Identification of a reusable requirements structure for embedded products in a dynamic market environment

Hauksdóttir, Dagný; Mortensen, Niels Henrik; Nielsen, Poul Erik

Published in:
Computers in Industry

Link to article, DOI:
10.1016/j.compind.2012.10.008

Publication date:
2013

Document Version
Early version, also known as pre-print

Identification of a reusable requirements structure for embedded products in a dynamic market environment.

1. Introduction

The main measure of the success of a system is the degree to which it meets its purpose. Therefore, identifying this purpose must be one of the main activities in the development of systems [1]. A survey conducted by the Standish group shows that some of the main reason for project failure can be directly related to incomplete requirements and poor requirement management [2]. The Requirements Specification (RS) is important to create a written agreement among players regarding what will be included in a new product development project. It aligns expectations and represents a contract between development, project managers, customer and "business owners". It has been long recognized that inadequate, incomplete, ambiguous, or inconsistent requirements have a significant impact on the quality of (software) products [1]. Problems with specifications are probably the principal reason for project failure where systems are delivered late, do not meet the real needs of their users, and perform in an unsatisfactory way [3, 4, 5, 6]. Quality RSs prevent uncertainty, ambiguity and misunderstanding between players [7].

Technical products may have hundreds of requirements and the activity of requirement documenting can consume considerable recourses. As product and market complexity increases requirements are becoming a valuable knowledge. Similarly as other valuable knowledge, requirements can be reused rather than reinvented. Reusing similar requirement frameworks is among the most promising ways to reduce time and resources, and increase quality of requirements [8]. By reusing requirements companies can improve the efficiency of documenting and reviewing requirements. Systematically reusing requirements can therefore contribute to reduced time to market and increased product quality.

The requirement structure is an important enabler for controlling and managing requirements reuse. A consistent requirement structure will support companies in the planning and management of requirements activities and enable downstream reuse of other design documents. A good requirement structure will not only need to be sufficient for documenting requirements, but also for efficiently managing activities such as finding, selecting and maintaining requirements. It is not really possible to start systematically reusing requirements unless a consistent and good structure is in place. This topic should therefore be a top priority for every company that wants to reuse requirements across product development projects and needs further attention.

Even though the requirement structure is important for reuse, this topic currently lacks theory and practices. The objective of this paper is to identify what characterizes a good structure for requirement reuse and to suggest a structure that fulfils the identified criterions. The structure shall be used for documenting requirements in a re-usable RS. A RS is a collection of requirements that describes the demanded behaviour and properties of a system or a product. The RS can be a document or it can be a collection of requirement items in a repository. The structure will both be used for a RS containing selected requirements for a product instance and for the collection of reusable requirements. It is about finding a structure that can cover all types of requirements, include all technology fields and support identification and reuse of relevant requirements. This is important to enable companies to gain benefits from reusing requirements in practise. Without having control over the structure, practical challenges could prevent capturing benefits from reuse. The objective is that this structuring technique can be applied to technically complex, embedded products, as they typically involve a high level of system complexity. With the proposed structure the focus is on the high level grouping and structuring of requirements, contributing to creating a mature way to manage the reusable requirements.

The sections of this paper are organized as follows. In the next section, section 2, the criterions defining a good reusable requirement structure are discussed. In section 3, the state of the art methods for structuring and reusing requirements are presented. Section 4 presents a gap analysis comparing the

previous contributions to the criterions identified in section 2. Section 5 presents the proposal for structuring requirements for reuse. In section 6 a case study, where the proposed structure is used to document and reuse requirement for solar inverter products, is described. The paper then concludes in section 7 where the findings of the paper are discussed.

2. What needs to be in place to reuse requirements?

It has been recognized that requirements are valuable information and that reusing existing requirements can be a promising way to increase efficiency in the documentation process and to increase the quality of requirements [8, 9, 10, 11, 12, 13]. It is suggested that both the requirements them self, their structure and other domain knowledge can be reused in new projects. In order to reuse requirements several things should be in place such as well written and correct requirements, a definition of what is general and what is specific, how to manage variability between products, roles and responsibilities for how to document and to maintain the requirements, selection constrains, dependencies or relationships between requirements etc.

However in practice, the first thing that must be done before building a reusable requirements asset is to define a high level grouping and classification patterns which create a system for organising the documented requirements.

When there are hundreds of requirements they can not be documented in one big pile of items and it becomes unavoidable to group the requirements to create some logical overview. Requirement will not be independent, but it will be in contextual relationship with other requirements. Requirements might thus, specify related things, elaborate on or be a rationale for other requirements. In some cases requirements directly affect the value other requirements. There must therefore be a defined structure which defines the logic for how to document requirements. The structure of the requirement repository should make it easy to understand it, retrieve and analyze its items, follow dependency links, trace items back to their rationale and make appropriate changes [11]. This structure is important for all RSs but especially for those that shall be reused. The authors of this paper have identified several criterions that must be realized in order to establish a good requirement structure in general and additional ones to create a good requirement structure for reuse.

