Visual product architecture modelling for structuring data in a PLM system

Bruun, Hans Peter Lomholt; Mortensen, Niels Henrik

Published in:
IFIP AICT - Advances in Information and Communication technology

Link to article, DOI:
10.1007/978-3-642-35758-9_54

Publication date:
2012

Visual product architecture modelling for structuring data in a PLM system

Hans Peter Lomholt Bruun, Niels Henrik Mortensen
Department of Mechanical Engineering,
Technical University of Denmark, Copenhagen

1 ABSTRACT

The goal of this paper is to determine the role of a product architecture model to support communication and to form the basis for developing and maintaining information of product structures in a PLM system. This paper contains descriptions of a modelling tool to represent a product architecture in a company to support the development of a family of products, as well as the reasons leading to the use of the specific model and its terminology. The fundamental idea for using the architecture model is that an improved understanding of the whole product system, will lead to better decision making. Moreover, it is discussed how the sometimes intangible elements and phenomena within an architecture model can be visually modeled in order to form the basis for a data model in a PLM system.

Keywords: Architectural Design Process, Product lifecycle management, Modularisation, Interface modelling.

2 INTRODUCTION

In the last decades many companies have moved design and manufacturing activities to low cost countries in order to sustain productivity growth. Due to the increasing global product development activities, companies experience an extended value chain which is global and often fragmented and thus creates new challenges that companies must overcome [1]. The challenges of overcoming the complexity of a fragmented value chain is supported and made possible by advances in information and communication technology. The manufacturing industry has in decades been an intensive user of data-systems to drive quality and efficiency, adopting information technology, and automation for design, production, and distributing products. It's thus not new that manufacturing companies use IT-systems to manage the product lifecycle including computer aided-design, engineering, manufacturing and product management tools [2]. The increasing amount of data from multiple IT-systems can however often be hard combinable because the large datasets generated by different systems have tended to remain trapped in their respective systems because of their different format and information structure. One of the strategies for handling product data from multiple IT-systems is Product lifecycle management (PLM). The idea behind PLM-systems is that companies can create more value by integrating data from multiple systems which is comparable with the idea of product architecture modelling. PLM can be seen as an integrated, information-driven strategy of managing the whole life cycle of a product starting from generating an idea, concept description, business analyses, product design and solution architecture, technical implementation, and product testing, to the entrance to the market, service, maintenance, and product improvement [3]. One of the challenges in using PLM-systems is the aspect of developing a sound data model representing the product family and its related information. How is the back bone for the data structure model developed, ensuring a suitable alignment of technical domains in a lifecycle perspective? The approach in this study has been to use an architectural product modelling method to capture the information of product structures seen from different technical domains and from different life phases. The paper is structured as follows: Section 2 explains the articles scope and goal for introducing the modelling tool. Section 3 describes different reasons for making representations of product architectures and what they can be used for in terms of developing modular products. Section 4 describes the reasoning for using the modelling method. Section 5 describes significant contributions to the research area of modelling and representing products. Section 6 describes the modelling formalism of the *Interface Diagram* (IFD) and elaborates further on the process for developing and maintaining it. Furthermore the section gives a short introduction to the data model for a PLM system based on the structures developed in the IFD. The section 7 sums up some reflections on the experience with implementing the tool in a company setting, and describes an outline for future research topics.

3 REPRESENTING PRODUCTS USING ARCHITECTURAL MODELS

Product architectures are the mindset behind a product and a business which in brief describes how products and business processes are built up including the rules for designing within product projects [4]. In other words the conceptualization of a product family expressed in a product architecture model assists the understanding of the product system's essence and key properties pertaining to its behavior, composition and evolution [5]. The approach of developing complex products or entire product families can be supported by using product architecture models in which high level descriptions improve multidisciplinary communication and cooperation. Simple products may have little use for architecture descriptions but when developing complex products like mechatronic products or entire product families, it is documented in literature that it is beneficial to use architectural models [6-8]. In general whether the architecture is seen as a characteristic of the product or a description of the product, an architecture has to do with the following three things [9]: *Decomposition* - An architecture is a decomposition of a product into sub-systems (modules); *Arrangement* - An architecture describes the relative arrangement of these sub-systems (modules); *Interfaces* - It describes the relations (interfaces) between these sub-systems (modules) and with the surrounding environment. Architecture descriptions are used by the parties that create, utilize and manage product systems to improve communication and co-operation; enabling them to work in an integrated, coherent fashion. Product architecture representations describe a certain part of the world in a conceptual system model with boundaries, components, internal organisation and behavior. This is done in order to allow more easily intervention with the product the model represents. Whichever format an architecture model has some basic concepts seem to be valid for them. Cooper [10] describes some characteristics for representations: *Displacement* – moving the world into a technology; A*bbreviation* – simplifying the complexity of the reality; and R*emote control* – handle the reality through models in order to intervene. By abbreviation, remote control and displacement it's possible to focus on a particular part of the product program or product itself. Latour [11] emphasises and arguments for the power of using visual two-dimensional inscriptions of the world because they, among others, are mobile, scalable, can be reproduced, recombined, and be superimposed. These are often conditions that are utilised when product architecture representations are deployed in companies. Product architecture descriptions takes place within the context of a project and/or an organisation and is performed throughout the system's life cycle. Consequently a product architecture potentially influences processes throughout the system of products' life cycle. The myriad of information belonging to a product family is however difficult to encapsulate and manage in a single 2D representation, and detailed information belonging to the product architecture are therefore preferable handled in IT systems. IT systems like dedicated PLM systems are consequently seen as tools for modelling product architectures because models that have a computer-readable format, allow fast, precise, and safe data transfer, as well as reducing the effort to replicate and modify information [12].

