INFORMATION RICH MAPPING REQUIREMENT TO PRODUCT ARCHITECTURE THROUGH FUNCTIONAL SYSTEM DEPLOYMENT: THE MULTI ENTITY DOMAIN APPROACH

Hauksdóttir, Dagný; Mortensen, Niels Henrik
Technical University of Denmark, Denmark

Abstract

Successful transformation of design information from customer requirements to design implementation is critical for engineering design. As systems become complex the tracking of how customer requirements are implement becomes difficult. Existing approaches suggest so called domain modelling for mapping requirements to architecture. These approaches do not fully support the steps and information created during product design synthesis. Design Specifications used to guide the design are often documented in text based documents, outside the design models. This results in lack of traceability which may impede the ability to evolve, maintain or reuse systems. In this paper the Multi Entity Domain Approach (MEDA) is presented. The approach combines different design information within the domain views, incorporates both Software and Hardware design and supports iterative requirements definition. The results suggest that it is possible to present design information in structural domain views, presenting more elaborate information of the design synthesis than provided by previous approaches. However, further validation in a practical project setting is required to validate the approach.

Keywords: Design specifications, Domain engineering, Systems Engineering (SE), Requirements, Knowledge management

Contact:
Dr. Dagný Hauksdóttir
Technical University of Denmark
Department of Management Engineering
Denmark
dagnyhauks1@gmail.com

Please cite this paper as:
Surnames, Initials: *Title of paper*. In: Proceedings of the 21st International Conference on Engineering Design (ICED17), Vol. 6: Design Information and Knowledge, Vancouver, Canada, 21.-25.08.2017.

1 INTRODUCTION

In engineering, compelling arguments justify why an early understanding of stakeholder' requirements lead to systems that better satisfy their expectations (Nuseibeh, 2001). It is generally considered as good practice to capture solution neutral requirements, reflecting the "problem domain" separately allowing engineers to devise the best solution without being limited by premature design constraints. Typically the effectiveness of a solution is determined with respect to a defined problem, however, the nature of the problem and its scope could depend on what solutions already exist or what solutions are plausible and cost-effective (Chen et al., 2013). Existing systems, infrastructure and capabilities provide a rich base from which to create new capabilities but also introduce a set of complex constraints. Architecture at one level must support requirements at that level, but since generates constraint to the solution space; it drives requirements at lower levels (Cole, 2006). There is a common misconception that requirements engineering is just a single phase carried out and completed at the outset of product development. Instead requirements and design should rather be treated as interactive activities, handled simultaneously throughout the development life-cycle (Hull et al, 2011).

One approach to improve engineering design performance is through reusing previous knowledge. Ideally, identifying and reusing front-end knowledge such as market information and Customer Requirements (CRs) should be the optimal origin of information reuse, leading to reuse in subsequent process steps. This requires a successful transformation of design information. The mapping between requirements and architecture has e.g. been addressed by Quality Function Deployment, Axiomatic Design and Product Family Engineering methods. In these approaches requirements are mapped to functional systems or physical design parameters, using matrixes or hyperlinks, resulting from a given solution. These approaches might be sufficient for analysing the product design or for configuring products, but in engineering design it is important to capture in more detail the information synthesis as moving between the product design domains.

The solutions concepts explaining how components of the systems work together to achieve an end result and the rationales for a particular design are usually described in so called Design Specifications (DSs) used to guide the design. The DSs are often documented in text based documents, outside the design models. In this context, an explicit link for how CRs are address in the DS is missing and no mapping method is clearly available (Sun et al., 2009). The loss of this understanding may impede the ability to evolve, maintain or reuse systems. Information models capturing the design synthesis between requirements and physical design realization in a more elaborative manner could therefore provide improved documentation support for engineering design.

The purpose of this paper is to present a conceptual modelling method supporting the mapping from requirements to design realization capturing the information needed to sufficiently support design synthesis. Such a modelling method will need to integrate different types of design information in a more iterative manner than currently available and to support requirements management on different design abstraction levels. The objective is further that the method can manage information for electronic products and thus integrate hardware and software design information. Hutcheson et al. (2007) present a method for Functional based System Engineering (FuSE) describing stepwise activities to identify and synthesize system solutions. The method presented in this paper, the *Multi Entity Domain Approach (MEDA),* has been inspired by the functional system deployment described by FuSE and the intention is that the modelling method can capture the information needed to reflect the activities included.

