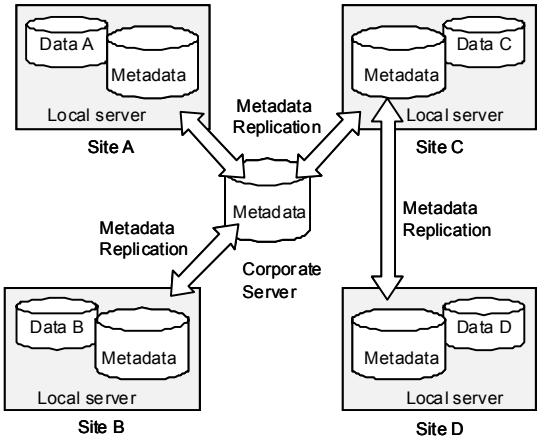

Figure 7. Server replication in a typical PDM environment

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

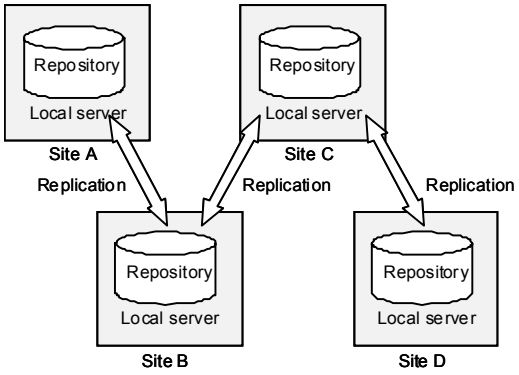

Figure 8. Server replication in a typical SCM environment

3.4 Product Structure Management

Product structure management is a basic and fundamental functionality in PDM systems [5]. The product structure is a configuration of parts connected by relationships. A database model supports the building of a

product structure. Figure 9 shows an example of a product structure of a bicycle. The structure is a so-called quantified Bill-Of-Material (BOM) used in production for collecting all objects and information.

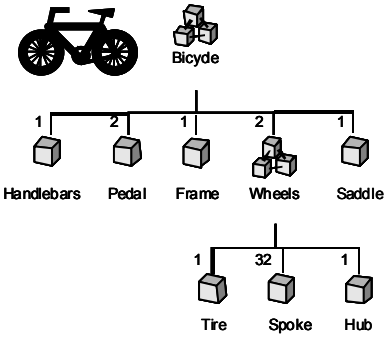

Figure 9. Example of a product structure in a PDM tool

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

3.5 Process support

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

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

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

3.6 Document Management

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

3.7 Conclusion

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

4. People and Cultural View

The cultural differences between hardware and software development groups play a much more important role than visible when building integration between PDM and SCM. First of all, both domains are huge using completely different tools. Secondly, users from the different domains do not have knowledge about the other domain. Low communication between the domains causes poor understanding of each other's problems and requirements. Thirdly, users from both domains believe that the system they use can manage all situations from the other domain [2],[11]. Fourthly, PDM and SCM users are often located at different departments within the company. Their geographical separation can increase the gap in their understanding of the other group. Fifthly, the hardware designer uses a lot of documents to describe the product. These documents are transferred to the production and manufacturing part used of another person to produce the actual product. Hence, the hardware designer focuses on documents. The software designer writes a lot of source code. The designer then generates the actual product, the load modules, with no other person involved. Hence, the software designers focus on source code more than documents and have

small understanding of the importance of writing documents.

4.1 Terminology

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

4.2 Conclusion

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

5. Integration

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

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

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

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

5.1. Process integration

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

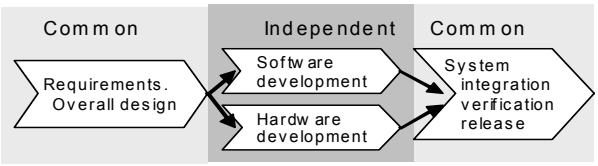

Figure 10. Integrated process

5.2. Tool integration

The identification of the integrated process will lead to decisions, which tools can be used and which are the integration requirements. Further a policy for the integration of the tools should be decided: integration can be achieved through a common information model (tight integration), or in a loose way in which the tools preserve their internal structure, but interpolate through Application Program Interfaces (APIs), integration languages and commands, or web-based services and components [3]. A tight integration is based on a consistent information model, which makes simple interoperation between the tools. However, a tight integration requires a lot of efforts to achieve agreement about a common information model. Since different tool providers want to keep their advantages on the market, they usually are not willing to change their internal

representation to standard formats and models. Instead of that they focus on enabling integration with other tools. In a loose integration there is not one common information repository, but the same data may be saved in several, different, repositories. For this reason a policy for information management must be decided. For example: (i) which system should be the main archive for documents (drawings, source code, etc.), (ii) which system should manage the product structure and the revisions of all products included, and (iii) which system will manage metadata of delivered products. In particular the problem of version and configuration synchronization might be problematical.

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

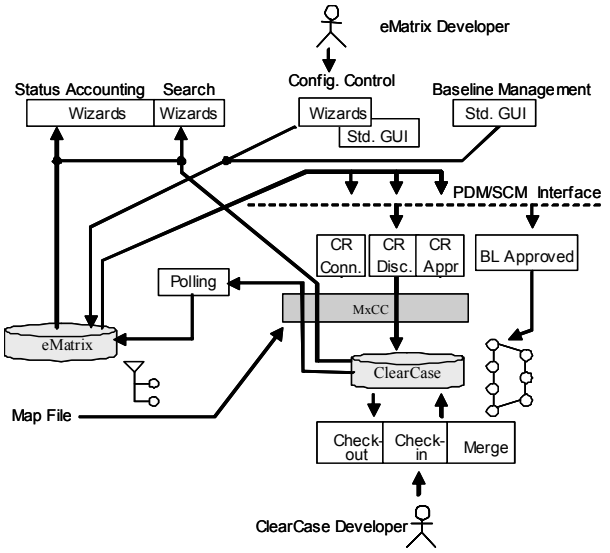

Figure 11. PDM and SCM integration example

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

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

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

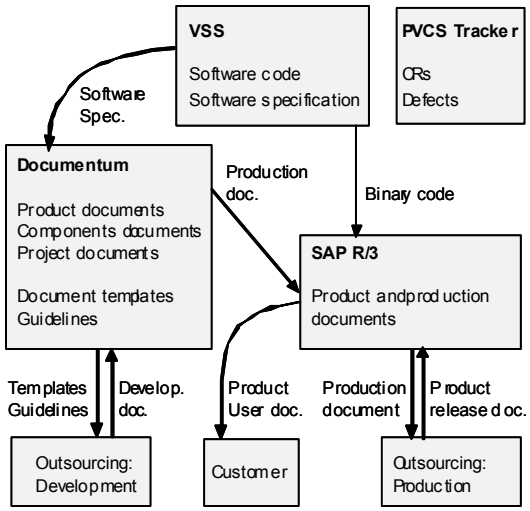

Figure 12. Example of a complex integration of PDM and SCM tools

5.3. Common Understanding

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

In [3] a case is discussed, which did not succeed to integrate a SCM and PDM tool. The integration did not succeed because the tool vendors focused only on

technical interoperability issues building automatic import/export tools, but did forget the two other important factors. First, they neglected the process issues – which actions and which tools are performed in which phases. Second their decision was that the user interface, the terms, and in general the overall philosophy should follow PDM standards. This caused large problems for software developers, which did not, understood the PDM concepts, and were not willing to accept them.

6. Conclusions and Future Work

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

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

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

7. References

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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