The criterions for good requirements structures in general are:

- *Overview and context:* It is important that related requirements can be grouped together so users can easily get an overview and understanding of the content of the RS. This is challenging since different requirements might be related depending on the viewpoint taken.
- *Coverage of all requirements:* The requirement structure must have a place for all requirement topics and types of requirements.
- *Coverage analysis:* The structure must create an overview of whether the RS is complete.
- *Representation to stakeholders:* There are many stakeholders to the RS. The structure must make it easy for different groups of stakeholders to view relevant requirements. It must also be straight forward which stakeholders can provide the necessary information to define the requirements.
- *Reflection of solution domain:* To enable a transparent transformation to design, the structure must reflect the solution domain of the product.

In addition, the criterions for good requirements structures for reuse are:

- *Consistency:* Although the product requirements will differ between product projects the structure must be consistent and stable across products. The names and order of the main groups must be consistent and each requirement should have a certain location in the structure. This will create familiarity of the structure for users, enabling better planning of the requirement documentation and stable connections to other tools.
- *User friendliness:* The reusable structure has some additional criterions regarding user friendliness. Different stakeholders will work with the RS from one project to another. It must be transparent where to find what they are looking for and where to locate requirements.

The structure should create a structural guide for reuse, guiding the users from one requirement to the next creating the order in which requirements are selected.

- *Maintenance:* It must be easy to maintain reusable requirements. Requirements have different characteristics and sources that affect their volatility and maintenance. Therefore, requirements with similar maintenance characteristics should be grouped together. It is beneficial if the grouping of the requirements makes it transparent which stakeholders should be responsible for its maintenance.
- *Downstream reuse:* To enable downstream reuse it is important that the requirements have a consistent location in the structure. It is also important the RS reflects other specifications, such as test specification and that knowledge from the RS can be transferred to other documents, such as the design specification. This enables a stable and transparent connection between requirements items and corresponding items in other specifications.

The goal of this paper is to present a RS structure that will fulfil these criterions.

3 State of the art

The state of the art work related to the proposal of this paper is focused on previous contribution regarding high level grouping, categorizing and structuring for requirement reuse.

3.1 Requirement types

It has been widely recognised that requirements can be defined and divided into different requirement types [9, 11, 14]. Requirement types are used to describe the purpose of requirements and have often been the basis for grouping requirements. The most common types are *functional-* and *non-functional Requirements.*

Functional requirements (FRs) are specifications of what a product should be able to do. That is the functions, activities or actions that are to be part of the product and if carried out contribute to its goals. FRs may specify calculations, technical details, data manipulation and processing and other functionality. FRs are often captured in use cases or work descriptions. FRs can be further described by non-functional or quality requirements.

The *Non-functional Requirements* (NFRs) specify qualities that the product must have or a criteria that the product must meet. They describe the spirit of its appearance, how easy it must be to use, how secure it must be, what laws apply to it and other qualities that must be built into the product.

NFRs are often further specified as quality attributes (QAs). Common QAs are; *look and feel, usability, reliability, performance, maintainability, portability, security and legal requirements*. Most of the QAs that are important for a product emerge from the system as a whole. Because QAs emerge from the structuring principles of the whole system, it will be difficult to change these properties after the system is completed. Finding out these QAs and carefully prioritizing them is crucial for a successful development process [15].

Additionally requirement types, such as compliance requirements, architectural requirements and development requirements have been suggested [11]. The distinction between FRs and NFRs should not be taken in a strict, clear cut sense since the boundary between them is not always clear. For example, functional safety requirements such as warning signals can be considered as functional and safety requirements and similarly NFRs can overlap, such as legal and security requirements [11].

The requirement types are recognised way to categorize requirements. However the downside of the approach is that suggest a separation of FRs and NFRs which is unfeasible to disregard due to their close connection. Furthermore, requirement types don't reflect the problem domain well enough. The requirement types therefore don't provide a sufficient coverage analysis. Finally, how to implement a

high level grouping and structuring that could be implemented in a RS document or a requirement tool is not specified.

3.2 Requirement models

Several theories contributed to formal modelling techniques where requirements are presented and analysed in system models. These approaches make it possible to model, identify and rationalize requirements for complex systems.

One of the tools often used for capturing FRs are Use Case Diagrams [16, 17, 18, 19]. Use case diagrams visually represent the behavioural requirements of a proposed system, from which the analyst can derive the FRs that must be implemented. In object-oriented software development, requirements of different stakeholders are often manifested in use case models which complement the static domain model by dynamic FRs [16]. A number of methods which extend use cases with variability points have been proposed. Each of them applies use case diagrams with an explicit representation of variability to model product line requirements [20, 21, 22]. They differ mainly in the expressiveness of the notation that they support. In addition [23] extended the product line use case specification to include the dependencies between variation points [15]. The use case driven approach is however limited since it only documents FRs and leaves out other requirement types that might be needed for a complete the RS.

