Development and maintenance of product configuration systems: Requirements for a documentation tool

Article *in* International Journal of Industrial Engineering · January 2005

CITATIONS
12

READS
270

4 authors, including:

Lars Hvam
Technical University of Denmark
165 PUBLICATIONS **1,571** CITATIONS

SEE PROFILE

International Journal of Industrial Engineering, 12(1), 79-88, 2005.

DEVELOPMENT AND MAINTENANCE OF PRODUCT CONFIGURATION SYSTEMS: REQUIREMENTS FOR A DOCUMENTATION TOOL

Lars Hvam, Simon Pape, Klaes Ladeby Jensen, and Jesper Riis

Centre for Product Modelling
Dept. of Manufacturing Engineering and Management
Technical University of Denmark
DK2800 Lyngby, Denmark
www.productmodels.org

Product configuration systems are increasingly applied to automate the configuration of complex products. A configuration system is an expert system designed to combine specified modules according to constraints. The constraints are stored as product data and rules in a product model, and one of the most essential tasks is thus to develop a complete and consistent product model which can reflect the actual product.

A procedure for building configuration systems has been developed at the Centre for Product Modelling (CPM), and the procedure has been successfully applied in several industrial companies. CPM's experience with the procedure and the hitherto empirical experience from companies having applied the procedure have revealed that there is a need for an IT-based documentation tool to support the process of constructing and maintaining product configuration systems. Time can be saved by letting a documentation tool handle trivial time consuming tasks (notification on change, consistency check etc.), as a computer often handles these tasks in a better way. Thus, a serious bottleneck in the maintenance of configuration systems can be eliminated by applying Information System (IS) technology to support the documentation process.

This paper deals with the requirement specification of a documentation tool for product configuration systems, based on CPM's procedure and experience from four Danish industrial companies which have applied the procedure. The requirements have been gathered and structured by using object-oriented system development techniques based on an analysis of the existing product configuration processes of the companies. The analysis was based on the procedure for building configuration systems as developed at CPM, and revealed several common requirements within the different companies.

Significance: There is an actual need for a documentation tool for product configuration systems. The objective of the study is to identify and capture requirements of such a documentation tool, and serve as a basis for future design and implementation.

Keywords: Product modelling, product configuration, object-oriented system development, IT-based documentation system.

(Received 1 October 2003; Accepted in revised form 1 April 2004)

1. INTRODUCTION

Several companies have acknowledged the opportunity to apply IT-based configuration systems (e.g. Oracle Product Configurator, iBaan Configurator, or Siebel Configurator) to support the activities of the product configuration process. Companies like Dell Computer and American Power Conversion (APC) rely heavily on the performance of their configuration systems, as a configuration of their complex product portfolio would not be feasible if the product configuration processes should be carried out manually (Tiihonen et al., 1996).

The core of any configuration system is the underlying product model. The product model contains "*all relevant information concerning a given product during the life cycle*" (Krause et al., 1993) in a structured manner, focused on the knowledge necessary for fulfilling the desired tasks of the configuration system (e.g. design, sale, manufacturing, etc.). The product model is interpreted in a configuration system by the inference engine, that "*reason with both the expert knowledge in the knowledge base and [knowledge] specific to the particular problem being solved*" (Cawsey, 1998). Thus, the product model serves as a repository, containing information about the function and structure of the product as well as life cycle information (Schwarze, 1996). The amount of information can easily become very large and hence difficult to cope with – both when establishing the product model and during the following maintenance (Barker & O'Connor, 1989).

At CPM, Technical University of Denmark, a procedure for building configurations systems has been developed (Hvam, 1999). The procedure describes a staged approach with seven phases and concepts for documenting the product

model by using well-known techniques from the domain of IS development (object-oriented analysis and design). The procedure and the concepts have proven to be useful and applicable for different industrial areas in several industrial cases (e.g. Hvam et al., 2001; Hvam & Malis, 2002b; Hansen & Hvam, 2002). The application of the procedure to still more complex and comprehensive product portfolios has also revealed the need for a documentation of the product model by using an automated documentation tool.

Hvam & Malis (2002a) deal with the development and testing of a prototype for a documentation tool in a Lotus Notes environment. The prototype has been further developed and is now being applied successfully in two companies. Though the two Lotus Notes solutions do not support all the documentation activities, they have – as pilot projects – proved that an IT-based documentation tool can help remove the bottleneck of trivial tasks of storing and retrieving information in the process of construction and maintenance of product configuration systems.

