Creating a documentation system to support the development and maintenance of product configuration systems

ANDERS HAUG[1], ANDERS DEGN[2], BJARNE POULSEN[2], LARS HVAM[1]
[1] Department of Industrial Engineering and Management
[2] Department of Informatics and Mathematical Modelling
Technical University of Denmark
[1] Building 425, 2800 Kgs. Lyngby
[2] Building 322, 2800 Kgs. Lyngby
DENMARK
[1] {ahaug, lhv}@ipl.dtu.dk www.productmodels.org
[2] bjp@imm.dtu.dk

Abstract: A product configuration system (PCS) can be defined as a product-oriented expert system that allows users to specify a product while restricting how different elements and properties may be combined. The use of configuration technology has in several cases led to improvements of product specification processes, such as shorter lead times, reductions of resources needed, and fewer errors.

A procedure for building product configuration systems from the Centre for Product Modelling at the Technical University of Denmark has been applied in projects for more than ten years. The CPM-procedure includes three main modelling techniques to support the development and maintenance of PCSs. However, no software, which supports all three techniques in an integrated fashion, currently exists. This means that when developing PCSs based on the CPM-procedure there is no automatic integration between the created models, wherefore some information has to be transferred between models manually. CPM has, therefore, for some years worked on creating a basis for developing a documentation system that supports the development and maintenance of PCSs. Research focusing on the requirements for a documentation system has been produced, and more recently detailed definitions of the included modelling techniques have emerged. This paper describes how these definitions have been converted into a software prototype and what have been learned from the evaluation of the prototype.

Keywords: Product configuration, Knowledge engineering, Documentation of product models, Class diagrams, Product variant master, CRC-cards

1 Introduction

The use of configuration systems has for a number of years been a successful application of artificial intelligence techniques [1]. A product configuration system (PCS) can be defined as a product-oriented expert system, which by applying knowledge of a domain, lets the user specify a product under restriction of valid combinations of product components and properties. In many cases, the application of configuration technology has led to a range of improvements, such as reductions in lead times, number of errors, and use of resources in the specification of products [2, 3, 4, 5, 6].

The representation of domain knowledge is often one of the greatest tasks in a PCS project [7, 8, 9]. In a PCS project knowledge is often represented in two distinctive kinds of models, namely analysis and design models. The models that describe the knowledge of a domain (in this context product knowledge) without considering implementation aspects are called analysis models (or domain models and conceptual models). When creating analysis models, besides describing the knowledge of a domain, also a need for defining new knowledge can exist, as when a PCS project is started, sometimes the included product families do not have a suitable architecture to form a basis for the creation of a generic product knowledge model. The creation of analysis models can therefore require great involvement of the relevant domain experts of a company. Based on the analysis models, design models can be created in order to facilitate the implementation of the domain knowledge in the PCS. Design models can be seen as formalised and possibly further detailed versions of analysis models that take implantation issues into consideration. This means that the requirements for the analysis and the design language are different, in that the analysis

language has to be adequately simple in order to be understood by the domain experts involved, while the design language has to be adequately rich and formalised in order to make accurate descriptions of what is to be implemented.

In many cases where manufacturers of customised industrial products apply configuration technology, the knowledge base of the PCS consists of thousands of classes, attributes, constraints, and methods. When a product changes, the knowledge base of a PCS has to be updated in order for the PCS to support the company processes. As the modelling environment of a PCS seldom provides an adequately comprehensible overview for the domain experts to understand the current model that is to be changed, these kinds of updates often require external documentation. Unless the models created as a basis for the development of the PCS have been maintained, these have to be updated or new models created.

The mentioned aspects present a demand for a coherent documentation system that supports the applied modelling techniques in order to avoid tasks such as: manual transfers of information between models, reconstruction of models, and ensuring consistency across models.

Based on an often applied procedure for the development of PCSs from the Centre for Product Modelling (CPM) at the Technical University of Denmark, research on how to create a documentation system has been carried out. However, so far this research has not resulted in systems that include all the three main modelling techniques of the CPM-procedure. This paper describes the creation and evaluation of the first prototype capable of supporting these three techniques in an integrated manner.

The rest of this paper is structured as follows: Firstly, in section 2, the mentioned procedure for the development of PCSs is outlined with a focus on its techniques for the creation of analysis and design models. Next, in section 3, research concerning a documentation system to support the development and maintenance of PCSs is resumed. In section 4 a prototype developed on the basis of the research carried out is described. Section 5 describes the evaluation of the prototype. The paper ends with a conclusion in section 6.

2 The CPM procedure

To carry out a PCS project is often a great task, both in relation to the change of business processes and in relation to the creation of the PCS itself. To support these tasks, in 1994 Hvam [10] presented a procedure for the development and maintenance of PCSs. This procedure has since continuously been updated at CPM, as experience with the procedure has been obtained. The CPM-procedure or part of the procedure has been applied in a number or cases in Danish industry [2, 3]. The CPM-procedure in its current form consists of seven phases to support the course of a PCS project: 1 Process analysis, 2 Product analysis, 3 Object-oriented analysis, 4 Object-oriented design, 5 Programming, 6 Implementation, and 7 Maintenance.