3.1 PRODUCT ARCHITECTURES AND MODULARISATION

By exemplifying what product architectures represent, one has to address the phenomenon of modularisation. To explain modularity, one must distinguish between modular and integral product structures where the distinction is based on the correlation between functional and structural elements in the product [13]. A modular structure is a structure consisting of self-containing, functional units (modules) with standardised interfaces in accordance with a system definition. A product structure will rarely be classified as either completely modular or completely integral and the structure of a product should thus be classified as being more or less modular or integral. Modular architectures can be perceived as a way of reducing structural complexity [14]. If structural complexity is defined as the number of interactions between components in a system, a system can be perceived as complex when it consists of many closely connected parts with interdependencies. This again makes it difficult to decompose and decouple the systems into structural units. Modularisation is in this context perceived as arranging components in such a way that relations are within units (modules) than among units.

4 WHY ANOTHER TOOL FOR MODELLING PRODUCT ARCHITECTURES?

Two major ways of modelling product architectures seems to exist: *Computer modelling* efforts with an intention to build software computer models, such as those supported by PLM systems or configuration systems and *Phenomenon models* describing the concept of product architectures from a relatively theoretical standpoint. The product architecture

modelling method described in this paper belongs to the group of phenomenon models, with a format that enables its information to transfer to a computer model. The method for modelling product architectures in block diagrams mapping the functional entities of a product and allocation structure to functions is not new. There is nevertheless still a need for modelling architectures that represents different types of information and data elements in a way that address relations to the product family's life phase systems. The optimum of a visual representation provides the necessary details and, yet, still maintains the overview in order to support and improve decision-making. The objective for this paper is to establish a method for supporting this balance between creating overview and operational handling of architectures with the higher level of information they impose.

5 STATE-OF-THE-ART OF EXISTING METHODS AND MODELS

This section describes significant contributions to the research area of modelling and representing products. The chosen methods and models have different forces that will be described. In the end of the section a conclusion will try to compare the reviewed methods in relation to the phenomena supported by the modelling method named *Interface Diagram* abbreviated IFD in the following.

FUNCTION STRUCTURES:

The function-based design methods are characterized by establishing either a function model [15][16] or the schematics of the product [17]. The function structure describes the flow of material, data, and energy through sub-functions of the product using a set of rules (e.g. the rules that are referred to as the functional basis which basically is a common language to describe functional elements. The schematic of the product is somewhat similar to the function model. But where the function model describes the product using functional elements the schematics on the other hand can describe both functional and physical elements, whichever being the most meaningful. The functional structure forms the basis for several different approaches to design or re-design of products. In general, the strength in these methods lies in the use of the functional view point on the product structure, i.e. functional structure.

PRODUCT FAMILY MASTER PLAN

The product family master plan (PFMP) [6] is a visual object oriented modelling approach. The basic idea behind the PFMP is to gather large quantities of information in a visual way on a poster in order to present an overview. The alternative is often to have the information in bills of material and technical drawings in a PDM system. The object oriented approach is somewhat similar to the principles used in the generic BOM as described below, and it makes it possible to gather the information of several product variants in one model without requiring large of redundant data. PFMP introduces three views upon a product family: *Customer view* - the customer view describes the product family as seen from the customer's point of view; E*ngineering view* - the engineering view describes the product family from an engineering point of view; *Part view* - the part view should describe the product family from a physical point of view, i.e. explain how the physical entities (parts and assemblies) are structured and how they vary.