NFRs are usually the most difficult and expensive requirements to achieve [10, 24] but are also among the most important system QAs [10]. Goal-oriented approaches to requirements engineering (GORE) [8, 25, 26, 27, 28] are based on refining vague objectives into concrete formal goals and then decomposing these further into sub goals until a set of primitive goals, which can readily be expressed as system requirements, have been derived. Several modelling methods support the application of GORE such as the NFR framework, KAOS and *i** [25]. The NFR framework concentrates on the modelling and analysis of NFRs. The main idea of the approach is to systematically model and refine NFRs and to expose positive and negative influences of different alternatives on these requirements [29]. The main modelling tool that the framework provides is the Soft-Goal interdependency Graph (SIG) [25]. Overall, these NFR models provide a process-oriented approach for dealing with NFRs. Instead of evaluating the final product with respect to whether it meets its NFRs, the emphasis is on trying to analyze and rationalize the development process in terms of NFRs [29, 30].

In Use-Case driven approach, the description of NFRs is not specific enough and in the NFRs framework approach NFRs is separate from dealing with FRs. This increases the complexity of analysis and design in later development stages [29]. Cross cutting concerns include functional concerns and non-functional concerns that crosscut and constrain the FRs. Cross cutting concerns are responsible for producing complex representation that are difficult to understand and maintain. As an example a system might be required to perform a certain function with certain non functional characteristics such as accuracy or processing speed. This would then be a cross cutting non-functional concern. An efficient presentation of cross cutting concerns is essential.

[29] Presents a method called NFR/AUC (AUC stands for Aspectual Use-Case driven approach) for how to adapt the use case diagram and the NFR framework in order to integrate FRs and NFRs and identify crosscutting concerns. FRs are identified and build into the use case diagram and global NFR softgoals are identified. NFRs are associated at key association points in the use case model and decomposed into sub-softgoals. The model identifies functional and non-functional cross cutting concerns and helps to achieve a smooth transition between system analysis and design. Figure 1 presents the association points between the use case diagram and the NFRs.

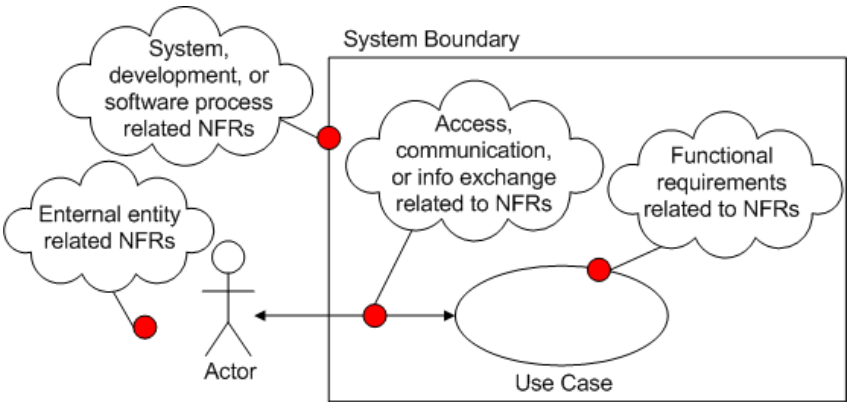

Figure 1: Association points connecting the NFRs to the Use Case Diagram [29].

These use case diagram and the Goal oriented approach have limitations since they don't include all types of requirements. The NFR/AUC method establishes the needed connection between the FRs and NFRs. However, the model is originates from the use case diagram and therefore the starting point is always the functionality which is limiting when documenting static goals or constraints for a product.

To conclude these methods describe modelling techniques to identify, visualize and rationalize requirements for a product. They model system knowledge where requirements entities are presented in relationship with other system entities. There is therefore a misalignment between the application of these modelling techniques and the proposal of this paper. In terms of structuring, the models present requirements that are related to and elaborate on each other. They therefore provide a way to break down requirements, which may be converted to the RS structure. However, a single model is likely to present only a branch of the structure. A high level RS structure that can contain all requirements is therefore still missing.

3.3 Domain engineering

Domain analysis was introduced by Neighbors in the DRACO system [31]. The goal of the domain analysis process is to generate a generic representation of the domain concepts. This model then guides the requirements development [32] or is itself the requirements model [33]. The main interest is in how variability is managed in domain specifications.

The software technology for adaptable, realisable systems (STARS) project developed comprehensive process models for software reuse [34]. During domain analysis they use commonality and variability assumptions to describe the characteristics shared among different systems and how these systems vary. Traditional domain modelling methods, such as the STARS method, focus on the existing systems and the general rules about the domain. This focus, however, can lead to a technical representation of variability and may lead to an insufficient account of potential future requirements.

[35] proposed using two different domain analysis activiteies: application domain analysis and solution domain analysis. The application domain analysis corresponds to the traditional view of domain analysis. The solution domain analysis discovers the limitations of the solution domain technology and finds out how concepts in the application domain and the parameters of variation can be expressed in terms of the solution domain concepts. Coplien (1999) [35] present the importance of analysing both the problem- and solution domain for requirement analysis.