The paper is organized as follows: In Section 3 an overview of the current research is summarized. In Section 2 the research design is presented. In Section 4 the FuSE method is presented. Section 5 provides a conceptual basis for the paper, followed by a description of the suggested method; MEDA, in Section 6. A discussion of the method is provided in Section 7 and the paper is concluded in Section 8.

2 STATE OF THE ART

In this section, a review of well-established approaches mapping requirement to design and how they address requirements in the solution domain is summarized.

2.1 Requirements Engineering Traceability

Hull et al. (2011) suggest four abstraction levels of requirements; stakeholder-, system-, sub-system- and component requirements. The highest levels of system description should be firmly rooted in the problem domain, whereas subsequent layers, starting with system requirements, operate in the solution domain (Hull et al., 2011). In the requirements engineering context, tracing is about understanding how high level requirements are transformed into low-level requirements (Hull et al., 2011).

Elementary traceability is where requirements are simply broken down to more detailed requirements, e.g. user requirements are broken down to system requirements. The traceability relationships are usually many-to-many and therefore, the simplest way is to implement layers of requirements with traces between them. However, this way the rationale for why a lower-level requirement meets a higher-level requirement is not documented (Hull et al., 2011).

A variety of information can be captured in between levels of requirements to provide a more knowledge rich traceability Hull et al. (2011). This helps in assessing the logic, validity and completeness of the requirements breakdown. Visual models can furthermore provide important background information for requirements and design realization.

These layers of design information are like the "filling" between the layers of requirements and can be gathered into a DS document. In this context the DS summarizes – textually and visually – why one layer of requirements is sufficient and necessary to satisfy the layer above. However, the approach does not provide traceability to requirements or a structured layout of the design information is not provided.

2.2 Matrix Based Design Transformation

Quality Function Deployment (QFD) originally developed by Dr. Yoji Akao, in 1966 (Akaor, 2004) divides the product development process into four steps using four correlation matrices. The first phase transforms the CRs for the product (the WHATs) into technical measures (technical requirements, product design specifications, engineering characteristics, etc.) (the HOWs). This phase, called product planning, is given a special attention in the House of Quality (HOQ) approach. The second phase, called Part Deployment transforms the prioritized technical measures into part characteristics. The four phases in QFD are demonstrated in Figure 1.

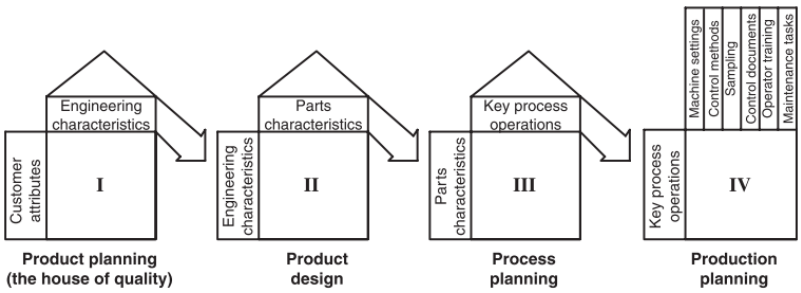

Figure 1. Quality function deployment overview

The mapping between the information domains is done by a correlation matrix, one matrix between each two domains. In the model, the design elements are not provided by a comprehensive description, as would be required for a DSs. Furthermore, the flowdown from requirements to design is one directional that is requirements are not synthesized on the basis of a given design. It can in general be assumed that the application of QFD is rather to analyze the quality and completeness of a given design, than to drive or document design implementation or to generate DSs.

Axiomatic design (AD) developed by N.P. Suh in the late 1970s to provide a systematic and scientific basic for making design decisions (Gonçalves-Coelho, 2005). The system is represented by four domains; customer-, functional-, physical- and process domains. The customer and functional domains describe the "WHATs" while the physical and process domains describe the "HOWs".