However, why not simply use the standard configuration systems as documentation systems? According to Hvam & Malis (2002a) the answer is that though many standard configuration systems offer a (varying) functionality to document the product model, none of the systems offers a sufficient documentation facility. For example, product data and rules were separated in all tested configuration systems, and the program code was regarded as self-explanatory. Their conclusion was therefore that a separate documentation tool had to be created.

The main objective of this article is hence to present a requirement specification describing the concept and basic principles for documentation of product configuration systems inspired by the functionality of typical Computer Aided Systems Engineering (CASE)-tools and Product Data Management (PDM)-systems. The article can thus serve as an inspiration for software providers to develop a tool to support the documentation of product models. The providers may either be providers of standard configuration systems (integrating documentation as a part of their systems) or third party providers (developing a standalone documentation tool).

1.1 Approach for identification of Requirements, and the Structure for the Present Article.

Regardless of whether the documentation tool is to be developed from scratch or a commercial-off-the-shelf product can be purchased and adjusted to product configuration, it is important to have a proper requirements specification in order to be able to base the choice on actual needs for documentation of product configuration systems, and not on assumptions.

In the first case, developing a documentation tool is not different from developing any other IS. It has been proved that both money and time can be saved, and quality and productivity improved, when an organisation handles a development process as a project by using a structured approach – a methodology (Whitten et al., 2001). The methodological approach is likely to require a larger effort at first, but advantages in terms of improved quality and saved costs makes it feasible in the long run.

An object-oriented approach has been followed in this project in order to set up a requirements specification for a documentation tool. The approach consists of three steps and starts by identifying a set of use-cases in the domain where the documentation tool is to be used, based on CPM's procedure and disclosure of documentation needs when using the procedure. The use-cases are used to describe the interaction between the user and the system – before the system is designed. Thus, the use-cases serve as the basis of the requirement specification, and every design decision should be traceable to a requirement described in a use-case. A use-case diagram depicts the relation between all use-cases and each use-case should be specified in detail with scenarios.

Second step is the identification of needs for functionality of a documentation tool through qualitative interviews with product modellers in four Danish companies who has used the procedure to build and operate product configuration systems.

The last step is the drawing up of the requirement specification for a documentation tool, that can support the documentation process when building and maintaining configuration systems by applying CPM's procedure for development of configuration systems. This is also done by describing a set of use-cases that contains the identified requirements and functionality for the documentation tool.

The object-oriented approach is not a simple sequential task. Both use-case diagrams and class diagrams are refined iteratively. The Unified Modeling Language (UML)-notation (see e.g. Booch et al. (1999)) is quite simple and understandable even to people without IS-development experience, and the diagrams are therefore often presented to a user-committee for "approval" and in order to get constructive comments. The class diagrams and use-case diagrams should – after a number of iterations – represent the requirements for the upcoming documentation tool.

The present article is organized as follows. After an analysis of the documentation needs in section 2, the requirements specification is presented in section 3. Finally the conclusions of the present article are summed up in section 4.

2. ANALYSIS OF THE DOCUMENTATION NEEDS

2.1 Process Analysis

The requirements specification is based on the structured approach for building configuration systems as developed at CPM. The procedure consists of seven phases as depicted in figure 1.

The starting point is an analysis and redesign of the business processes affected by the configuration system; including a determination and demarcation of the configuration system to support the processes (phase 1). The scope is now set and the next phase is to analyze the product portfolio and identify modules and their interrelations (phase 2). Phase 3 covers the final design of the configuration system using object-oriented techniques, and a specification of the requirements and the user interface. Phases 4 through 6 cover the actual design, programming and implementation of the system in the organization. Finally, phase 7 encompasses the operation and the maintenance of the configuration system. The phases 3 through 7 generally follow the common object-oriented life cycle (Hvam et al., 2000; Coad & Yourdon, 1990; Booch, Rumbaugh & Jacobson, 1999).

The procedure is not a rigid sequential approach. Normally, the configuration system is developed through iterations, where the phases are run through a number of times.

Phase	Description
1	**Process Analysis** Analysis of the existing configuration process. Statement of the functional requirements to the process. Design of the future process, and definition and delimitation of the areas to be supported by a configuration system.
2	**Product Analysis** Products and possible life cycle systems are analysed and modelled by the use of a Product Variant Master (PVM).
3	**Object-oriented Analysis** Creation of object classes and structures based on the PVM. Description of object classes on CRC-cards (Class-Responsibility-Collaboration). Definition of user interfaces. Requirements for the configuration system.
4	**Object-oriented Design** Defining and developing the model for a specific programming tool.
5	**Programming** Programming the system e.g. in a standard configuration system.
6	**Implementation** Implementation of the product configuration system in the organisation. Training users.
7	**Maintenance** Maintenance and further development of the product configuration system.