DESIGN STRUCTURE MATRIX:

This approach takes a starting point in the decomposition of a product into components/systems and an identification of interfaces/relations among these [18][19]. Subsystems are mapped against each other for correlation purposes, in order to cluster subsystems that are closely interrelated and separate those that are not. By the use of algorithms, it is possible to encapsulate components into modules or chunks that are closely related to each other from an interaction point of view [20]. This process is referred to as clustering. The Design Structure Matrix is not a decidedly visual model, but it does have the ability to hold a large amount of information for complex systems and make it available for retrieval. The outcome of a DSM can be a proposal for a future modular product architecture.

MODULAR FUNCTION DEPLOYMENT

The modular function deployment (MFD) builds on the methodology of the Quality Function Deployment (QFD) method and on the formulation of eight so-called module drivers [21, 22]. The purpose of MFD is to enable cross functional teams to create a mapping from the physical structure of the products within a family to the functional structure of those products and to ensure that the functional structure corresponds to the demands of the customers. Modular Function Deployment method consists of five consecutive steps. Customer requirements are mapped to

functional criteria and subsystem design characteristics and subsequently forming a physical design in which a modular architecture supports a carefully selected set of modularization incentives called module drivers. Module drivers are formulated as modularisation incentives, i.e. different reasons to modularize a product family. Some drivers are more related than other drivers and each of them will have different implications on the product design. The fourth and fifth steps are rather traditional product development steps in which concepts are evaluated and modules refined.

GENERIC BILL OF MATERIALS

The generic BOM originate from the assemble-to-order environment [23]. The end-products typically have a number of features for which a number of options are available to choose from. Not many options are required in order to make the number of combinations (i.e. end-products) enormous. The number of end-products can easily become too large to be able to define specific BOMs for every single combination. A specific BOM is generated from the generic BOM by replacing the generic items with specific items. When all generic items are replaced by specific items the product is unambiguously defined. The key to efficient generation of specific BOMs is to link the generic BOM to a set of parameters and parameter values (customer options) in such a way that when a full set of parameter values are obtained. The generic BOM is a concept that is introduced to enable creation of a specific manufacturing BOM when the customer places an order. The generic BOM is used to describe related products in one all-embracing model by using generic and specific items.

DECISION TREE

The decision tree [24] is used by [25] as a product configuration model, which basically represents all the valid combinations of the components that can be used to obtain the desired functions for the customer. The decision tree presents the multitude of component variety within a product family and by the use of positive combinatory relationships and/or incompatibility relations it defines the possible product configurations. The model has some visual strengths, the applicability of the visual representation is however questionable if the number of product groups, modules, module variety, and variants is high because the possible configurations model can be extreme.

5.1 CONCLUSION ON STATE-OF-THE-ART LITERATURE

The review of existing methods and models reveals a multitude of different visualization approaches and models for assessment of product designs and flow in operations in relation to product families and single products. All of the methods can play a role in identifying structural aspects of an architecture and some of them also takes functional aspects into consideration. The modelling formalism of the IFD is based on the same mindset as the functional structure models and is further developed in order to support modelling of product variety and to form data structures handled in PLM systems. The PFMP and the generic BOM are both product models and they only map the variety in the product domain. There is no direct link to the production and supply chain. Design structure metrics is excellent for handling relations between entities in a product system, but the method is not very visual and it makes it hard to intervene with the represented product or product families. The primary strength of the MFD tool is the tool's ability to link potential effects in other life phase system to the product structure, i.e. linking the module drivers to the technical solutions. The generic bill of materials and the decision tree are aimed for generating specific BOMs at customer order entry. They are also powerful methods to describe the variety within a product family. Both the generic BOM and the decision tree have their forces as product configuration methods more than actual architecture descriptions. **Figure 1** presents an overview of how well the evaluated methods and models relates to the phenomenon addressed in the IFD. The comparison uses a five step scale going from *"not addressed at all"* to *"fully addressed"*. No single method or model addresses all phenomena handled in the IFD. The methods can generally be divided into two groups: (1) Methods that are relatively visual but only focus on aspects related to a single or few requirements and (2) methods that have a broad focus and touch many aspects but lack visual qualities.