The CPM-procedure prescribes the use of three major techniques for the creation of analysis and design models, where product variant masters (PVM) are prescribed for the creation of analysis models, class diagrams for the creation of design models, and CRC-cards (Class, Responsibility and Collaboration) for making detailed descriptions of the classes in PVMs and class diagrams. However, in some cases PVMs provide an adequate basis for implementation of the product knowledge into a PCS, which, obviously, means that class diagram models do not need to be elaborated [3].

2.1 PVMs

A PVM is a diagram that is applied for describing generic product models. A PVM consists of two sections that describe part-of structure and kind-of structure of a product model. These two structures, in object-oriented terms, roughly correspond to aggregation and generalisation respectively. Class descriptions, attributes, and constraints can be stated below the classes in a PVM. In figure 1 the latest definition from CPM of the notation formalism of a PVM is shown.

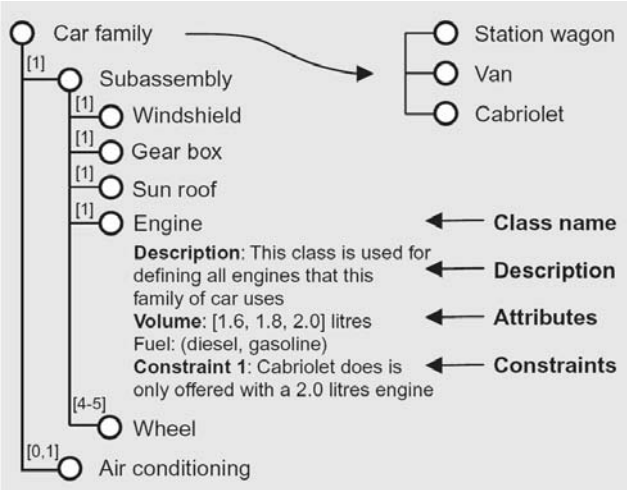

Fig.1. The most recent definition of the PVM formalism [3] from [11]

PVM models are intended to be discussed and refined in repeated sessions of knowledge engineers and domain experts until the descriptions of the product assortment are adequately extensive and detailed. The experience with the use of PVMs in configuration projects is that PVMs can be a strong tool for discussing and defining the product assortment [2]. As no software distinctly aimed at the creation of PVMs exists, PVMs are normally elaborated by the use of programs such as MS Excel or MS Visio.

2.2 Class diagrams

The Unified Modelling Language (UML) is a modelling language with a formal syntax that is defined by the Object Management Group (OMG). UML 2.0 defines thirteen types of diagrams, where one of these is the class diagram [12]. Class diagrams are used for describing the objects in a system together with the various kinds of static relationships among them [13]. Compared to PVMs, class diagrams are far more widespread in use and have a much broader range of application. The notation for the class element and the most common relationship types are shown in figure 2, where a *navigability arrow* can be used to show the direction of association, aggregation, and composition relationships. For more detailed descriptions of class diagrams, see [12, 13].

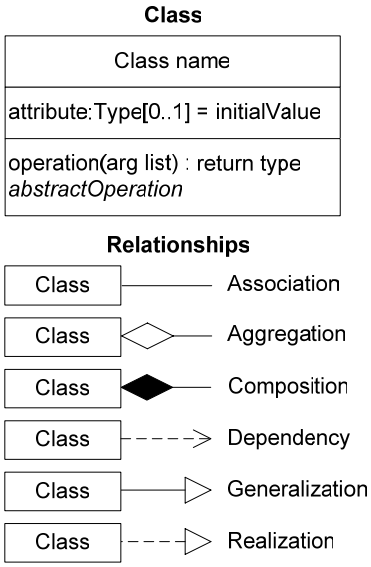

Fig.2. Commonly applied class diagram elements

The CPM procedure prescribes the use of a sub-selection of the relationship types of class diagrams, including generalisation, aggregation, and association [3]. The argument of the CPM-procedure for including both PVMs and class diagrams, which basically are two ways of illustrating the same thing, is that PVMs seem to be more easily understood by domain experts with limited modelling prerequisites, while class diagrams provide a much richer and more formal language, which makes these better suited for the creation of design models [3].

2.3 CRC-cards

The CPM-procedure proposes the use of CRC-cards for holding detailed descriptions of classes in PVMs and class diagrams in order to enhance the clarity of such models. For this purpose, CPM has provided a definition of special CRC-cards to be used in PCS projects. While the original CRC-cards by Beck and Cunningham [14] only include the class name together with two columns for responsibilities and collaborators, the CRC-cards of CPM have been extended with fields for stating: author, date, superparts, subparts, superclasses, subclasses, attributes, and more [2, 3].

3 Towards a documentation system

As mentioned, the CPM-procedure prescribes the use of PVMs, class diagrams, and CRC-cards during the development and maintenance of PCSs. As there is no software available, which supports all three techniques in an integrated fashion, different kinds of software are applied. This implies a need for manual transfers of information between PVMs, class diagrams and CRC-cards, which are time consuming and hold risks of errors. Documentation of the knowledge base of PCSs is therefore an aspect of product configuration that has been subject to some investigations.