Figure 1: The procedure for building configuration systems (Hvam et al., 2000).

The Documentation of the Configuration System

The documentation of the configuration system is developed through phase 2 (creation of a Product Variant Master (PVM)) and phases 3-4 (development of class diagrams and CRC-cards). In phase 5 the documentation is used as basis for the programming, and the documentation has to be maintained during phase 7. Thus, a documentation tool is to support the activities in phases 2-5 and 7.

The procedure for building configuration systems uses three elements to document the product model: 1) the PVM, 2) CRC-cards and 3) class diagrams. The PVM (see figure 2, left) is developed in cooperation between product modellers and the domain experts, usually by drawing the PVM manually on a large piece of paper (e.g. 2×4m) by using tools like e.g. Visio. Information about the nodes/classes in the PVM is filled into CRC-cards (see figure 2, right) during the process, usually by using a simple word processing tool like MS Word (Bellin, 1999). In this way, a CRC-card is a kind of elaboration of the nodes in the PVM. The contents of the CRC-cards is linked semantically to the corresponding nodes in the PVM, e.g. node name, attributes, methods, relation to super and sub nodes etc.

When the PVM is finished, a class diagram is developed (see figure 3). The diagram complies with the UML-notation (See also Felfernig et al. (2000) about using UML in the domain of product configuration) and can be drawn by using any diagramming tool, e.g. Visio. Again, a CRC-card exists for every class. The structure of the first class diagram and the PVM is very similar, but as the class diagram is further developed, it differs more and more from the PVM. From this point

forward in the development process, the class diagram (incl. the CRC-cards) serves as the documentation for the product model.

Experience from companies having applied the CPM procedure for building product configuration systems shows that it is a laborious and time-consuming task to maintain the diagrams and CRC-cards during both the development process and the subsequent maintenance process. Empirical experience from several industrial projects using CPM's procedure for product configuration (see e.g. Hvam et al., 2001; Hvam & Malis, 2002b; Hansen & Hvam, 2002) has stressed the need for an automated tool. The documentation task includes the handling of dependencies between CRC-cards and diagrams, consistency check, and version and release control. These tasks are usually carried out manually as no single IT-based tool has proved to be applicable. The Lotus Notes prototype described in Hvam and Malis (2002a) showed how some trivial tasks could be supported, but also revealed important areas where a Lotus Notes solution would be inadequate due to technical limitations (e.g. no support for a graphical representation of PVM and class diagrams owing to the lack of an adaptable graphical user interface).

As the model grows bigger and more complex, these tasks take up an unacceptable large part of the total modelling effort.

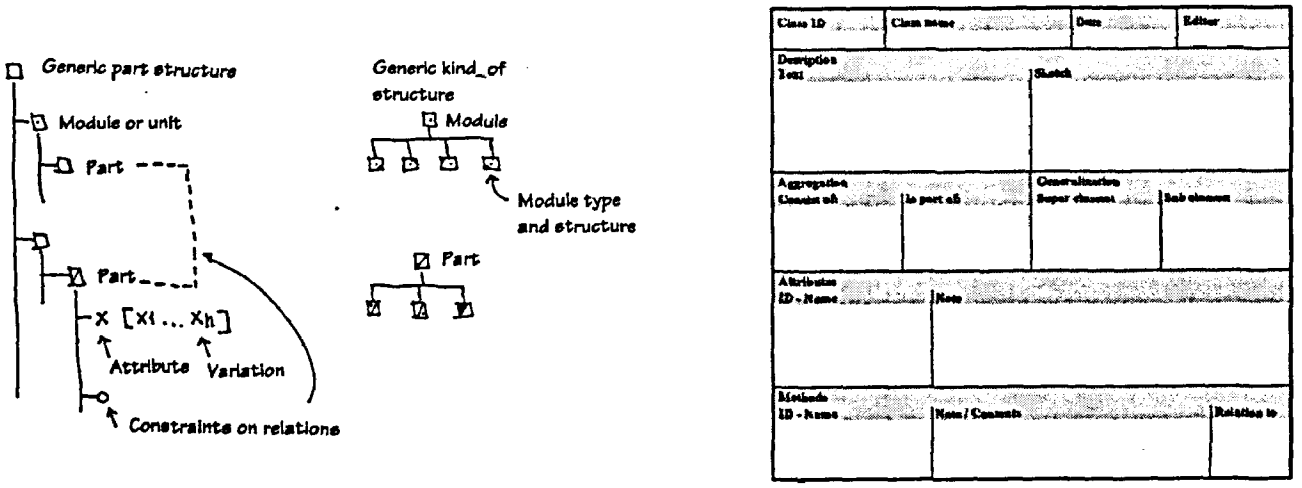

Figure 2: Left: Example of a PVM (Mortensen et al., 2000). Right: Example of a CRC-card (Hvam, 1994).