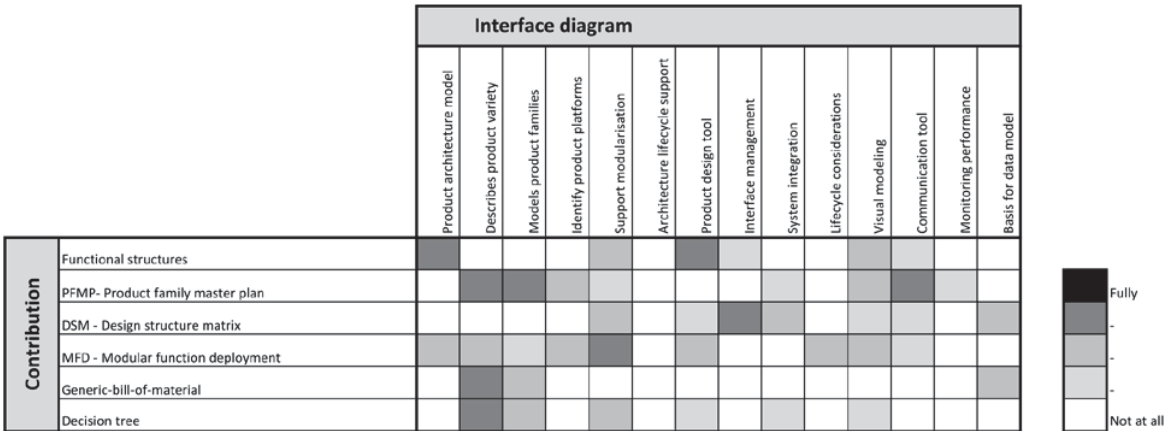

Figure 1 Overview of how well the presented state-of-the-art contributions relates to the phenomonon adreesed in the IFD

6 PRODUCT ARCHITECTURE MODEL - THE INTERFACE DIAGRAM

The Interface Diagram (IFD) are capturing a certain structural characteristic of a system, mostly combining an aspect of mapping between domains, and mainly a mapping between function and form. The system level is for the model regarded as the family of products that can be developed on the basis of the architecture. Systems are of course recursive meaning that systems can be within systems etc., which also is addressed by the IFD. The following sections describes the modelling formalism, the process for creating and maintaining the modeled architecture, and elaborates on the description of product architectures as conceptual systems models in order to allow intervention. The architecture model puts emphasise on managing technical interfaces showing the relations from a system and a module viewpoint, hence the chosen name of the modelling tool.

6.1 MODELLING FORMALISM

This section introduces the modelling formalism. The formalism is denoted *Interface diagram* which has its basis in the *Generic organ diagram* presented in the work of Ulf Harlou [6]. There are several objectives for creating an IFD. Among the most important are: Acting as a vehicle for communication between stakeholders; providing guidance for the decisions of detailed design; to support a modularization process; keep track of latest design changes; complete overview of the product; to ensure simple interfaces; dealing with transformation of the product over time; point out responsibility of interfaces, and to form basis for a PLM data structure. Because of secrecy issues for the company involved, the formalism is illustrated and described by using an example of an IFD conducted for a product family of *Bobcats.* The Bobcat is a mobile power loader that is a small, self-propelled utility vehicles used for a large variety of purposes such as heavy duty agricultural earth moving, construction site utility and integrity support of public and private infrastructure. The IFD is modeled by means of blocks and lines in the standard software program Microsoft Visio, that has some basic functionality that is suited for structuring and visualizing different perspectives of architectures. The interface diagram is normally printed on large blue prints in order to get the overview of the architecture it represents. **Figure 2** shows a screen shot of the IFD for the Bobcat product family with explanations of the overall layout and content. In order for the reader to quickly understand the structure of the product it is suitable to model the diagram so the layout is established as a cross section of the actual product. In that way the physical layout of the product is possible to recognize in the diagram.