The experience with product configuration in twelve Danish companies was investigated in a research project in 2003 and 2004 [15]. The investigations showed that the documentation task was often the first to be given a lower priority or even cut out of the project. It was also pointed out that a lack of documentation of the PCS knowledge base can have very negative consequences, as some companies were unable to further develop their PCSs. Another effect of the lack of documentation was a negative impact on the daily communication between people involved in product development and PCS development respectively.

As models grow in extent and complexity, the effort required for the creation of documentation

increases. It is the general impression of Hvam et al. [16] that many companies settle for less comprehensive documentation than what is actually needed in order to be able to further develop a PCS. They further argue that this to some extend could be avoided by the presence of a documentation system that supports the maintenance of PCS documentation, which would relieve the companies from the time-consuming task of ensuring consistent product models.

3.1 Requirements for the system

As a response to the apparent need for more advanced software for the documentation of the knowledge base of PCSs, research on this topic has been carried out. Based on a survey of five standard configuration systems and the experience gained from several Danish configuration projects, Hvam and Malis [17] define the requirements for a documentation system: easy to maintain, facilitates the modelling techniques of the CPM-procedure (PVMs, class diagrams, and CRC-cards), has central storage of data, supports network distribution of data, supports multiple user access, integrates the modelling techniques, includes version control, and allows integration with PCSs. Furthermore, they describe the development of a prototype, created in Lotus Notes. This prototype has since been further developed and is today applied by the companies GEA Niro A/S and American Power Conversion A/S (APC). Although the use of the Lotus Notes based documentation system has shown that there are significant benefits from applying such a tool for the maintenance of PCSs [2], much is still to be wanted. From a modelling point of view the documentation system fails to offer support for the elaboration of class diagrams and PVMs, but only includes CRC-cards and a hierarchical list of classes that does not show attributes, constraints, and kind-of structure/generalisation. Consequently, both the mentioned companies, who use the Lotus Notes application, apply other software for the creation of PVMs when making big changes or additions to their existing models.

Based on [17] and interviews with four Danish manufacturing companies who apply configuration systems, Hvam et al. [16] propose an extended list of requirements for a documentation system to support the development and maintenance of PCSs. This includes: a coherent product model, version control, access control, change notification, user-friendliness, web-based access, integration to other software systems, possibility of informal rule expressions, hyperlinks to internal and external files, flexibility, configuration system integration, the use of English as language, and an inexpensive solution. Furthermore, a high level description of a possible architecture of such a documentation system is presented in [16].

3.2 Why has the system not been created?

Having established that seemingly there is a need for a documentation system to support the development and maintenance of PCSs, and research that deals with defining such a system has been produced, it could seem strange that a complete documentation system does not exist at present. Based on the experience (of the first author) from studies of several configuration projects, the following possible explanations are offered:

1) While the creation of a documentation system to support the development and maintenance of PCSs seems to require a significant deal of software development, it is presently unclear how many potential customers there will be. It seems that the companies who show the greatest interest are the ones who have a great need for external documentation of the knowledge base of the PCS. These can typically be characterised by having extensive and complex PCS knowledge bases and having to describe product knowledge that is possessed by others than the ones implementing the knowledge in the PCS. However, the number of these kinds of companies in Danish industry is limited. But if a relatively inexpensive documentation system emerged, companies with less need for external documentation of their PCSs may show greater interest.

2) The companies who apply PCSs apply different modelling techniques when documenting their PCSs. Even those who base their development on the CPM-procedure make individual adaptations. Therefore, creating a system that is adequately flexible to support the different kinds of uses could turn out to be very difficult.

3) Although research concerning the creation of a documentation system [16, 17] specifies which modelling techniques should be included, this research does not in a detailed manner deal with topics like: user interface design, detailed definitions of the included modelling techniques, and how to handle interrelated models. An important part of the basis for developing a documentation system to support the development and maintenance of PCSs has therefore been missing until recently [18].

3.3 Definition of a documentation system

Addressing the need for adequately flexible and software prepared notation formalisms of PVMs, class diagrams, and CRC-cards in order to include these in a documentation system, Haug and Hvam [18] present such a definition. This includes three main views, namely a CRC-card view, a PVM view, and a class diagram view. In the documentation system it should be possible to switch between these views dependant on what level of detail and what kind of information the users are interested in. The three views are shown in figure 3.