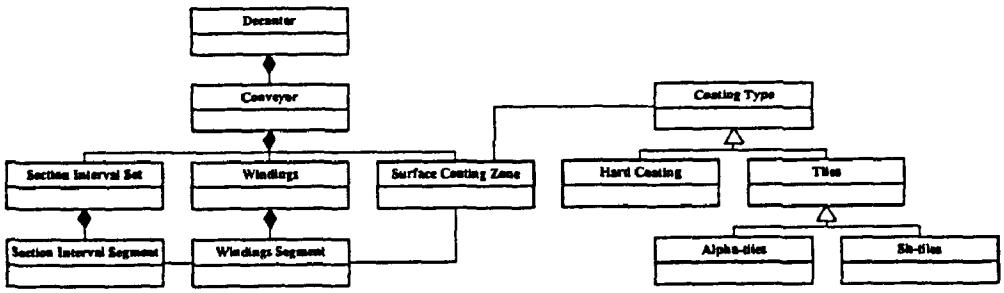

Figure 3: Example of a class diagram (Hvam & Riis, 2003). Each class is detailed in CRC-cards, as shown in figure 2 (right).

2.2 Interviews: Four Danish Manfacturing Companies

As indicated in section 1 the literature describes several reasons for using an automated tool to support the structured development process. CPM's growing experience from numerous product configuration projects applying the CPM procedure also emphasizes the need for a documentation tool to ensure a better documentation and liberate the product modellers from the time-consuming tasks of ensuring a consistent product model (change management, updating relations etc.). The product configuration process has been studied at four Danish companies (APC Corp. (power protection), GEA Niro (industrial drying), F.L.Smidth (cement plants) and Vitral (glass panels)).

Even though the product portfolio in the four companies strongly differs in complexity, functionality, price, etc., the experiences from the empirical work at CPM shows that the basic challenges in the development, maintenance and documentation process of configuration systems are comparable.

Three of the companies are using the CPM procedure and the fourth one has just recently become acquainted with the procedure. The product modellers were asked – bearing the procedure in mind – how a documentation tool could support the process. The experience and suggestions were evaluated and summarized in a set of superior requirements, describing the primary functionality of a documentation tool (see figure 4).

The companies did not all use all aspects of the CPM procedure for building configuration systems, despite the fact that they were all familiar with the procedure and acknowledged it. The general impression was that the procedure can enhance the process of developing and maintaining product configuration systems dramatically, but the manual documentation task requires too much effort as the model grows bigger and more complex. Thus, they settled for a less comprehensive and inadequate documentation. This choice was due to the lack of a documentation tool to automate the task, and the companies were confident that a documentation tool could make CPM's procedure easier to use and thereby enhance the overall product configuration process. In this way the tool and procedure form a whole with a symbiotic relationship.

Requirement	Description
One whole product model	A database containing all information from PVM, class diagram, and CRC-cards. The same information is often across a CRC-card and a PVM or a class diagram, but should be stored in a single place.
Version control	To manage several versions of CRC-cards, PVM's, and class diagrams – inclusive of their status ('working', 'approved' etc.).
Access control	Different users have different rights – well-known from other systems. This is used to prevent unauthorized changes of the product model.
Notification	For process management – e.g. to notify CRC-card editors about change suggestions from domain experts.
Easy to use	Especially domain experts must not see the tool as an obstacle – they shall be encouraged to use the tool.
Internet access	Users must be able to access the tool despite their geographical position. It is desirable to use a browser, but a local client is also acceptable.
Integration	Information from an existing systems (e.g. ERP and PDM) must be accessible/viewable (but not editable) from the tool.
Informal rules in CRC-cards	Domain experts must not be forced to learn a programming language to specify rules.
Hyperlinks	Both internal (e.g. a related CRC-card) and external (e.g. CAD drawings) electronic resources must be accessible via hyperlinks.
Flexible structure	The tool shall be easy to extend, and the integration to external systems in the specific company must be adaptable.
Configuration system integration	It must be possible to export (part of) the developed model to the configuration system.
Language	As English is the working language in many companies, the language of the tool must also be English – perhaps with multilingual support.
Inexpensive	A low-budget solution is preferred.