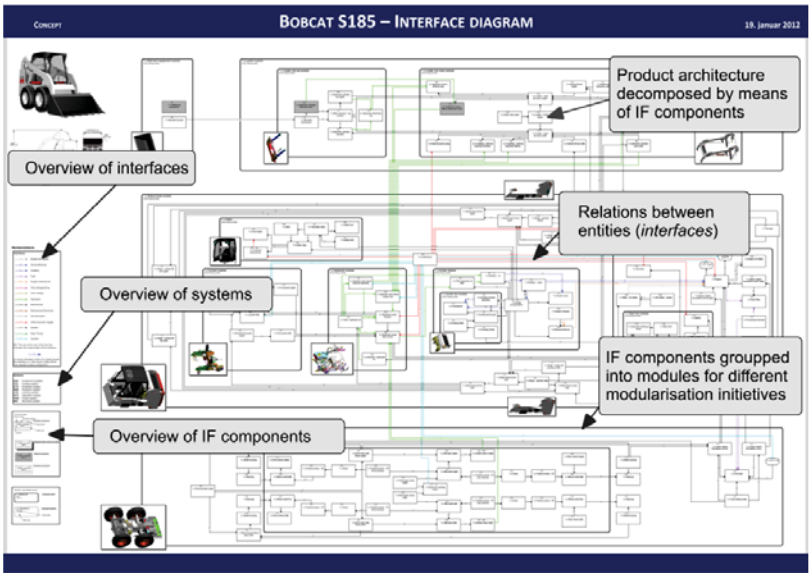

Figure 2 Representation of a Bobcat's Interface diagram

The diagram can be read following the interfaces. For products that process or transforms objects this gives a logical reading direction, for other products it is up to the reader to find a suitable flow in the model. **Figure 4** is a symbolic representation of an Interface diagram. The diagram follows the approach of modelling architectures by use of block diagrams. The main elements of the diagram formalism are functional objects denoted Interface components (*IF component*). The purpose of the IF components is to decompose the systems and modules into smaller functional building block. IF components can have different characteristics and are thus modeled in different ways. An existing IF component represents a generic assembly or part in the architecture and is symbolized by a white block. An optional assembly or part is symbolized by a grey block. A future assembly or part which has to be taken into consideration in the architecture but has not been developed yet has a dotted line around a white block. Finally variety of assemblies or parts is modeled by placing blocks on top of each other with a little offset. The representation shows that the specific IF component has at least one variant. Each block has a designation for system relationship. A primary system means that the system designs the underlying parts and holds the responsibility for the functionality, while a secondary system has requirements to the design. An example could be that a cooling unit is designed by the team responsible for cooling and conditioning, while the system responsible for the engine has requirements to the cooling performance they need for their engine design.

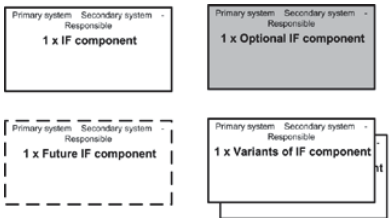

Figure 3 Four representations of an IF component

Each IF component belongs to a product *system*. The systems deal with the product's main functions or its related lifecycle. To avoid confusion, systems and their interaction must be clearly defined. This is done by choosing the relevant interaction as a basis for determining the system boundary. There is no fixed list of systems to be included in the development. Systems can be modeled and thereby control important properties of the final product. Modules are modeled by arranging IF components inside boxes with a thick black boundary and rounded corners, see **Figure 4.** All modules are assemblies of physically joined components forming one bill of material (with possible multiple levels). Modules can contain (smaller) modules, but they do not overlap. As it is clearly defined to which modules any element

in a product belongs. Modules are suitable for splitting responsibility in the product development process. The structure of the interface diagram appears as the relations are added to the diagram.