The domain analyses in general identify that to identify requirements each domain should be studied. They also identify that the domain analysis can reveal domain differences which are sources for general and variable requirements. This provides a foundation for a coverage analysis of the specification. However, the domain engineering does not provide a concrete way to structure the RS on a high level, covering all requirements.

3.4 Viewpoint approach

[7, 28, 36, 37, 38, 39, 40, 41, 42] Suggest a viewpoint approach for eliciting and structuring requirements. The preview (Process and Requirements engineering viewpoints) viewpoint [7, 28] is an entity which encapsulates some but not all information about a system's requirement. In the viewpoint approach, requirements are structured according to different views of the system. For each viewpoint a set of requirements are identified. A viewpoint-based approach to requirements engineering recognises that all information about the system requirements cannot be discovered by considering the system from a single perspective. Information from different viewpoints must be integrated to form the complete system specification.
The preview approach introduces a set of concerns which reflect the high level objectives of the system. They are used to align the requirement for the system with the business goals. Concerns are a way of expressing critical holistic requirements which cut across all viewpoints and apply to the system as a whole. Examples of concerns are; safety, cost, availability, functionality, reliability and maintainability. Figure 2 illustrates how concerns cut across possible classes of viewpoints.

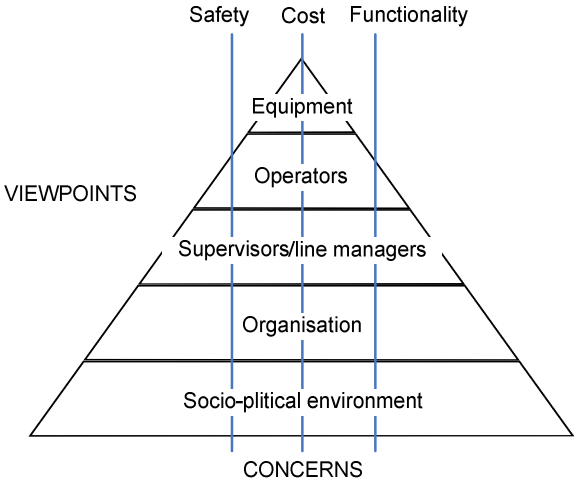

Figure 2: Preview application of viewpoints and concerns [7].

A viewpoint is defined by its focus. No two viewpoints have the same focus but viewpoints may have focuses that interacts or overlap. It is important to identify these overlapping focuses as they help to discover potential requirement conflicts. The notion of focus forms a link between the problem space and the system which is to be developed and provides a base for a coverage analysis. Finally it helps to encapsulate viewpoints and requirements which are potentially reusable across a range of systems. Identifying viewpoints and organising information around them reduces the possibility that critical information will be missed and provides a traceability mechanism for linking requirements with their sources [28].

Mannion et al (1998) [43] proposed using viewpoints to define product line requirements. In their approach requirements viewpoints are created according to the domain concepts. By analysing the requirements organised into viewpoints, reusable requirements are created.

The viewpoint approach is a sufficient way to group requirements. The approach identifies that it is relevant to look at different sources of requirement individually to elicit requirements and to ensure that the coverage of the problem domain. The viewpoint approach resembles to the domain approaches in this way. The viewpoint approaches also don't give a concrete suggestion regarding how a consistent RS for reuse should be structured.

3.5 Hierarchical structuring

Hierarchical classification system is a classification system where entries are arranged based on some hierarchical structure. Hierarchical structuring involves creating a decomposition tree structure of items. The most important hierarchies for requirement reuse are aggregation hierarchy and

Generalisation/specification hierarchies defined by [32]. Aggregation hierarchy describes the decomposition of object types. The objects in the upper levels of the hierarchy represent subsystems, whereas the leaf nodes are the concrete object types (classes or components). *Generalisation/specialisation* hierarchies support the derivation of a specialised object for each target configuration by replacing the generic object type by the target specific object [32].

Kuusela and Savolainen (2000) [15] present a method called Definition hierarchy. A definition hierarchy consist of nodes that represent design objectives and design decisions. *Design objectives* define the goals and the functionality that the future system should have. *Design decisions* are a reflection of the solution domain to the requirements analysis phase. The hierarchy is a logical AND tree, that is, the child requirements are used to define the meaning of the parent requirements. The complete hierarchy has a special root node that describes the purpose of the system under development. The topmost nodes in the definition hierarchy represent the architectural drivers and other QAs that the system is supposed to fulfil. The structure does not include international standards and suggest they should only be used as a checklist during requirement analysis. Instead the major requirements are chosen so that they are the most important requirements in the application domain and will satisfy the needs of the most important stakeholders.

The definition hierarchy reports several benefits. It provides concise specification of a general design objective that otherwise would need a lot of textual explanation. Structuring also helps in sorting out the requirements conflicts and inconsistencies. Reversing the tree structure is helpful when one tries to find missing requirements by finding out if the underlying sub requirements are by themselves enough to completely define the corresponding super requirement. Finding out these missing requirements is aided by the definition hierarchy because it expresses the need an the way how to satisfy the need. The authors also report support for testing. When using the definition hierarchy abstract and general user needs are divided into more defined, compact and testable sub-requirements. High-level user needs are rarely testable as such. In this case one assumes that if all testable sub requirements are fulfilled then also the super requirements are satisfied.