Figure 4: Superior requirements for the documentation tool (Pape, 2002).

3. REQUIREMENT SPECIFICATION FOR A DOCUMENTATION TOOL

The high-level processes and interactions for the future documentation tool is documented with use-cases, as described in section 1.1. Figure 5 shows an example of this approach. A top-level use-case diagram specifies the superior interactions and thus provides an overview of the documentation tool. The diagram contains several packages ('Administration', 'Product Variant Master', 'CRC-cards', and 'Class Diagram'), and each of these packages is then further detailed in separate diagrams as depicted in figure 5 for the package 'Product Variant Master'. A total number of 48 use-cases and a class diagram constitute the requirements of the documentation tool.

The set of use-case diagrams gives a good impression of what tasks the documentation tool is intended to support. Of course, the use-case diagrams alone do not explain in detail *how* the tasks are to be supported. Therefore, each use-case (an oval shape) is described in a table as shown in figure 6. In this example for the use-case 'Add part-of node/class' it is seen how the table describes the interaction between the user and the documentation tool – both in the most typical scenario and in alternative scenarios. In this way the table describes the condition *before* the use-case is invoked, what happens *during* the use-case, and finally the condition *after* the use-case is completed.

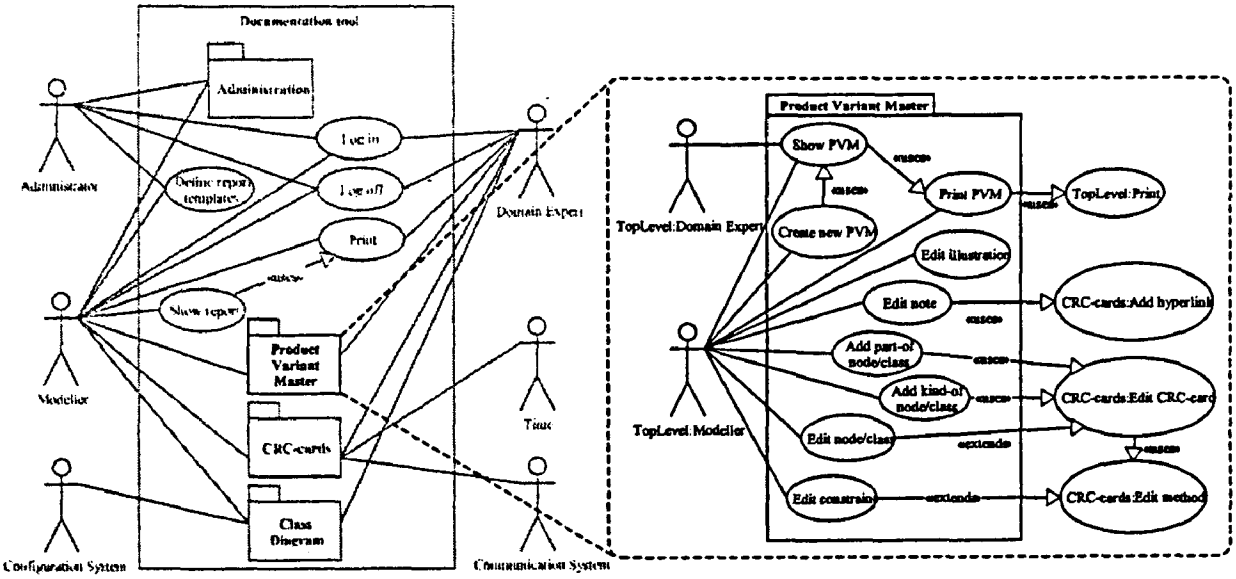

Figure 5: The top-level use-case diagram (left) is detailed in separate use-case diagrams (right) (Pape, 2002).

ID – Name	3.6 – Add part-of node/class «uses» CRC-cards:Edit CRC-card	
Reference	A21	
Actor(s)	Modeller	
Description	Part-of nodes/classes is added to the left side of the PVM-structure. If an existing node/class is selected before this use-case is initiated, the node/class is added after the selected one.	
Pre-condition	None, so far.	
Post-condition	A new node/class is added to the part-of structure of the PVM.	
Typical course of events	**Actor action** 1. The use-case is initiated when the actor right-clicks an existing node/class and selects 'Add sub-node'.	**System response** 2. A new class is added in the product model database as a subclass to the one selected (composition relationship). Next, CRC-cards:Edit CRC-card is invoked. 3. The PVM is redrawn when all information is entered. The changes are stored in the log.
Alternate courses	**Step 1a:** Can also be initiated by right-clicking in an empty PVM and selecting 'New top-node'. **Step 1b:** Can also be initiated by right-clicking an existing node/class and selecting 'Add new node below'.	**Step 2a:** The first class in the PVM is created in the product model database. All subsequent classes are created as subclasses to this class. Continue from step 3 above. **Step2b:** A new class is created in the product model database at the same level (same superclass) as the one clicked on. Next, CRC-cards:Edit CRC-card is invoked, and the use-case continues from step 3 above.
Assumptions	None, so far.	