Mapping between the functional and physical domain has been described as the primary step in identifying a good solution and is where the transformation is made from the problem domain to the solution domain. The decomposition of these vectors cannot be done by remaining in a single domain, but can only be done through zigzagging between the domains to decompose the design problem (Suh, 1998). At a given level of design hierarchy, there exists a set of FRs defined as the minimum set of requirements needed at that level. Before zigzagging to the next level of FRs, the corresponding

hierarchical level DPs shall be selected. The design team will develop different solutions and optimize the best alternative at each level (Tang et al., 2009). The relationship between FR and DP vectors can be presented in a *design equation* where the design matrix (DM) characterizes the product design. The zigzagging and a DM for one level of decomposition are shown on Figure 2.

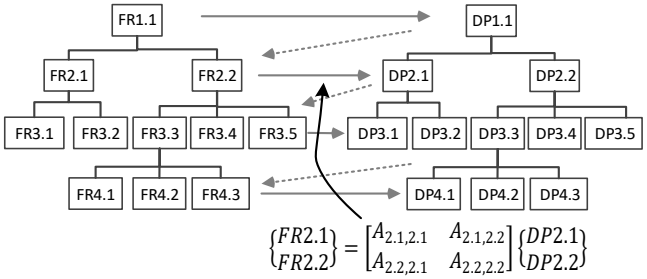

Figure 2. Zigzagging between the Functional and Physical domain

The DM construction is guided by Solution Principles (SPs). The designer considers through what SPs a set of FRs can be realized and then conceives the corresponding structure candidates in the DM (Tang et al., 2009) as shown in Figure 3.

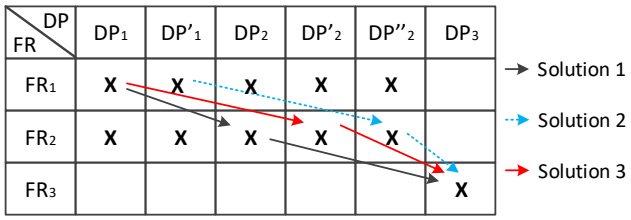

Figure 3. Three solutions presented in the DM (Tang et al., 2009)

Similarly, as in QFD design matrixes are applied for the mapping between the domains. However, instead of mapping all items from one domain to the next, in a single DM as in QFD, in AD the mapping is done for each level of abstraction (Suh, 1998) unveiling the interplay that exists between the design elements (Gonçalves-Coelho, 2005). However, the AD approach does not incorporate a description of the identified SPs or other design information, as required to construct a DS. Furthermore, there is not a "backward" matrix, mapping the DPs to FRs. It is therefore not explicitly demonstrated which FRs result from which DPs.

2.3 Product Family Engineering

Product Family Engineering (PFE) (or Product Line Engineering) is an engineering practice that focuses on developing a stream of products by creating an underlying architecture from which new product variants can be generated by applying shared designs in a new context. A significant body of research has been presented for domain engineering and modelling of a reusable reference architecture. In Software Product Line Engineering four domain views have been suggested; Requirements Engineering, Design, Realization and Test (Pohl et al., 2005). For Hardware PFE a well-established approach called the Product Variant Master (PVM) suggests three domain views, Customers-, Engineering- and Parts views (Hvam et al., 2008).

The different domain views are modelled independently, each presenting one type of information, and trace links are then established between the domains. In both approaches the requirements are only addressed in one domain; the requirements engineering or customer view. The requirements are furthermore focused on identifying the variance in the product family from a customer point of view rather than identifying a completed set of requirements.

The application of PFE is to mass customize products. This means that most of the solutions have been implemented in the product architecture, prior to the product configuration and a product variant can be generated based on a set of customer attributes. In product development, the objective is however to identify optimal solutions to incorporate CRs is a new product. Therefore product development requires including more detailed design knowledge to negotiate the optimal product design. It could be a valuable addition to more information rich domain models to better facilitate engineering design by domain

modelling, or one could say to extend the PFE modeling concepts to better support traditional engineering design.