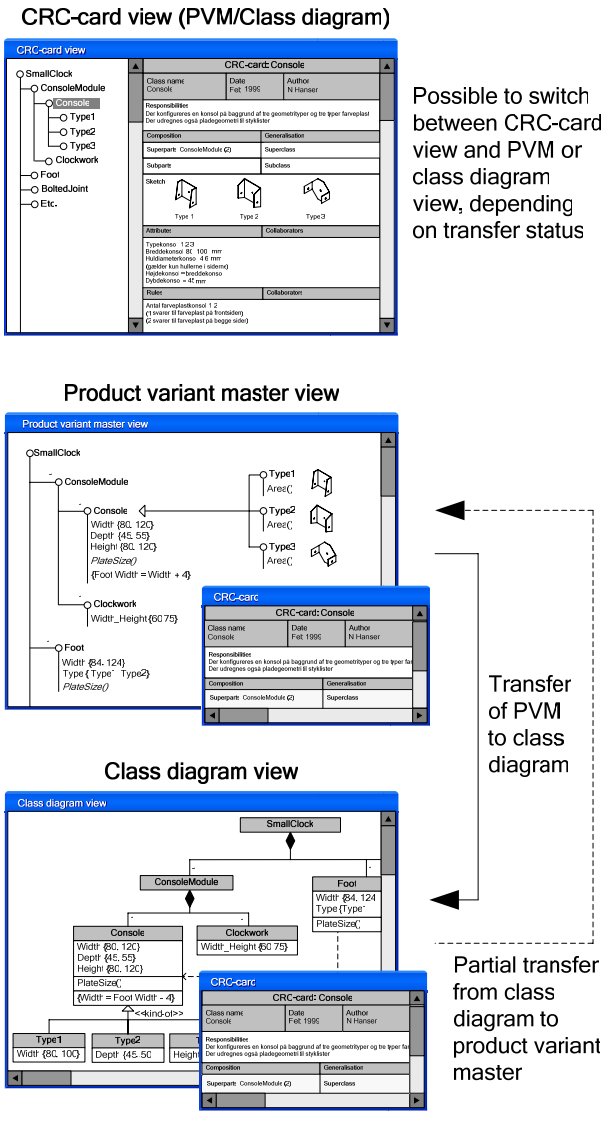

Fig.3. The three main views in the documentation system (adapted from [18])

Yet another step towards the creation of a documentation system that supports the CPM-procedure was taken by Haug and Hvam [19], who, based on the current definitions by CPM [3], present a revised definition of the CRC-card layout, which takes into account future software supported elaboration of the CRC-cards. The new layout includes several new fields, e.g. for documenting additional relationship types and for organising information about attributes, constraints, and methods. Another extension of the CRC-card layout concerns the inclusion of fields for managing change requests and versions.

4 The creation of a prototype

Besides supporting the elaboration of PVMs, class diagrams and CRC-cards in an integrated fashion, the documentation system should include functionalities similar to the ones of product data management (PDM) systems in order to support the development and maintenance of PCSs in a satisfactory manner. While PDM-related functionalities have been included in numerous software systems, a system which supports and integrates PVMs, class diagrams, and CRC-cards does not exist. Creating a prototype to evaluate this kind of modelling environment was therefore a natural starting point. Based on the definitions in [18, 19], a prototype including the three main views was created. Besides evaluating the defined modelling techniques, another main point was to find a solution principle for handling the model elements which are represented differently or only present in some of the three views.

It was chosen to create the prototype using MS C# .Net, as the Microsoft .Net platform supports Windows software development and because it was considered to ease object-oriented development, offer a development environment that is easy to use, and have good debugging functionality.

As mentioned, an important argument for the creation of a documentation system is to avoid the manual transfers of information between PVMs, class diagrams, and CRC-cards. The three techniques, however, do not only include the same information, but also have individual information, e.g. is the *Sketch/Picture* field of the CRC-card view not included in the class diagram view. One possibility is to create a solution where information is transferred between the three kinds of representations when making changes. This, however, would result in a need for high memory, and redundancy in the stored data. To avoid this, the prototype has only one data model for all three diagrams, which means that the three kinds of representations are different views of the same data. This for instance means that when changing a part of a class diagram model this information would automatically be changed in the corresponding PVM

model and CRC-cards if these include this specific type of information. The data model of the prototype is depictured in figure 4, where the attributes and methods are not shown due to lack of space.

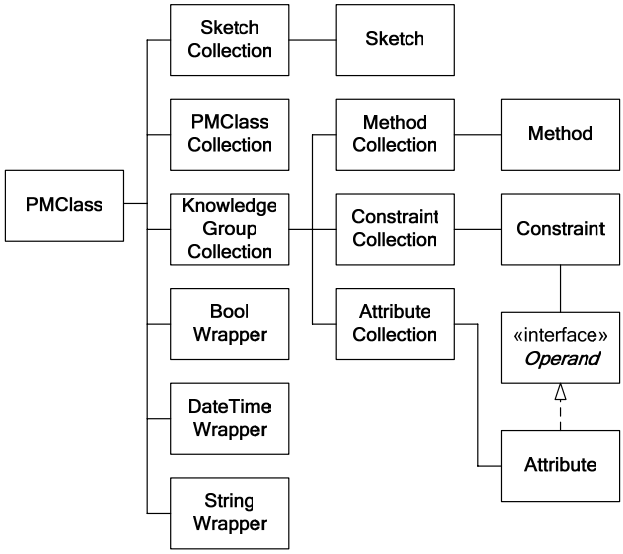

Fig. 4. The common data model

While the three views of the prototype are bound to the data model, the data-model on the other hand, is not bound to any of the views. It would, therefore, be possible to add, change and remove views of the prototype without adding or deleting classes in the data model.