Figure 6: Each use-case is described in detail without focusing on implementation aspects (Pape, 2002).

3.1 Architecture of the Documentation Tool

To illustrate the structure and functionality of the documentation tool in a simple and easy-to-read model, the architecture of the documentation tool (see figure 7) has been worked out based on the use-cases and the class diagram. As depicted in

figure 7 the requirements are prioritized to serve as a basis for a phased development of the tool – the lower the number, the more important is the functionality of the module. The modules are described below.

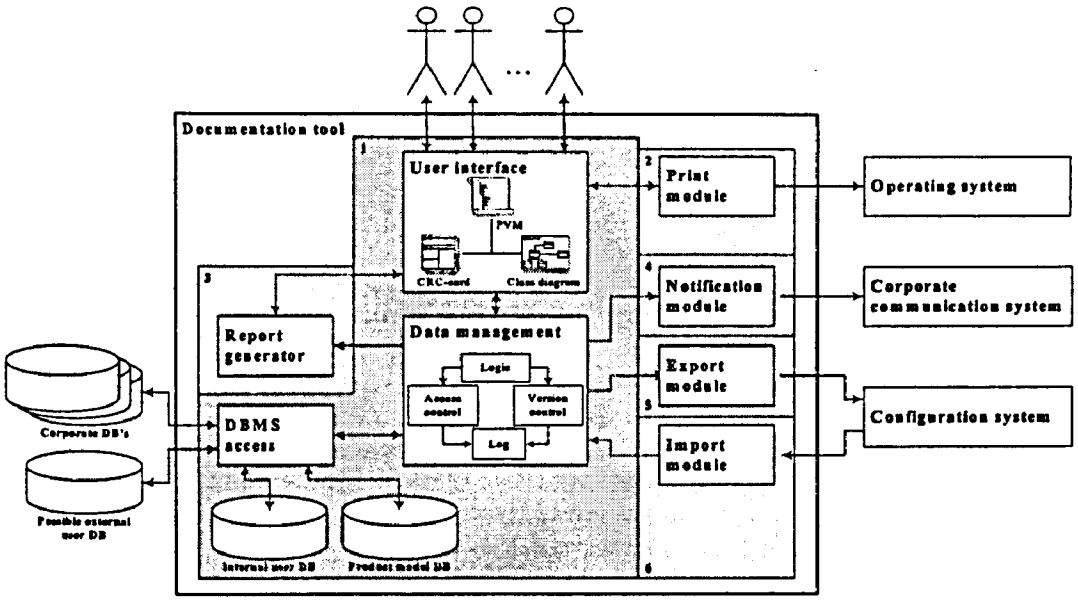

Figure 7: The architecture for the documentation tool, prioritized with no. 1 as the highest priority (Pape, 2002).

Product model database (DB)
The core element in the tool is a database where all product information is stored. Information from the database is used across PVM's, class diagrams and CRC-cards to ensure integrity. For example, a class diagram may show information about a class' name, its attributes, methods etc., and this information is also shown on the CRC-card for the particular class. The documentation can easily become inconsistent if e.g. changing an attribute requires changes in both a class diagram and a CRC-card. Therefore, information must be stored in a database system.

User database
Every single element in the documentation (class diagram, CRC-card etc.) is owned by a user who is responsible for the maintenance of the element. For each element different access privileges can be assigned to different users or groups of users. The user management system can be enhanced by using an existing corporate user management system.

Corporate databases
Product information is often available in corporate databases (e.g. an ERP-system). This information can be accessed by the documentation tool and viewed by the user in e.g. a CRC-card. However, the information is not editable from the tool.

Data management
Covers Access Management (different users have different access privileges), Version Control (e.g. what version of a CRC-card have the status of 'implemented'), Log (all activity must be traceable – "what, who, and when"), and Logic (business rules and transaction specifications – e.g. that suggestions for changes trigger a notification of the owner of the class).

User interface
The main concept is to use the PVM and class diagrams as central elements of the user interface. This ensures that working with the tool is very similar to working with the diagrams on paper, just faster and easier.

Database Management System (DBMS) access
A DBMS module handles all communication to internal as well as external databases by using standardized communication protocols. This ensures adaptability to various database systems.