3 RESEARCH DESIGN

The objective for the research is to identify an improved method to document DSs that could support platform based product development by systematically incorporating additional design knowledge in structural domain model views. Another object was to find a way to explicitly specify and manage solution dependent requirements to drive testing of sub-systems and modules. This implies a set of characteristics that the methodology should possess. The research questions are defined as follows:
RQ1: Can the DSs capturing the information required for design synthesis be established in a structural domain view?
RQ2: For embedded products, how can both hardware and software related design information be incorporated in structural domains?
The presented technique was validated using an example from industry to demonstrate the suggested method and assess its utility.

4 FUNCTIONAL BASED SYSTEMS ENGINEERING

Function-based Systems Engineering (FuSE) is a design method that uses functional modeling throughout the first three phases of engineering design: product planning, conceptual design and embodiment design. A functional model is defined as a graphical model of the transformations of energies, materials and signals that occur through use of the product (Hutcheson et al., 2007). The process for FuSE defined by Hutcheson et al. (2007) is as follows.
During the Product Planning phase of design, FuSE includes five activities.

1. *Black box functional modeling:* The first step is to begin modeling a black box functional model representing the overall, desired, functionality of the product. It is created by mapping customer needs to overall functionality and product-level flows.
2. *Identification of Product-level requirements:* To accomplish this step, the input and output flows are listed and used as a reference for placing requirements.
3. *Conceptual Functional Model (CFM) development:* A CFM, including minimal information about form-specific solutions, is made to identify the basic functions required ensuring that the potential solution space of conceptual design is kept open.
4. *System boundary identification:* Based on the CFM boundaries for functional systems can be identified. Figure 4 shows the identified functions and the corresponding systems for a power train.

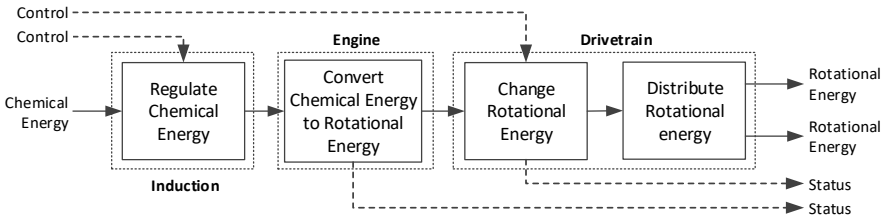

Figure 4. Powertrain System Boundaries (Hutcheson et al., 2007)

1. *Requirements flowdown from product level to the system level through the use of conceptual behavioral models:* Mathematical models are established for the individual functions in the CFM and assembled into a solvable model. This model is then used to determine the states of internal flows in the system. The objective is to establish a relationship between the inputs and outputs, for each function, with minimal assumptions regarding the form of the system. The result is a series of equations that allow the internal states of the system to be solved.

The system-level requirements are identified in a similar manner to the product-level requirements, but using the internal flows of the CFM.
During the conceptual design of a system, FuSE is applied to four activities.

2. *Identifying potential system solutions:* Involves finding components or collections of components that solve the desired functionality for the systems identified in the CFM.

3. *Developing behavioral models for these solutions:* Behavioral models allow the performance of concepts to be evaluated and compared to the product and system-level requirements.
4. *Updating requirements and functionality:* The functional model should be updated to reflect changes in functionality that have occurred as a result of the identified solutions. At this point, the functional model begins to become a Form-Specific Functional Model (FSFM). The FSFM is used to identify additional requirements specific to the particular solution. It is then used to further refine the design for the remainder of the design process.
5. *Iterating this process until feasible solutions have been found:* If multiple feasible solutions are desired for a product, the previous steps should be repeated until a sufficient set of solutions has been identified.

FuSE then follows a similar process (Activities 10-13) for identifying sub-systems and auxiliary functions during the embodiment design phase.
When the activities are concluded, feasible solutions should be identified for all functionality. The remainder of the design process, the detailed design phase, is then performed.
FuSE misses an identification of how the elaborated information can be documented in a structured way. This is the target of the method presented in this paper.

