Managing Complex Systems – Challenges for PDM and SCM

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

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

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

ABSTRACT

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

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

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

1. Introduction

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

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

2. SCM and PDM Domains

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

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

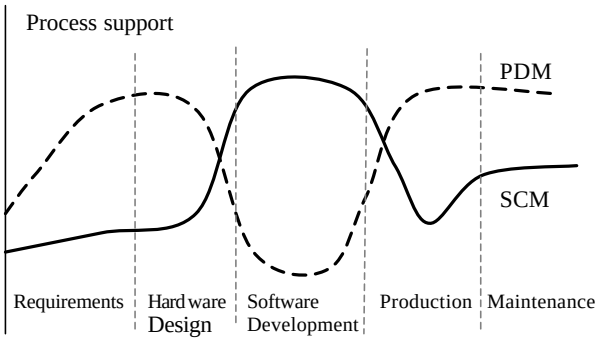

Figure 1. PDM and SCM Process support

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

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

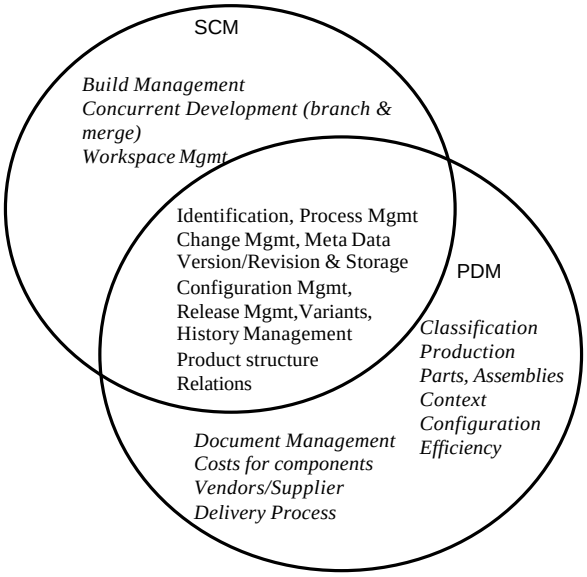

Figure 2. The main functionality of SCM and PDM

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

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

3. Integration Possibilities

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

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

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

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

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

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

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

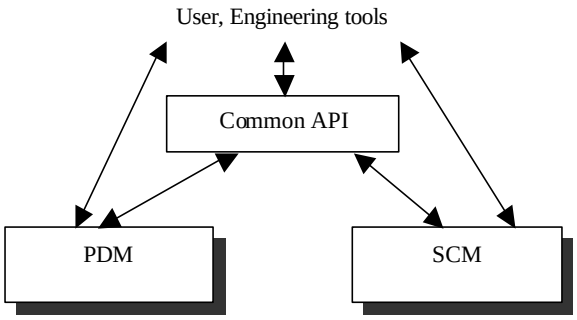

Figure 3. Direct and indirect use of tools

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

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

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

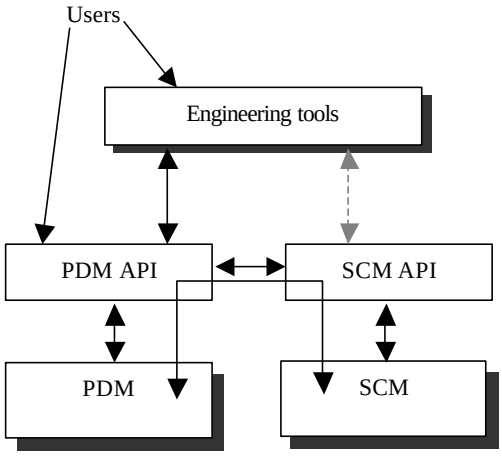

Figure 4. Partial direct SCM/PDM integration

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

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

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

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

4. Integration Experience

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

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

In Metaphase the software products will be managed together with all hardware products within the same product structure. In Figure 5 shows an example of a

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

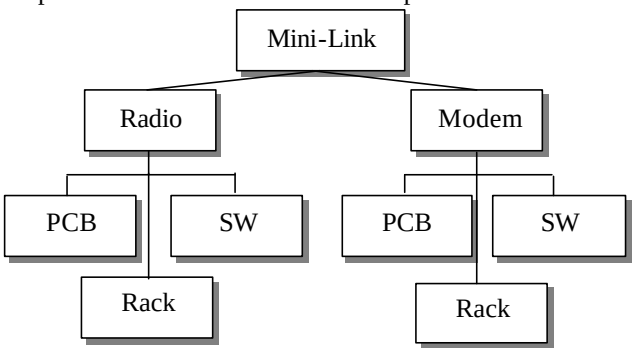

Figure 5. Example of a Product Structure

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

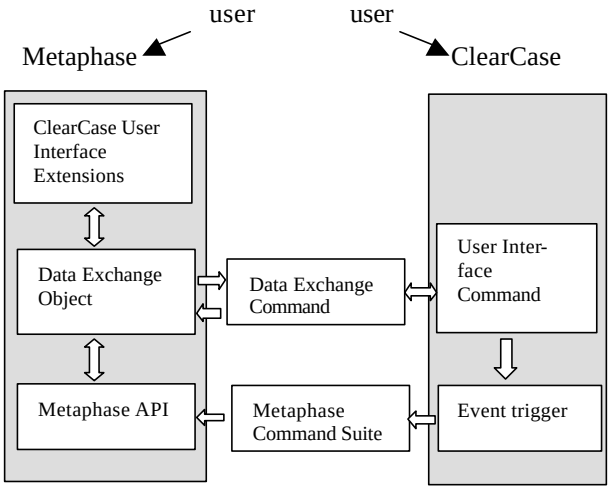

Figure 6. Data Exchange Architecture

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

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

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

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

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

These required conditions show how complicated the integration is developed. There exists a high risk that data

will not be synchronized. In addition to these implementation problems, there exist problems of a more general nature. SCM users do not understand how PDM systems work and vice versa. All PDM systems, including Metaphase, are designed to meet the needs of the hardware people including their terminology and not the SCM people.

5. Investigation Initiative for PDM/SCM

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

The purposes for the project are to:

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

The work is performed with the following activities:

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

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

6. Conclusion

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

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

7. References

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

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

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

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

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

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

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

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