Export and import modules
An export module is required to export the data from the documentation tool to a standard configuration system (like e.g. iBaan, Oracle, Siebel, and Cincom) with various data formats. No single intermediate format satisfies all systems. Likewise, an import module converts data the other way.

Report generator
Information can be retrieved from the documentation tool with reports. A report generator can generate reports based on the information in the product model database and associated databases. For example, a list of all attributes can be generated which is sorted in ascending order by name and with information about which class that "owns" them. The users can design different report templates.

Notification module
The tool can be configured to let different events trigger a notification of the relevant users and hereby support the working process in the company by using an existing corporate communication system (e.g. e-mail), for example by sending an e-mail to a class owner about a pending change suggestion from a domain expert.

Print module
Reports, diagrams, and CRC-cards can be printed. Printing PVM's and large class diagrams requires support for large-format plotters and/or printing of large documents on normal A4-paper as "tiled pages".

3.2 Making the elements of the documentation "intelligent"

The documentation of a product model consists of three major elements (as described in section 2.1): 1) PVM, 2) CRC-cards, and 3) class diagrams. When handling the documentation manually you lack the advantage of linking the elements dynamically and of generating and updating information automatically. This is one of the main reasons why a manual process is tedious and counterproductive.

Figure 8 illustrates an example of how an automated tool can enhance the process. The tool can generate the content of all the fields marked with hatching from existing information. For example, the date can be captured from the system clock when the CRC-card is created, and the information in the 'Aggregation' and 'Generalization' fields can be generated from the relations between the classes in the PVM or class diagram. In addition, the tool can update the information in these fields automatically.

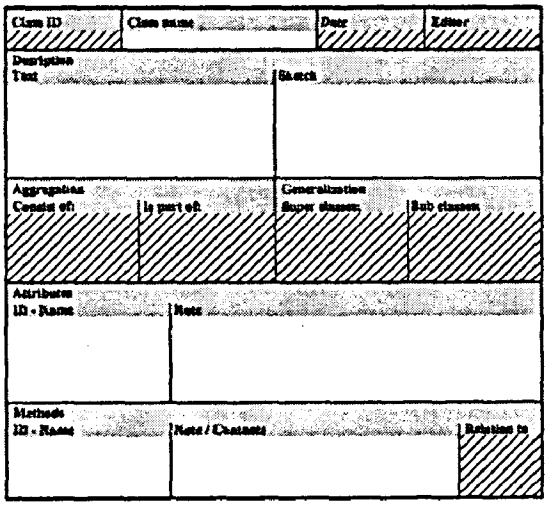

Figure 8: Information in one half of the fields in a CRC-card can be generated automatically.

The tool can maintain a lot of the information in the diagrams in a similar manner, e.g. by linking the name of classes to the same field in the product model database as the content of the 'Class name' field in the corresponding CRC-card is linked to. This concept eases the task and ensures a more consistent model.

Furthermore, the diagrams can be made "active" and navigable when using an automated tool, for example by opening the corresponding CRC-card when clicking on a class in a PVM, or by browsing a sub class diagram when clicking a box in a class diagram.

3.3 From documentation to configuration system

The storing of the product model in a single "knowledge repository" also enables a streamlining of the process of designing a configuration system from the creation of a PVM to the maintenance of the final class diagram. The tool can, for example,

automatically generate a suggestion for a class diagram from an existing PVM, something which so far has been done manually.

A more interesting feature is the opportunity to exchange information with a configuration system. Four configuration systems (Oracle Product Configurator, Cincom Knowledge Builder, Baan Configuration 98, and iBaan Configurator) have been examined with focus on data exchange, and the result was that it is possible to exchange structured product data with all configuration systems, though it would be necessary to adapt an import/export module to every single configuration system. With an export module it will be possible to export product model data to the configuration system directly, instead of typing class names, attributes etc. into the system manually. An import module will enable the tool to read the current implementation in the configuration system and to compare this with the current documentation, and highlight any discrepancies.

4. CONCLUSION

The task of handling the documentation of a product model has shown to take up too much time of the process when developing a product configuration system, as the product models grow more complex. Experience from the field of product configuration has proved that an IT-based documentation tool can enhance the process by eliminating the bottleneck of administrative tasks. The concept of such a documentation tool is very similar – but not identical – to CASE-tools and PDM-systems and the effect is therefore well-known from other domains.

Requirements and a concept for a documentation tool for the development of a product configuration system have been presented based on CPM's procedure for building configurations systems.