5 CONCEPTUAL BASIS

This section introduces the conceptual basis used in this paper:

- *Requirement:* A thing (characteristic or function) that is needed or wanted.
- *Function:* A transformation of input flows into desired output flows.
- *System:* A abstract unit of intelligent activity that provides a specific function or collection of functions.
- *Solution:* A working physical structure that solves a specific function or collection of functions.
- *Module:* A component of a collection of components that are grouped by their physical interfaces.
- *Interface:* A surface that forms a common boundary or a point of interaction between two components or systems. Here 4 types of interfaces are included:

– *System interface:* Interactions between two systems, in form of energy, material or information flow.
– *Technical interface:* Interactions between technological disciplines as a part of a specific solution.
– *Hardware interface:* A plane forming the common boundary between two parts of matter or space.
– *Software interface:* The layout or design of the interactive elements of a computer program.

In the context of the presented approach it is important to understand the different meaning of functional requirements (FRs) and a function A FRs present a functionality or an effect that the product provides to its external environment, where the product is presented as a “black box” while a function describes the transformation of input and output flows inside the product, therefore describing a “white box view” of the product in an abstract way (Negal, at al., 2011).

6 THE MULTI ENTITY DOMAIN APPROACH

In this paper the *Multi Entity Domain Approach (MEDA)* approach is presented. The main identifier of the approach compared to other structured design approaches, is that it suggests describing the product in 3 separated domain views while at the same time combining more than one information types in the domains. This enables expressing closely linked information worked on in the design process as the information are defined. It should therefore establish a logical route from requirements origin, to the design implementation.
This is accomplished by using a functional system and design concepts as a link between requirements and design. The design concepts can include both HW and SW based solutions which enables an identification of which functions will be implemented using which technology. Traceability between design elements is captured through a tree structure and hyperlinks between the views. Figure 6 provides an overview of the domains included in MEDA.

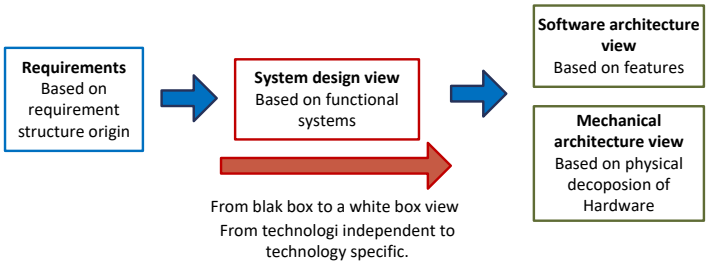

Figure 5. Multi Entity Domain Approach overview

6.1 The Requirement Domain

The first step of the development process is to identify the requirements to the intended product as described in activities 1 and 2 in the FuSE approach. The requirements primarily origin from the customer needs, the operational environment of the product as well as from the standards and legislations that apply to it. To capture the requirements from their origin in a structured way, the structure presented by Hauksdóttir et al. (2012) organized based on the requirement origin, is suggested. The Requirements Specification (RS) contains, properties and constraints, combined in the same structure.
When a RS that covers all the necessary requirements, for the black-box view of the system, have been identified the requirement domain is completed. However all the development activities are interactive and at later stages, some of the initial requirements might need to be negotiated due to the capabilities of identified solutions and existing product architectures.

6.2 The Functional System Domain

The next step is to define the functional system of the product. When defining the functional systems the viewpoint is on the main internal functionality of the product, starting from its most high level functionality. This step, corresponds, to activities 3, *Conceptual Functional Model (CFM) development*, and activity 4, System *Boundary Identification*, described by Hutcheson et al. (2007). Here it is however suggested that the main auxiliary functions of the product, identified in activity 10, *Auxiliary Functionality Identification*, would be included from the beginning. The identified systems form the structure for the system DS structure. It can be expected that the highest level of systems definition will remain stable between products within the same product domain. A consistent and logic structure provides support for reuse based product development. The black-box requirements are mapped to the systems which, either fulfil them or are affected (e.g. constrained) by the requirements.
The system item will describe the functions the system provides, its inputs and output and a behavioral model. For each system interface, one of the interfacing systems must be declared an owner of the interface. This means that the design team responsible for this owning system is also responsible for the interface and has the authority to make decisions regarding changes of the interface. The interfaces between the systems are presented explicitly, as child elements of the interface owning system. This is to enable capturing the relevant information of the interface. A traceability link should furthermore be established between interfacing systems.
The next step is to identify system requirements by the use of conceptual behavioral models, as described in activity 5 in FuSE. Instead of modeling the identified system level requirements in the requirement structure, they are added as child items to the systems, in the functional system view domain. This approach is taken since the system level requirements have the system viewpoint; they would have multiple relationships to the product requirement structure and would not fit into the requirement view. Furthermore, since the systems and the system requirements are closely linked and worked on at the same time, it is practical to view them in the same domain. An attribute value is given to the design elements to identify different types of elements in the same structure.
Next the design team identifies and evaluates possible solution concepts for the identified systems, corresponding to activity 6. At this point the solutions are quite abstract. Activities 7, *Developing behavioral models for these solutions*, activity 8, *Updating requirements and functionality*, and 9, *Iterating this process until feasible solutions have been found*, are followed. Hubka et al. (1988) define a process that provides better support designers when working out the form specific design solutions, that can be used at this stage. When adequate solutions have been established, a solution concept description is added to the system view structure. For embedded products a special attention is given to