The definition hierarchy provides a high level structuring approach. From the perspective of reusability, structuring the topmost nodes based on architectural drives might cause the structure to be unstable between projects, since different projects are likely to have different drivers. Furthermore the structure focuses less on other none critical requirements and constraints, and standards are excluded. It does therefore not include the complete RS. Finally there is a lack of connection to the problem and solution domain and the coverage of the specification is not ensured.

4. Gap analysis

Summarizing the previous contributions it can be seen that qualified ways for modelling and categorizing requirements have been suggested. To evaluate how well each of the structuring technique fulfils the criterions defined in section 2 they are grouped into categories consistent with the structure in section 3. For section 3.5 only the definition hierarchy is evaluated. Each method category is then compared against each criterion. The method categories include more than one technique or contributions which differ in how well they meet the criterions; however the grouping well reflects the general performance of the techniques against the criterions.

Table 1: Gap analysis.

	Requirement types	Requirements Modelling	Domain analysis	Viewpoint analyis	Defenition Hierarchy
Overview and context	+	+	++	++	+
Coverage of all requirements	+	0	0	+	0
Coverage analysis	+	+	+	++	+
Representation to stakeholders	0	+	+	+	0
Reflection of solution domain	0	++	+	0	++
Consistency	+	0	+	+	0
User friendliness	+	+	+	++	+
Maintenance	+	0	0	0	0
Downstream reuse	+	+	0	0	0

0 Does not support criterion
+ Moderately supports criterion
++ Fulfils criteria.

Table 1 reveals that all of the existing methods meet some of the criterions but none of them accomplishes in meeting all the criterions. Three criterions; overview and context, coverage analysis and user friendliness are well addressed by the current theory regarding domain- and viewpoint analysis. Additionally the criterion, reflection of the solution domain, is considered to be fulfilled by requirement modelling and the definition hierarchy.
On the other hand there are criterions that have not been sufficiently addressed. The criterions: coverage of all requirements, consistency, maintenance and downstream reuse are lacking support. The approaches of requirement types and viewpoint analysis can cover most requirements, but focus on requirements for the product in its main operation and alternative sources of requirements and constraints lack attention. Different characteristics affecting maintenance of requirements in the reusable structure is disregarded in all of the modelling techniques although this is an important enabler for the lifetime of the reusable requirement structure. Finally the criterions: representation to stakeholders and consistency are only moderately supported by the current contributions and need further attention.
A consistent high level structure to group and categorize reusable RSs is missing. There is therefore a gap in the theory regarding a suggestion of a qualified structuring technique that can be applied to create a structure for a reusable RS in a practical product development environment.
In the following proposal the objective is to create a requirement structure where the quality criterions for a reusable structure, identified in section 2, are supported. Since the existing modelling techniques successfully meet a few of the criterions, some of them will be utilized in the proposal. The proposal will thus contribute to filling the gaps that current modelling techniques lack in order to accomplish a good reusable structure for a RS.

5. Proposal about the structure of requirements

The proposal of the Complete Reusable Requirement structure will now be described. The proposed structure has 5 groups of requirements that should be consistent for different products. Each group is broken down to sub groups based on the main domain viewpoints in the problem space of the product. For each sub group a set of requirements can be identified and decomposed hierarchically. In stead of grouping requirements based on requirement types (FRs and NFRs) or QAs, different requirement types are combined within the groups. First a look will be taken at the environment of a traditional embedded product, which is the logic for the proposed structure.

5.1 Analysis of product environment

To fit its purpose, a product should interact with system and users in its environment sufficient way, creating the right outputs and satisfy the expectations of its stakeholders. When documenting requirements for a product, especially for a new product member in an already existing product family the stakeholders tend to start thinking in terms of already existing product features and systems.

However, this is a limiting view and might result in a narrow focus, lack of understanding regarding the real needs of the users and little innovation to better solve these needs. When determining whether requirements are relevant for reuse, front end processes to indentify and elect requirements are still necessary. It is important to support innovation and creativity as well as to analyse the environment of the product and the needs of the customers. If the project shares purpose, customers, users of system domains etc. with previous products it is likely that many requirements can be reused. Once the context is known it is possible to look for requirements that deal with all or part of that context and use them as the source of potentially reusable requirement [8]. To develop quality requirements it is important to maintain focus on the environment of the product and its purpose, and how differences in the environment result in different requirements for the product.
In section 2 the criterions that need to be in place for a good requirement structure are addressed. The structure reflecting the problem space and having a location for all requirements are one of the main criterions. Therefore to create such a structure the problem space of a traditional embedded product was analysed. Figure 3 shows how a problem space of a product might be visualized.