Two general scenarios exist for the further development. The first is to let providers of standard configuration systems integrate documentation facilities in their systems. An example of this is seen in the iBaan configuration system where some simple documentation facilities have been built into the system, but it is far from sufficient. This scenario eliminates the need for import/export modules, but as a result the user may have to make a trade-off between the functionality of the configuration subsystem and the functionalities of the documentation subsystem.

The other scenario is to have providers of non-configuration systems develop a dedicated documentation tool which is independent of configuration systems. This approach gives a liberty to choose configuration systems more freely, but it will require more sophisticated import/export modules to be part of the documentation tool. Another advantage is that an external tool can be developed which has absolute focus on the procedure for building configurations systems (e.g. based on CPM's generic procedure) and which is not based on the limitations of an existing configuration tool.

5. REFERENCES

1. Barker, Virginia E. & O'Connor, Dennis E (1989). Expert Systems for Configuration at Digital: XCON and Beyond. Communications of the ACM, March 1989, volume 32 Number 3.
2. Bellin, D. & Simone, S. S. (1999). The CRC-card Book. Addison-Wesley.
3. Bennett, S., McRobb, S. & Farmer, R. (1999). Object-Oriented Systems Analysis and Design using UML. McGraw-Hill.
4. Booch, G., Rumbaugh, J. & Jacobson, I. (1999). The Unified Modeling Language User Guide. Addison-Wesley.
5. Cawsey, A. (1998). The Essence of Artificial Intelligence. Prentice Hall.
6. CIMdata (1997). Product Data Management: The Definition. 4th edition. CIMdata Inc.
7. Coad, P. & Yourdon, E. (1990): *Object Oriented Analysis (2nd Edition)*. Prentice Hall
8. Felfernig, A., Friedrich, G. E. & Jannach, D. (2000). UML as Domain Specific Language for the Construction of Knowledge-based Configuration Systems. In International Journal of Software Engineering and Knowledge Engineering, volume 10/4.
9. Hansen, B., Hvam, L. (2002). Design of Configuration Systems at APC. Proceedings of 15th European Conference on Artificial Intelligence. Lyon, France.
10. Hvam, L. & Malis, M. (2002a). A Knowledge Based Documentation Tool for Configuration Projects. In Mass Customization for Competitive Advantage (Ed.: M.M. Tseng and F.T. Piller).
11. Hvam, L. & Malis, M. (2002b). Product Configurators as a means to support the Exchange of Knowledge in Company Networks. Proceedings of the 7th Annual International Conference On Industrial Engineering. Busan, Korea.
12. Hvam, L. & Riis, J. (2003). CRC-card for product modeling. Computers in Industry, volume 50/1.

13. Hvam, L. (1994). Application of product modelling – seen from a work preparation viewpoint. Ph.D. Thesis. Dept. of Industrial Management and Engineering, Technical University of Denmark.
14. Hvam, L. (1999). A procedure for building product models. Robotics and Computer-Integrated Manufacturing, 15: 77-87.
15. Hvam, L., Riis, J., Malis, M. & Hansen, B. (2000). A procedure for building product models. Proceedings of Product Models 2000-SIG PM, Linköping, Sweden.
16. Hvam, L., Riis, J., Malis, M. & Hansen, B. (2001). Reengineering of the quotation process – Application of knowledge based systems. Proceedings of the International Conference on Industrial Engineering and Production Management. University Laval, Canada.
17. Krause, F.L., Kimura, F. & Kjellberg, T. (1993). Product Modelling. In Annals of the CIRP, volume 42/2.
18. Mortensen, N.H., Yu, B., Skovgaard, H.J. & Harlou, U. (2000). Conceptual Modeling of Product Families in Configuration Projects. Proceedings of Product Models 2000-SIG PM. Linköping, Sweden.
19. Pape, S. (2002). Requirements for Documentation Tool for Product Modelling. Master's Thesis (ID: IPL-128-02). Dept. of Manufacturing Engineering and Management, Technical University of Denmark.
20. Schwarze, S. (1996). Configuration of Multiple-Variant Products. BWI, Zürich.
21. Tiihonen, J., Soininen, T., Männistö, T. & Sulonen, R. (1996). State-of-the-practice in Product Configuration – a Survey of 10 Cases in the Finnish Industry. In Knowledge Intensive CAD, volume 1 (Ed.: Tomiyama, T., Mäntylä, M. & Finger S.). Chapman & Hall, pp. 95-114.
22. Whitten, J.L, Bentley, L.D. & Dittman, K.C. (2001). Systems Analysis and Design Methods. Irwin/McGraw-Hill, New York, USA.