the technical interfaces as different technologies will interact to provide the required results. Technical interfaces are defined explicitly as child elements of the corresponding system.
The systems are then broken down to sub-systems following activities 11, *CFM development for sub-systems and auxiliary functionality*, 12, *Detailed behavioral modeling including system to sub-system level requirements flowdown*, and 13, *Identification of solutions for auxiliary functionality and sub-systems*, generating the corresponding information items in a hierarchical tree-structure, in the System SD domain. The product design is synthesized in this manner until the detailed design can start. For each step sub-systems, requirements, system interfaces, solutions and technical interfaces are explicitly added to the system model. The resulting information structure is shown in the Functional System domain on Figure 6.

6.3 The Architecture Domains

As system solutions are detailed, the design team begins to unravel the physical components. The same components might be present and have different roles in different systems (Bruun et al., 2013). Therefore if components were presented in the system view, the same component might be duplicated at different locations. Furthermore it is important to communicate and optimize the physical structure of the product to enable platform based development. Therefore, the components are modeled in an architecture view, presenting the architecture that is most suitable for the physical allocation of the components (presenting each component only once). The communication of which components are a part of which solution is accomplished with traceability links between the domains.
For embedded products the SW and HW architecture structures are separated. A description of how design elements from different technical domains work together is provided in the technical interface description, in the system view. This gives the ability to map from the system view, to the SW and HW architectures. A module or a component contains a description of the module, its capabilities and its usage in existing products. Interfaces between modules are documented explicitly in the architecture structures. To ensure that the components are designed in such a way that they enable fulfilment of higher level requirements, the requirements for components and interfaces are explicitly documented in the architecture structures. Figure 6 demonstrates how information is constructed in the requirement, system design and architecture views.

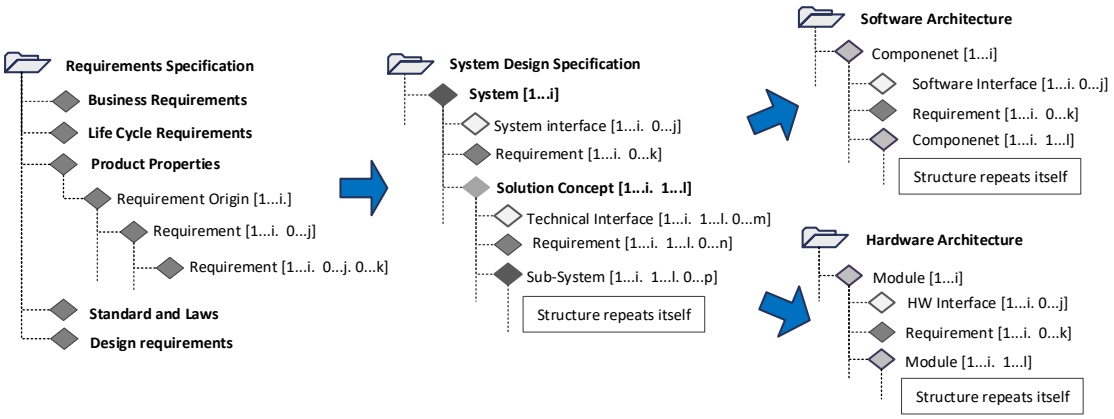

Figure 6. Data structure overview for the three domain views in MEDA

Figure 7 shows an example for the cooling system in a frequency converter product.