In its current form the prototype displays the same user-defined classes and relationships in the PVM view and the class diagram view. However, in a further developed version of the documentation tool it should be possible to associate visual content of the data model to particular views. This would among other things be useful in contexts where the PVM view is used for creating an analysis model, and where the class diagram view is used for creating the corresponding design model. For instance, an analysis model could include classes that are not going to be implemented, for which reason these should not be part of the design model. The other way around, the design model could include implementation-oriented classes that are irrelevant in the analysis model, which, therefore, should not be shown in the PVM view.

CRC-cards can be accessed from any of the three views; in the CRC-card view by clicking on a class in the hierarchical list, and in the PVM view and class diagram view by double-clicking on a class. In figure 5 the CRC-card view of the prototype is shown. The CRC-card view consists of a hierarchical list together with a chosen CRC-card. In the hierarchical list, aggregation and inheritance are shown by different symbols, a white node if it is an aggregation class and a black node if it is a specialisation class.

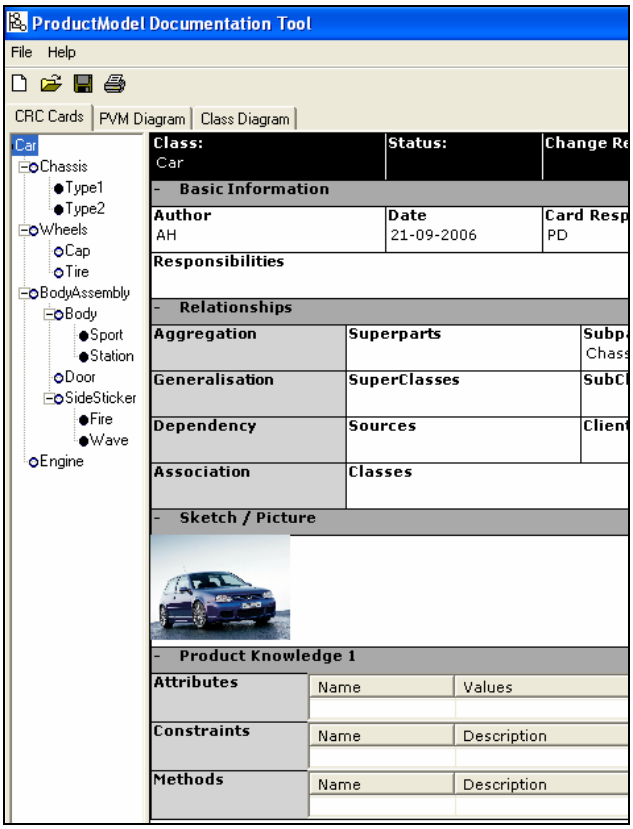

Fig.5. CRC-card view (part of the window)

In the hierarchical list of the CRC-card view classes can be moved, copied or linked by using the standard MS Windows functionalities of drag-and-drop and context menus. Moving a class from one place to another in the hierarchical list means a change in the class' relationships with other classes. This kind of change is automatically updated in the relationship-fields of relevant CRC-cards and on the matching PVM and class diagram. When copying a class, an identical copy is created, which can be pasted anywhere in a given model. A copied class represents a new class in a model, for which reason it must be given a new and unique name. Changes done to the original class or the copied class does, therefore, not affect the opposite. A class may also be linked to another class. Thereby the same class appears several times in a model. Using linked classes means changes to the class at one place in the product model automatically affect the linked classes. This can be useful when a model includes components that are parts of different assemblies, and where these should hold the same information at

every place they appear in the model. For instance, if a class *Screw* appears several times in a model, and new screw-dimensions must be introduced, this only has to be done once.

In figure 6 and figure 7 the PVM view and the class diagram view is shown.