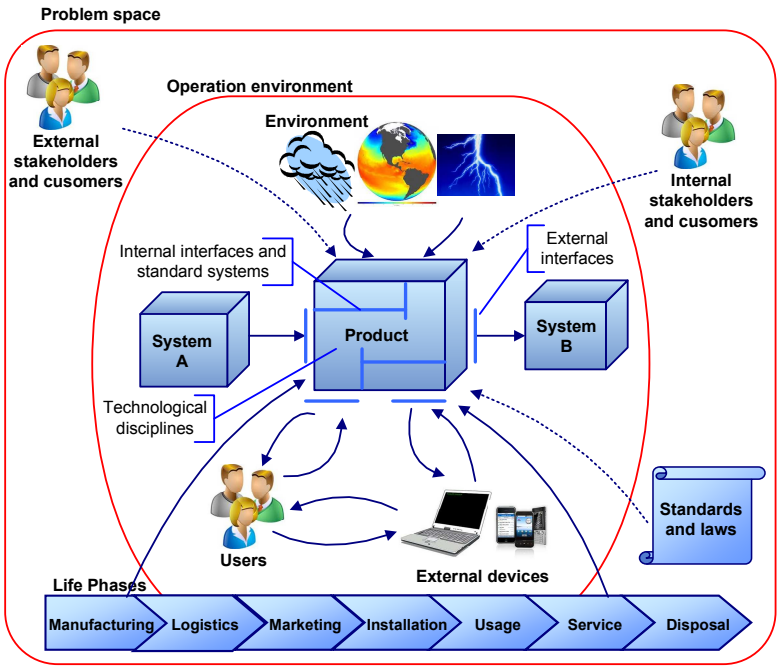

Figure 3: Product environment.

Figure 3 outlines stakeholders and domains in the problem space of a typical embedded product. Here the problem space of the product includes all external factors directly interacting with the product or having interest in its outcome. A boarder is drawn in the problem space between items that directly interact with the product in its operational environment and those that only have an interest in the products design or affect it indirectly.
The items in the operation environment of the product are domains, systems and users directly exchanging material or information with the system in addition to environmental issues specifying the operational condition of the product. For a reusable product family, one can imagine that the operating environment will include the same domain types, but with variations. On the other hand, some domains might only be included in the environment of some products.
Additionally to the product's intended operation, the product will go through life phases such as manufacturing, installation, services and disposal. These life phases will include processes, machines and systems that interact with the product creating constraints and requirements that if addressed in the products design contribute to its overall quality. Since these life phases do not include the main operations of the product they are on the boundary of the operating environment. Products in the product family are likely to have similar life phases as they use the same supply chain.

The product outcome is affected by influences outside the immediate operating environment of the product. Standards and laws applying to the product don't interact with the product directly, but might specify requirements affecting the products operations. Products, with similar application, that operate on the same market are likely to operate within the same standards and laws. Internal stakeholders are people within the company that have an interest in the product such as the owners of the company, portfolio managers, sales and marketing staff, designers, etc. External stakeholders are customers, suppliers, market regulators, etc.
The product has interfaces to the items in its operating environment. These might include standard designs that should be respected in product design. Assuming that the product belongs to a product family it is likely that the product realization will additionally be required to respect standard designs and systems specified for the product family. These belong to the solution domain of the product. The solution domain will also include different technologies for implementing the requirements.
Finally, a transparent connection is required between the RS and other design specifications such as design specification, design implementation, test specifications etc.
This analysis is a source of inspiration for the Complete Reusable Requirement structure presented in the following section. The focus on the domains in the environment of the product is similar to domain engineering and viewpoint analysis discussed in section 3.3 and 3.4. However, in this paper 5 groups that should be consistent for embedded products and cover all the subjects in the problem space of the product are introduced. Additionally the structure aims at addressing the criterions for a good requirement structures such as user friendliness, consistency and maintenance issues.

5.2 Groups

Grouping is a system for classifying similar things together into groups. A group is a collection of objects considered as a whole, creating a planned arrangement of things. Ideally, when subjects or objects of knowledge are sorted into groups it makes them easier to understand and to see the relationships between them. Scientists use classification systems to organize information and objects to help them make sense of the world around them. The world would be a crazy place if we didn't have a way of organizing things. Imagine trying to look for a book in a library not having any sections. The requirement groups create the main sections of the RSs, guiding users looking for information and helping them make sense of the requirements. There must be a section for every requirement topic, it must be obvious for users where to find the information they are looking for and these groups must be consistent for all products within the scope of this proposal (embedded products). A good classification should have groups where all the items in each group have something common and where each item can only be placed in one group, i.e. it is clear if an item is a part of each group or not. Identifying an effective grouping logic should be the first step in creating a good structure.
Often requirements are dedicated to customer requirements belonging to the usage phase of the product. From the environmental analysis, subjects, separate from those typically in focus in domain- and viewpoint analysis, are identified. These subjects result in 5 requirement groups presenting a generalization of the items in the product environment, each having a special purpose. This grouping should be applicable for all embedded products. The groups are the following:

Business Requirements: The purpose of this group is to present the product positioning and success factors of the product. What are the goals and main architectural drivers that the project has to fulfil for the project to be successful? What customer groups must the product satisfy? What must the product cost to be competitive? These are coming from internal and external stakeholders, customers and business owners and define their expectations and targets. The rest of the requirements will need to be aligned to these.
The Business requirements are characterised by a few high level requirements coming from the project definition and scoping. These requirements should be in central focus during the project. Their main stakeholders should be internal and external customers and the business and project owners. Changing one of these requirements would likely to result in major changes for the project.