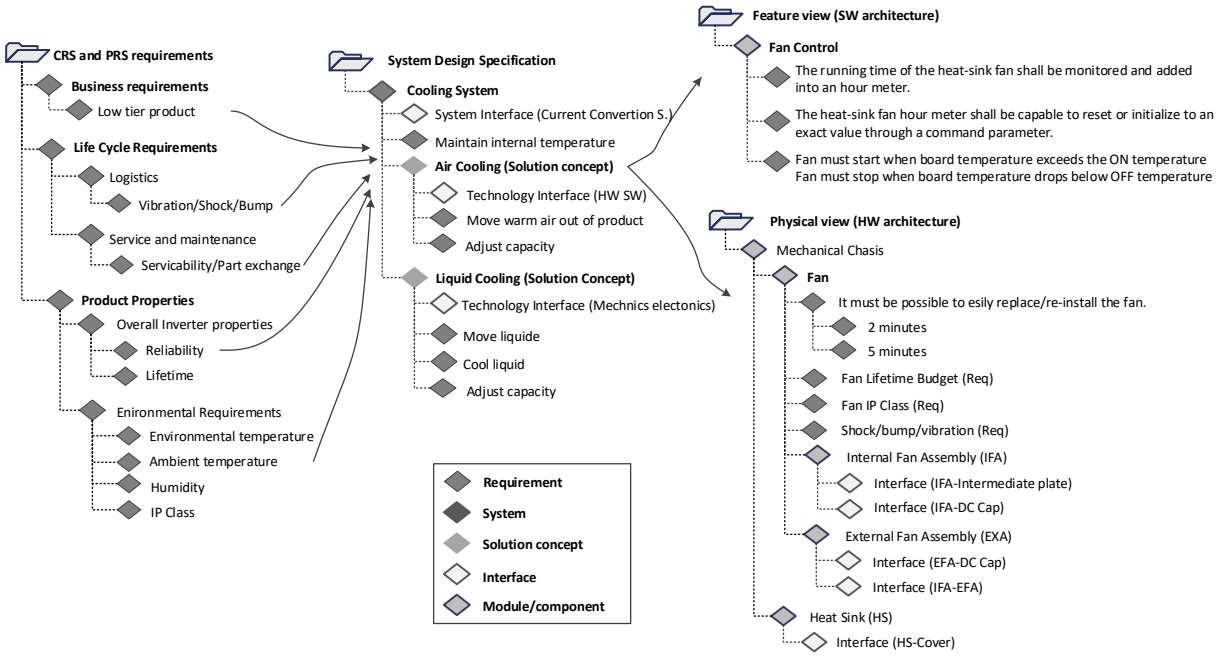

Figure 7. Cooling System for a Frequency converter

The example shows that several customer requirements coming from separate origins all affect the cooling system of the product. Two solution concepts for the cooling system exist; air cooling and liquid cooling. In this example the air cooling is selected based on a set of customer requirements. The air cooling solution concept consists of the mechanical parts of a fan and a heat sink and software to controls the fan. The fan is a critical part for the lifetime of the product and therefore it is critical that one can replace the fan easily and that it has a robust design. These requirements are a result of the selected solution.

When applying the MEDA approach the first step is to define the high level system and architecture structure based on existing product platforms. When the structures are ready the information content for each structure is generated, either based on existing products or build up as new products are developed. When a new project is started, the design team reuses and generates new requirements. The design team then works their way thought the system view, evaluating which solutions can be applied to fulfil the requirements and what the capabilities of the existing product architecture allows. New requirements or new combination of requirements might require implementation of new solutions. This is likely to result in adjustment of technological or physical interfaces and the corresponding design modules. The design team then enters new items in the structural views describing planned items and changes that will be implemented in the detailed design phase. After the design has been implemented the new items are consolidated to the model to be used by future products.