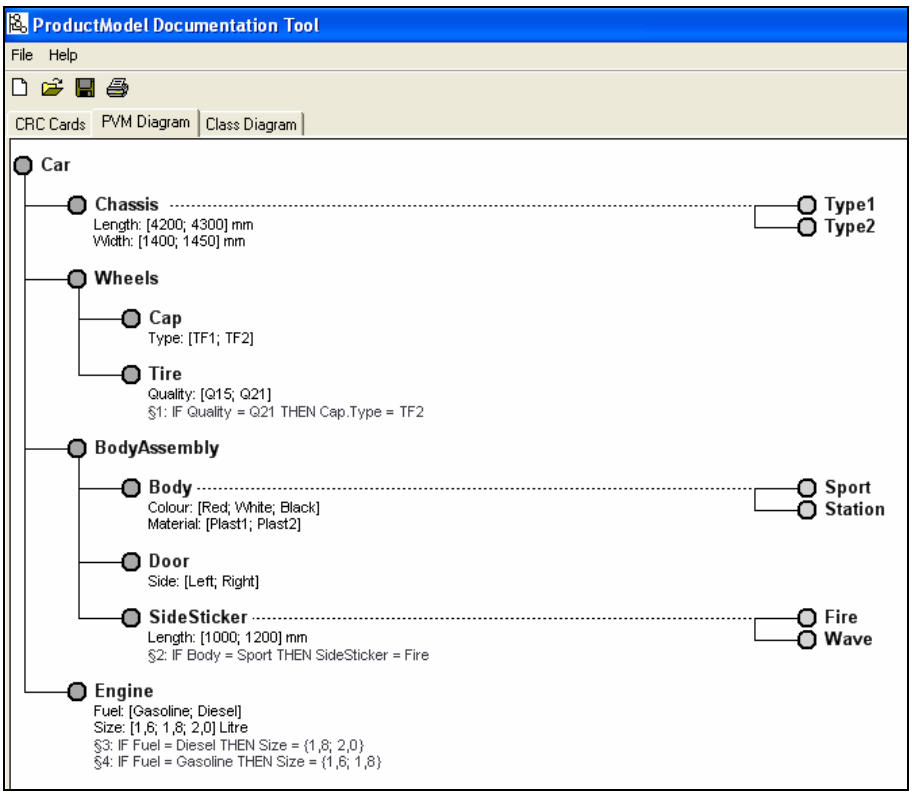

Fig.6. PVM view

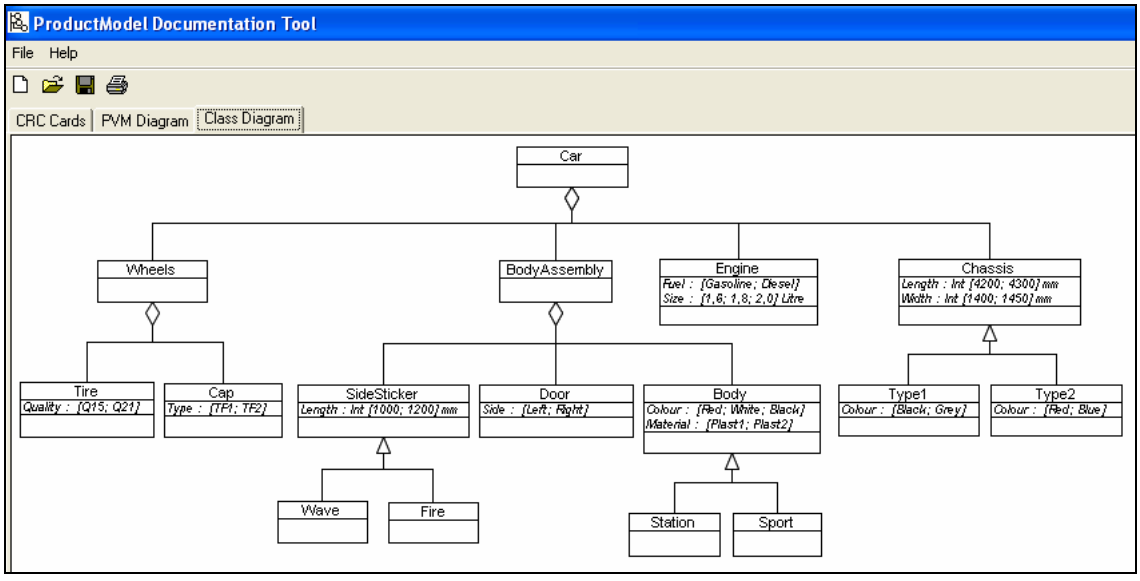

Fig.7. Class diagram view

The elements of the class diagram are created automatically when defining a model in either the CRC-card view or the PVM view. However, placing the classes must be done manually by the user, while the system remembers this the next time the class diagram view is accessed. To increase the user friendliness of the system, a future version should include placement rules for the class diagram view.

In the class diagram view classes can be moved by dragging them to any desired place in the

diagram. It is also possible to move several classes at the same time by dragging a box around them. Resizing of classes can be done by dragging the corner points, which are shown when the cursor moves over a class. Like in the PVM view, classes can be created by right-clicking on a class and selecting to insert a new class with a chosen relationship type.

The prototype is capable of saving a created model to a file, as well as printing out the diagrams and CRC-cards of the different views.

The prototype was developed from February to May 2006, and approximately 200 hours were spent designing the software architecture and programming the prototype.

5 Evaluation of the prototype

To evaluate the prototype, two kinds of investigations were carried out. Firstly, data from an ongoing configuration project was put into the prototype to investigate possible limitations of the modelling environment of the prototype. Secondly, the prototype was presented to two companies in order to compare the prototype with their current documentation software and to get other kinds of feedbacks.

5.1 Testing the prototype

The PCS-project, which provided the product model that was put into the prototype, concerns the creation of a PCS to support the creation of tender documents, manufacturing drawings, bill of materials etc. in projects concerning balconies for existing buildings. While the project is ongoing, the company wishes to remain anonymous. The product knowledge in the ongoing project was represented in PVMs that had been created on the basis of the notation shown in figure 1, but with some minor extensions.