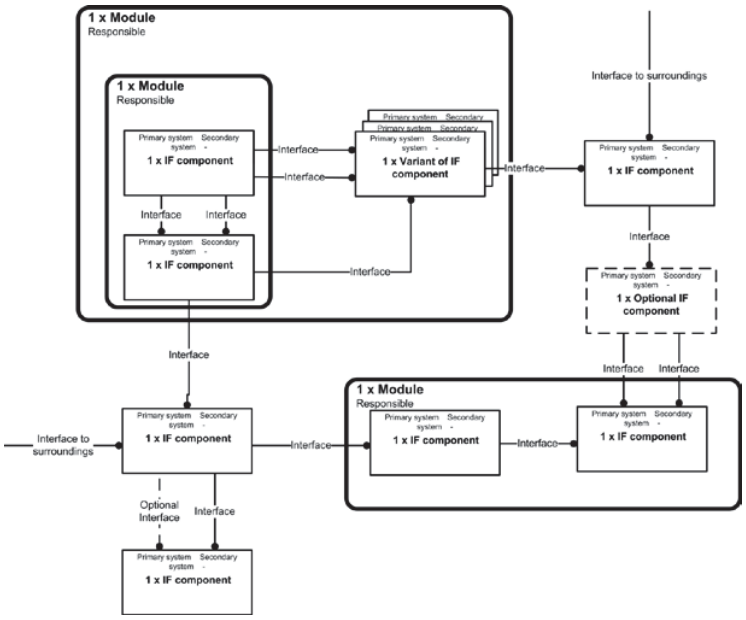

Figure 4 symbolic representation of a generic interface diagram

An interface among two IF components represent a physical relation, e.g. physical connection, energy transportation, information flow or flow of material. The purpose of working with interfaces is to ensure responsibility for the components interaction and to ensure that components are interchangeable, when relevant. In a product modelling context, the interface has to belong to a common structure, or some sort of generic placeholder, in order for the interfaces to be inherited to the involved elements. An interface is thus a relation between two or more IF components and in every set of IF components it is defined which is the master and the slave. This enables a responsible for an IF component to monitor whether he owns the right to change or modify the related interface. From a systems perspective it is sometimes problematic where to assign the interface and how to perceive it, mainly because an interface involves several elements – and is the interface then a feature on a part or a property of the relation between parts? From a description or modelling point of view, the interface is more of a design specification that the interacting elements have to accommodate. When transiting the interface structure into a PLM system, interfaces can be perceived as feature elements of a part in a system. In a modularisation context all interfaces between different modules and the outer environment should be carefully specified in order to complete the module. The interfaces between the IF components are drawn with lines which represents a relation. There exist a number of interface classes and the list can be extended in order to support the context in which a product belongs. Examples of interface classes are: Mechanical, spatial, cooling air, cooling liquid, electrical measurement, electrical signal, grid voltage, hydraulic, mechanical-electrical, oil lubrication, optical signal et. al. The dot at the end of the line indicates the responsible of the interface. Optional interfaces can be modeled on the diagram to show affected relations between entities or systems if the interface is established. Interfaces are a part of many product architecture definitions, and in the classic perception of modularisation it is the interface that ensures decoupling, and thereby enables reuse, sharing, substitution and serviceability. In principle a line is drawn for each interface in the architecture, but for relations like electronic products that would mean a myriad of lines. In such cases the interface diagram is simplified by only drawing one line for each type of interface between the individual IF components.

6.2 PROCESS FOR UPDATING INTERFACE DIAGRAM

The interface diagram is a dynamic tool which is updated and refined during the project life cycle. The technique for modelling the architecture is based on interviewing the persons with the insight to model systems in their totality. It is seldom that a single person in a company has the needed insight to draw the diagram. Consequently several domain experts have to be involved in giving input to the diagram. The process of establishing and maintaining an IFD for a product family is thus an iterative process consisting of the following activities:

- Interviews with each system responsible domain expert (the technical stakeholders) in a product
- Draw the design in a block diagram (MS Visio)
- Reviews with domain experts in order to approve the present status
- Load the structures in to the PLM system
- Enrich the data structure with information like CAD, requirements, cost etc.

6.3 BRIDGING FROM SYSTEM TO MODULAR ENGINEERING

A major strength of the IFD is its ability to handle the development of a product or product family in different lifecycle perspectives forming different structures. For highly complex products with many systems, groups of specialized engineers develop entire systems. Systems are however seldom the most appropriate way of manufacturing the product and combining system after system. Modularisation in respect to aspects of assembly, transportation, sourcing, serviceability, changeability and other drivers for modularisation are desirable to handle in the architecture model. An example from the Bobcat is used to illustrate this. **Figure 5** shows the architecture of two of the systems in the BobCat; the hydraulic system and drive train system. The two systems are physically allocated to different sections of the Bobcat connected by interfaces as hydraulic hoses, electrical wires, transmissions belts etc. Because of initiatives for modularisation it is decided to manufacture the Bobcat in modules. Modules can consist of elements belonging to different systems i.e. developed by different system teams. It is therefore crucial both to integrate systems in modules, but also to handle interfaces created along the module boundaries splitting systems. **Figure 6** shows the architecture in which the two systems are split out in three modules, Baseframe module, Engine module, and Hydraulic module. When monitoring the IFD in Visio it is possible to turn on layers containing specific systems and the interfaces belonging to it, and also monitoring interfaces to other systems. Moreover it is possible to see entities belonging to modules with the same respect to interfaces. Finally it is possible to monitor specific modules and turning on selected systems. The figure below illustrates this way of using the IFD to hold information from multiple product structures. The functionality of using the structures from the IFD in a PLM system makes it furthermore possible to link actual data from e.g. CAD to the structures.

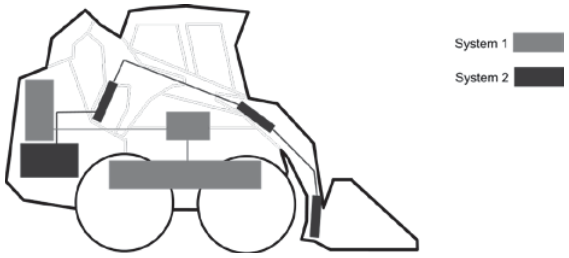

Figure 5 System structure of a bobcat showing entities beloning to systems and the relations between them

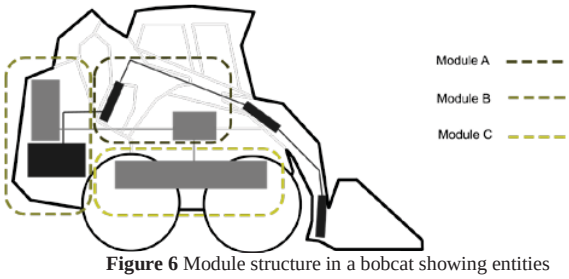

Figure 6 Module structure in a bobcat showing entities beloning to modules and relations between systems and modules

The IFD has in that way proven its worth for creating parallelism of development activities as well as distributing responsibilities for them. Moreover the model supports detailed development of systems and especially the integration of systems. The functionality of visualizing the product family's system and module structure in one superimposed view is recognized as a way of running development activities in a more parallel. By transiting the structures into the company's PLM system it is moreover possible to enrich the structures with information like CAD, ECAD, requirements, interface control documents, cost data, weight data etc.