7 DISCUSSION

In this paper the MEDA method is presented. The approach provides structural domain views capturing design information, which is considered sufficient to serve as DSs for the application of reuse driven product design for embedded products, thus addressing **RQ1**. The method is different from other comparable approaches, as it allows information of more than one type to be included in the same domain view and as it includes systematic steps to identify systems and solutions. Finally it allows requirements identification at different abstraction levels throughout the design life-phases.

Both HW and SW components are identified on an abstract level as the solutions are synthesized. The MEDA approach takes into account the relationship between technological domains (**RQ2**) by including technological interface descriptions, specified in dedicated interface items in the structure. This gives clear instructions for how the system solutions are realized in different technologies and thus, how to map solutions to the HW and SW architectures. At the same time HW and SW each have an independent architecture view to enable design.

Design reuse is shortly address by describing how the design team can generate DSs by selecting existing solutions for already implemented requirements, as well as to implement new ones, for describing how

they plan to implement new requirements, prior to doing it. This enables the design team to generate DSs communicating an agreed way for how to design the product providing instructions for the detailed design.
The validation of the suggested method has been established by constructing examples based on existing information, one of which is presented in the paper. The examples show that the information required in an engineering development process can be structured using the method. The approach has furthermore been introduced to a number of lead engineers, whom presume that the method would support a more systematic creation of the system DS. However, to convincingly validate the approach a more complete modeling of an entire product family as well as practical experience of using the method in a product development project would be required.

8 CONCLUSION

In this paper the MEDA method for mapping requirements to product architecture has been presented. The strength of the FuSE is that the method systematically demonstrates how to identify and evaluate solution concepts against requirements and how to break requirements and solutions down to a more detailed level. An important role of DSs is to show how requirements will be fulfilled by design and to describe how technological domains will interact to provide given results. By applying the MEDA method it is possible to generate DSs that provide this knowledge in a concrete manner.

REFERENCES

Sun, N., Mei, X., & Zhang, Y. (2009), A Simplified Systematic Method of Acquiring Design Specifications From Customer Requirements. *Journal of Computing and Information Science in Engineering*, *9*(3), 031004. doi:10.1115/1.3184600

Nuseibeh, B. (2001), Weaving together requirements and architectures. *Computer*, (March), 115–117. Retrieved from http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=910904

Chen, L., Babar,M.A., Nuseibeh, B., "Characterizing Architecturally Significant Requirements", IEEE Computer Society, 2013, pp.38-45.

Cole, R., "The Changing Role of Requirements and Architecture in Systems Engineering", Proceedings of the 2006 IEEE/SMC International Conference on System of Systems Engineering Los Angeles, CA, USA, 2006.

Hull, E., Jackson, K., Dick, J. (2011), Requirements Engineering (Third Edit.). Springer-Verlag London.

Hutcheson, R. S., Mcadams, D. A., Stone, R. B., & Tumer, I. Y. (2007), FUNCTION-BASED SYSTEMS ENGINEERING (FuSE), (August), 1–12.

Akao, Y. (2004), Quality function deployment. Retrieved from http://www.citeulike.org/group/1374/article/3944132

Goncalves-Coelho, a. M. (2005), Improving the use of QFD with Axiomatic Design. *Concurrent Engineering*, *13*(3), 233–239. doi:10.1177/1063293X05056787

Suh, N. (1998), Axiomatic design theory for systems. *Research in Engineering Design*, 189–209. Retrieved from http://link.springer.com/article/10.1007/s001639870001

Tang, D., Zhang, G., & Dai, S. (2009), Design as integration of axiomatic design and design structure matrix. *Robotics and Computer-Integrated Manufacturing*, *25*(3), 610–619. doi:10.1016/j.rcim.2008.04.005

Pohl, K., Böckle, G., and van der Linden, F. J. (2005), Software Product Line Engineering: Foundations, Principles, and Techniques. New York, Springer.

Hvam, L., Mortensen, N. H., and Riis, J. (2008), *Product customization.* Berlin, Springer.

Hubka, V., Andreasen, M.M., and Edger, W.E. (1988), Practical Studies in Systematic Design. London, Butterwort & Co.

Nagel, R. L., Hutcheson, R., Mcadams, D. A., & Stone, R. (2011), Process and event modelling for conceptual design, *22*(3), 145–164.