As regards the CRC-card view, the experiment showed a need for better possibilities for typing in comments, wherefore more such fields should be included in a final documentation system. In the models from the ongoing project many of the constraints were represented in table form. As the prototype at its current stage only supports the writing of textual expressions, the constraints represented in tables had to be transformed. To avoid this, the final documentation should support the use of tables.

The PVM view allowed putting in the model information from the ongoing PCS-project. However, in the PVM models from the ongoing PCS project, constraints and comments were not placed below classes as defined in the prototype, but were instead placed in boxes near classes. Although the ones, who are going to use the system, properly could be forced to place this kind of information below classes, it seems reasonable to allow the use of boxes in a final documentation system.

The class diagram view was also capable of holding the information from the PVM models from the current case. However, in a final documentation system it should be possible to: turn off some of the contents of a model, insert boxes for comments or constraints, and apply more types of class diagram relationships, as defined in [18].

5.2 Presenting the prototype to users

The prototype was presented to two Danish manufacturing companies, who both have several years of experience with the use of PCSs. The two companies were chosen due to the extensiveness and complexity of their PCSs, as this often implies a need for the creation of external documentation in order to overview the implemented product knowledge.

The first company the prototype was presented to was GEA Niro A/S. Niro is part of the GEA group and is an international engineering company that has a leading market position within the area of design and supply of spray drying plants. Niro primarily use their PCS to support the elaboration of tenders and, as mentioned, apply a Lotus Notes solution for documenting the knowledge that is implemented in the PCS. Sometimes PVMs, drawn in MS Visio, are used for knowledge acquisition. This use both includes the creation of new PVM models and the recreation of models based on the information in the documentation system, when this needs to be changed. Therefore, both manual transfers of information from PVMs to the documentation system and the other way around occur. Further descriptions of the PCS-project at Niro can be found in [2].

After having seen the prototype, Niro expressed that the use of a further developed version of the prototype most likely could produce several benefits compared to their existing solution. The primary benefit would be the possibility of showing model information in PVMs and class diagrams, which would save Niro time, compared to having to elaborate PVMs manually based on the information in their documentation system. Another benefit would be the possibility of easily creating graphical models, which was considered to be an aspect that

could improve the communication with domain experts. On the other hand, Niro requested that the final documentation system would include a range of PDM-related functionalities, such as being able to screen off users with limited modelling prerequisites from some functionalities and information during certain stages of a project. Another important request was that the system should support the use of tables for describing constraints.

The second company to which the prototype was presented was F.L. Smith (FLS). FLS is part of the Danish FLS Industries, and is an engineering and industrial company with an leading market position within the area of development and manufacturing of cement plants. FLS apply a PCS for the creation of budget quotations. FLS document their PCS by using MS Visio, Word, and Access. Compared to Niro, the documentation of the PCS of FLS is only used by the knowledge engineers/system developers. Further descriptions of the PCS-project at FLS can be found in [2, 20].

Having been presented to the prototype, FLS expressed that they might be interested in a similar solution, as this would be likely to enhance the overview of the implemented knowledge and make their PCS easier to further develop compared to the use of their existing methods. Of additional requirements, FLS expressed that they would like to have import/expert functionality so that the documentation system could be interfaced to their PCS, e.g. to allow that class structures and attributes from the documentation system are imported into a PCS, and rules in a PCS are imported into the documentation system.

6 Conclusion

The issue of documentation in PCS-projects is a topic that has been investigated in recent years. This research indicates a need for a documentation system that can provide easily understandable descriptions of what should be or is implemented in a PCS. In this paper the creation and evaluation of a documentation system prototype were described.

A procedure for the development and maintenance of PCSs, which includes three main modelling techniques, has been applied in several cases in Danish industry. However, no software currently exists that supports these techniques in an integrated fashion, which means that manual transfers of information between models are required. Research concerning functional requirements for such a documentation system has been carried out, and recently two papers, which in a detailed manner define the modelling techniques to be included, have been published. Based on these papers, a prototype was created. The aim of the prototype was to evaluate the definitions of the modelling environment, and not PDM-related functionalities. This choice was taken based on the fact that no software includes this kind of modelling environment as opposed to PDM-related functionalities, which are included in numerous software systems. The prototype was created by using C# .Net, and approximately 200 hours were spent on design of software architecture and programming.

The prototype was evaluated by putting a PVM model from an ongoing PCS-project into the prototype and by presenting it to two companies. The experiment of putting a PVM model from an ongoing PCS-project into the prototype showed that the prototype was capable of holding most of the information of this model. However, there were needs for: better possibilities of stating comments in the CRC-cards, support of constraints being formulated in tables, and it being possible to place boxes for comments and constraints next to classes of PVMs and class diagrams. The presentations of the prototype to the two companies indicated that the use of the prototype compared to the use of existing software in many ways could improve the quality of their documentation and save resources in the creation process. On the other hand, more PDM-related functionalities would be required in order to be a real alternative, just as the possibility of import/export to/from a PCS is an important request.