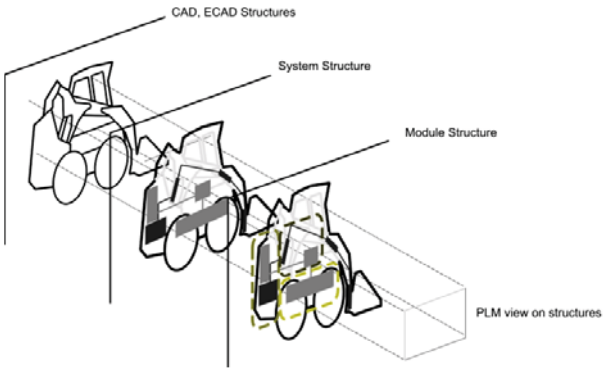

Figure 7 The IFD handles multiple product structures

The PLM system enables management of processes such as change/configuration management. But one of the most important functionalities is that the PLM system enables all product stakeholders to access a single, trusted, central data repository, in which it is possible to manage all forms of digital product development data – including mechanical, electrical and software.

6.4 THE INTERFACE DIAGRAM AS A BASIS FOR PRODUCT STRUCTURE DATA MODEL

As described in the previous section the IFD focuses on management of design data which are not possible to model in a 2D representation. The IFD is both addressing the product structuring as a integral part of the design process and as carrier of design data, capturing it and enabling management of it in a PLM system. Describing the actual functionality in the used PLM system are outside the scope of this paper, but an overview of the data model is provided for the reader in order to understand the structure loaded in to the system. **Figure 8** illustrates the entity relationship model (ER) modeled in UML. ER models are high-level conceptual models for database design and the purpose of showing it here is to get an overview of the data structure created in the IFD. Database designers often use this methodology to gather requirements and define the architecture of the database systems. The output of this methodology is a list of entity types, relationship types, and constraints. The product is the top entity. Assemblies and systems can be part of the products. IF components are a part of assemblies and a specialization of an assembly is a module. Systems are associated with IF components and IF components are associated with interfaces. CAD assemblies and components are parts of IF components. CAD parts can be a part of components but are always a part of CAD assemblies.

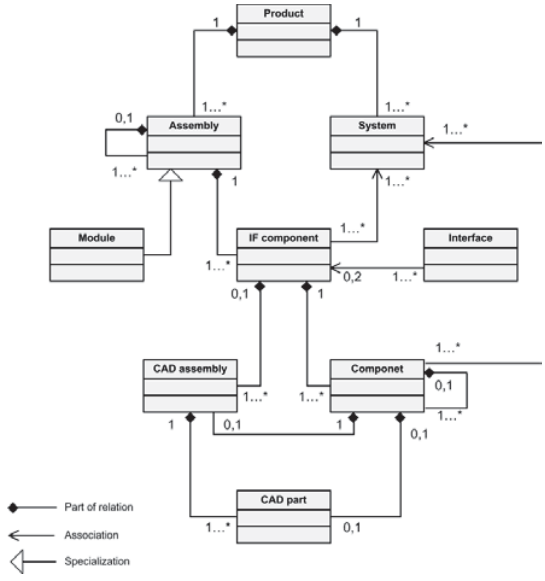

Figure 8 Enity relation model of the structures loaded into a PLM system

7 CONCLUSIONS AND FUTURE WORK

This paper presents a visual modelling tool for representing product architectures of mechatronic product families. Efforts have been made on modelling a product architecture that represents a product family's system and modular structures. The fundamental idea for using the architecture model is that an improved understanding of the whole product system, will lead to better decision making. Moreover, it is described how the sometimes intangible product structures within an architecture can be handled in a PLM system, based on the assumption that knowledge about a product's architecture has to be tangibly instantiated, in order for people and decision makers to successfully share it and use it. The model deals with transformation of the product over time and it offers a fundamental approach for proactive modular architecture development. The experience by using the model was that it acts as a vehicle for communication between stakeholders. It created a holistic overview of the product family, and it ensured system and module integration because of focus on simple interfaces. The product structures established in the architecture could be loaded directly into a PLM system which enabled to manage product information on a system, module and interface level in a lifecycle perspective. The most important reason for handling the architecture model in a PLM system is that it enables the possibility of managing the large amount of design information belonging to a mechatronic product family in a lifecycle perspective. An area of focus in future work is the format of interface descriptions linked to the product structures. In order to work professionally with structures in a PLM system, interfaces has to be documented in a way that documents more than responsibility and properties of the relations. Moreover emphasise in the ongoing research is on developing the IFD formalism to support the evaluation of the "goodness" of the modeled architecture in respect to market and production aspects.