Laws and standards: These are requirements coming from market regulations and laws. They are out of the company's control, but are something that the products must comply with. This should not only include identifying the standards that shall met, but an identification of what requirements they create for the product behaviour and design. Ordinary, a new project would be a trigger to update or enter new requirements. These requirements should however be maintained between projects so that when a new project starts knowledge about the standard will be up to date and ready to be selected from. This group will also need an immediate update if legal standards change. All though certain standards have been identified the project might choose to exceed those, then the additional requirements are defined as product properties.
Relevant specialists should constantly monitor changes in market regulations and update requirements according to these.

Product properties: In this group, objectives, features, functions and qualities that describe the capabilities of the product in its operating environment are documented. Most of these requirements evolve from the preferences of the users or are the result of domain analysis and other modelling of the product environment and come from the usage phase of the product life time. These are project specific requirements that the company can decide on itself although they must comply with the customers' requests, the operating environment of the product and the standard and law requirements.
The product properties are requirements identified by communicating with lead customers and users of the product and by analysing the work environment of the product. This analysis should be facilitated in the beginning of each project.

Life phase Requirements: Requirements and constrains surfaced from each of the life phases of the product such as production, logistics, installation etc. Often these are not actively documented and cause problems late in the development phase, or even when the product has been launched. By approaching each phase individually, as a source requirements, it should be possible to identify requirements evolving from different sources in the organisation and make them available for developers. Including this knowledge will contribute to improving the overall product quality and cost.
Gathering and documenting these requirements should involve relevant stakeholder for each of the major product life phases since it requires an analysis of each life phase. Not identifying these requirements as a special group increases the risk that they will not be actively included in the RS.

Design requirements: A product belonging to a product family has constraints for how the solution should be realized. Design constraints are those pre-existing design decisions that mandate how the final product shall be structured. These might be using standard product platforms, interfaces with other products, specified standard programming language, standard market designs, alignment with pre-existing products and other issues that limit the designers "freedom" when implementing a solution.
For design requirements the main input comes from the product architectures platform specialists and internal design standards.

Figure 4 gives an overview of the five requirement groups.

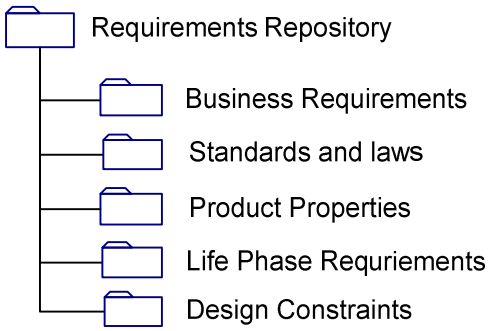

Figure 4: Five main groups of requirements.

The logic for these five groups is the different characteristics and background of the requirements. This is a new classification of requirements which should provide a better grouping than suggested by previous methods since it identifies all objects in the product environment and their characteristics. Each of these groups contains requirements with a specific purpose, origin and stakeholders. Separating the requirements into these groups will therefore provide a clear focus for arranging the requirements. It is an effective generalization of a complicated environment. The five specific groups are furthermore believed to be sufficient to cover and communicate all requirement topics and be consistent for all embedded products. It is believed to be a transparent to guide for users to find the section containing the topic in question and making it easy to see the relationship between the requirements in each group.

5.3 Sub-groups

A subgroup is a distinct and often subordinate group within a group. When analysing and identifying requirements for reuse, domain theory suggest analysing the main domains of the problem space of the future product and indentify general and variable characteristics of the domains. It is suggested that sub-groups reflecting the main domains in the problem space of the product are created. The requirements are documented in a sub-groups corresponding to the domains they surface from. The basic classification is therefore harmonized with where the requirement evolves from. All requirements that are needed to accomplish operation or accordance with a specific domain are documented in the same group, enabling a complete identification of relevant requirements. This is consistent with domain- or viewpoint analysis. This shifts the focus, on the highest structuring level, to the goals of the product in its problem domain and away from the solution space of the product. It is possible that sub-groups will have more than one level. However, it is suggested that the number of levels should be kept at minimum.

When documenting the requirements, it should therefore be transparent where to find the corresponding sub-group to place the requirement. A product family can be expected to share common domains in the environment of the product. When identifying requirements for reuse each of these domains can be analysed and similarities and variation can be identified. It is recognised that there are two types of variability between domains in the environment of a product within a product family. Some domain types exist in the environment of every product within a product family although they have variations between products, while others are only present for some products.