The development of the prototype indicates that the creation of the modelling environment of a documentation system is a task that can be done within reasonable time limits. The evaluation of the prototype indicates that a less complete system might be adequate, as this could still be a far better solution than existing technologies.

The creation and evaluation of the prototype have provided an important basis for the development of a complete documentation system, since many of the created solutions would be reusable, and because of the feedback the prototype has produced. The prototype developed is therefore, an important step in the direction of creating a final documentation system. If this system meets its expectations, this could have a great impact on the way in which the development and maintenance of PCSs are approached and contribute to higher success rates of such projects.

References:

[1] T. Soininen, J. Tiihonen, T. Männistö, and R.

Sulonen, Towards a general ontology of configuration, *AI EDAM-Artificial Intelligence for Engineering Design Analysis and Manufacturing*, Vol.12, No.4, 1998, pp. 357-372.

[2] L. Hvam, A Multi-perspective approach for the design of Product Configuration Systems - An evaluation of industry applications, *Proceedings of PETO Conference*, Lyngby, Denmark, Department of Manufacturing Engineering and Management, Technical University of Denmark, 2004.

[3] L. Hvam, N.H. Mortensen, and J. Riis, *Produktkonfigurering* (Preliminary edition, first edition is to appear in 2006), [Product configuration], Copenhagen, Denmark: Nyt Teknisk Forlag, 2006.

[4] J. Riis, *Fremgangsmåde for opbygning, implementering og vedligeholdelse af produktmodeller - med fokus på konfigureringssystemer*, [Procedure for building, implementing and maintaining product models - with focus on configuration systems], PhD thesis, Department of Manufacturing Engineering and Management, Technical University of Denmark, 2003.

[5] C. Forza & F. Salvador, Managing for variety in the order acquisition and fulfilment process: The contribution of product configuration systems, *International Journal of Production Economics*, Vol.76, No.1, pp. 87-98, 2002.

[6] C. Forza, A. Trentin, and F. Salvador, Product Information Management for Mass Customization: the Case of Kitting, *Proceedings of MCPC2005*, 18-21 Sept. in Hong Kong, 2005.

[7] D. Sabin and R. Weigel, Product Configuration Frameworks - A survey, *IEEE Intelligent Systems & Their Applications*, Vol.13, No.4, pp. 42-49, 1998.

[8] B. Hansen, J. Riis, and L. Hvam, Specification process reengineering: concepts and experiences from Danish industry, *Proceedings of the 10th ISPE international Conference on Concurrent Engineering: Research and Applications*, Madeira, Portugal, July 26-30, 2003.

[9] K. Edwards, and K. Ladeby, Framework for Assessing Configuration Readiness, *Proceedings of the 3rd Interdisciplinary World Congress on Mass Customization and Personalization* (MCPC2005), Hong Kong, Sept. 18-21, 2005.

[10] L. Hvam, *Application of product modelling – seen from a work preparation viewpoint* (Trans.), PhD thesis, Lyngby, Denmark, Department of Industrial Management and Engineering, Technical University of Denmark, 1994.

[11] U. Harlou, *Developing product families based on architectures: Contribution to a theory of product families*, Unpublished dissertation, Lyngby, Denmark, Department of Mechanical Engineering, Technical University of Denmark, 2005.

[12] OMG, *Unified Modeling Language: Superstructure* (Version 2.0: Formal/05-07-04), www.uml.org, 2005.

[13] M. Fowler, *UML Distilled* (3rd edition), Boston, MA, Addison-Wesley, 2005.

[14] K. Beck, and W.A. Cunningham, A laboratory for teaching object-oriented thinking, *SIGPLAN Notices*, Vol.24, No.10, pp. 1-6, 1989.

[15] K. Edwards, L. Hvam, J.L. Pedersen, M. Møldrup, and N. Møller, *Udvikling og implementering af konfigureringssystemer: Økonomi, Teknologi og Organisation*, [Development and implementation of configuration systems: Economy, Technology and Organisation], Final report from research project, Department of Manufacturing Engineering and Management, Technical University of Denmark, 2005.

[16] L. Hvam, S. Pape, K.L. Jensen, K.L., and J. Riis, Development and maintenance of product configuration systems - Requirements for a documentation tool. *International Journal of Industrial Engineering*, Vol.12, No.1, pp. 79-88, 2005.

[17] L. Hvam, and M. Malis, A Knowledge Based Documentation Tool for Configuration Projects, *Proceedings of World Congress on Mass Customization and Personalization*, Hong Kong, Oct. 1-2, 2001.

[18] A. Haug, and L. Hvam, The modelling techniques of a documentation system that supports the development and maintenance of product configuration systems, *Proceedings of IMCM'06*, Hamburg, Germany, June 22-23, 2006.

[19] A. Haug, and L. Hvam, CRC-cards for the development and maintenance of product configuration systems, *Proceedings of IMCM'06*, Hamburg, Germany, June 22-23, 2006.

[20] L. Hvam, Mass Customization for Process Plants, *Proceedings of MCPC2005*, Hong Kong, Sept. 18-21, 2005.