8 REFERENCES

[1] Manyika, J., et. al., 2011, "Big data: The next frontier for innovation, competition, and productivity," McKinsey Global Institute, .

[2] Stark, J., 2007, "Global product: Strategy, product lifecycle management and the billion customer question," Springer Verlag, .

[3] Sääksvuori, A., and Immonen, A., 2008, "Product lifecycle management," Springer Verlag, .

[4] Meyer, M.H., and Lehnerd, A.P., 1997, "The ¤power of product platforms building value and cost leadership," The Free Press, New York, pp. 267.

[5] IEEE, 2011, "
ISO/IEC 42010 *Systems and Software Engineering — Architecture Description*," .

[6] Harlou, U., 2006, "Developing product families based on architectures contribution to a theory of product families," Department of Mechanical Engineering, Technical University of Denmark, Lyngby, pp. 173 sider.

[7] Martin, M. V. V., 2002, "Design for Variety: Developing Standardized and Modularized Product Platform Architectures," Research in Engineering Design, 13(4) pp. 213-235.

[8] Ulrich, K., 1995, "The Role of Product Architecture in the Manufacturing Firm," Research Policy, 24(3) pp. 419-440.

[9] Kvist, M., 2009, "Product Family Assessment," .

[10] Cooper, R., 1992, "Formal Organization as Representation: Remote Control, Displacement and Abbreviation," Rethinking Organization: New Directions in Organization Theory and Analysis.London: Sage, pp. 254–72.

[11] Latour, B., 1986, "Visualization and Cognition," Knowledge and Society, 6pp. 1-40.

[12] Bergsjö, D., Catic, A., and Malmqvist, J., 2008, "Towards Integrated Modelling of Product Lifecycle Management Information and Processes," Proceedings of NordDesign 2008, pp. 253-264.

[13] Ulrich, K., and Tung, K., 1991, "Fundamentals of Product Modularity," American Society of Mechanical Engineers, Design Engineering Division (Publication) DE, 39pp. 73-79.

[14] Miller, T.D., 2001, "Modular engineering ; An approach to structuring business with coherent, modular architectures of artifacts, activities, and knowledge," Technical University of Denmark, Department of,Mechanical Engineering; DTU, Lyngby, .

[15] Pahl, G., Beitz, W., and Wallace, K., 1996, "Engineering design A systematic approach," Springer, London, pp. 544.

[16] Otto, K., and Wood, K., 2001, "Techniques in Reverse Engineering, Systematic Design, and New Product Development," .

[17] Ulrich, K.T., and Eppinger, S.D., 2004, "Product design and development," Irwin McGraw-Hill, Boston, Mass., pp. 366.

[18] Pimmler, T. U., and Eppinger, S. D., 1994, "Integration Analysis of Product Decompositions," American Society of Mechanical Engineers, Design Engineering Division (Publication) DE, 68pp. 343-351.

[19] Hölttä-Otto, K., and de Weck, O., 2007, "Degree of Modularity in Engineering Systems and Products with Technical and Business Constraints," Concurrent Engineering, 15(2) pp. 113-125.

[20] Steward, D. V., 1981, "The Design Structure System: A Method for Managing the Design of Complex Systems," IEEE Transactions on Engineering Management, (3) pp. 71-74.

[21] Erixon, G., 1998, MFD–Modular Function Deployment, A Systematic Method and Procedure for Company Supportive Product Modularisation, .

[22] Ericsson, A., and Erixon, G., 1999, "Controlling design variants modular product platforms," Society of Manufacturing Engineers, Dearborn, Michigan, pp. 145.

[23] Van Veen, E., and Wortmann, J., 1987, "Generic Bills of Material in Assemble-to-Order Manufacturing," International Journal of Production Research, 25(11) pp. 1645-1658.

[24] Rea, R. C., 1965, ""Helping a Client make Up His Mind - the use of a "Decision Tree" Helps a Client in Planning His Estate"," pp. 39-45.

[25] Tiihonen, J., and Soininen, T., 1997, "Product Configurators: Information System Support for Configurable Products," Helsinki University of Technology, .