This proposal suggests that the definition of the sub-groups is not default and organizations are given the flexibility to identify their own sub-groups. Different products will have specific sources of requirements and each company will have their own specific concepts. For usability it is important that the sub-groups make sense and are easily recognized by the employees. Typical sub-groups for life phase requirements are the main life-phases in the products life time. For product properties they are the main systems and users in the products operating environments and for standards and laws, they are the major market regulations the product must comply with. Similar sub-groups might be reused between product families and within companies. In each sub-group the requirements are decomposed in a hierarchical structure.

5.4 Requirement types

Some approaches suggest that different types of requirements, e.g. FRs and NFRs create the main classification for grouping requirements [8]. However, often NFRs support FRs and they are often closely related. In this proposal it is therefore suggested that instead of separating requirement types, different types are combined within the groups. Each requirement entity is defined as a certain requirement type. These requirements types are used to ensure that the structure is implemented correctly. Additionally to FR and NFR types (see description in section 3.1) the requirement types included are the following:

Group: This type presents the groups and sub-groups described in section 5.2 and 5.3 above. A group is a heading or a section for requirements. It would therefore not specify a requirement but since this is an entity in the structure it needs to have a defined type. This requirement type should contain a description regarding the content of the group.

Objective: An objective is a high level specification of a capability that the product possesses or an outcome it must achieve. An objective can describe a high level use case or a feature and can be further divided into FRs and NFRs that specify more detailed aspects of the objective.

Typically *solutions* are not supposed to be included in the requirement election. However sometimes it is required that a certain solution shall be used. Sometimes it is necessary to specify a solution to identify more detailed requirements. Then it should be included in the requirement structure, but categorised as solutions and documented as such. It is important to consider the solutions included carefully and only include the necessary ones.

Constraints are not exactly requirements in the sense that they are not raised by the product's demands, but they are issues that have a strong influence on the requirements and the outcome for the product. Constraints cause limitation on how the products design should be realised.

Figures 5 shows how each group contains of different requirement types.

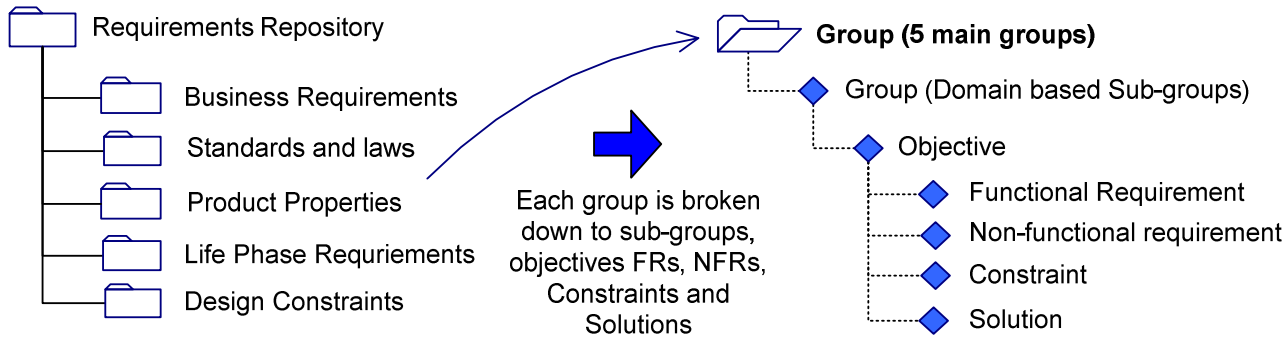

Figure 5: Requirements of different types are combined in the groups.

Figure 5 shows how each group is broken down to sub-groups as described in section 5.3. The main objectives are then found for each sub-group and the objectives are broken down to FRs, NFRs, solutions and constraints. This breakdown shall be followed for all groups. Identifying the different type of items ensures the structure is implemented in the correct way.
The reason for identifying these types is to give the items in the structure identification. Each of the types will have a specific scope and be written in a different way. Identifying the requirement types can therefore be an important enabler of documenting quality requirements.
Embedded products have a complicated connection between different technical disciplines. It should be avoided to write requirements for technologies as such since requirements should describe behaviour and characteristics of the product. It is the implementation of the requirements that should be carried out with different technologies. Identifying objectives regardless of technology is important to identify the real need independent on the implementation technique since many objectives can be carried out using more than one technical discipline. For lower level requirements it might be relevant to identify the implementation technique. Traces between lower level technical requirements and this need is critical for a quality RS.

5.5 Quality aspects

QAs are defined in section 3.1. They can be seen as overall quality targets of the product and are often associated with NFRs. As companies constantly have stricter quality targets understanding how requirements relate to critical quality aspects of the product becomes ever more important. In addition

to the requirement types, a requirement QA is identified for each requirement. Different QAs might be used after each company preferences. The theory [8, 9, 10] provides several definitions of QA categories that can be used.
It is identified that functional requirements also contribute to quality attributes. For example a functional safety requirement contributes to the safety QA. Monitoring functions or error detection functions are concerned with the reliability of the product etc.
Each QA can be filtered and viewed separately providing the structure a cross functional grouping and categorizing. For example, all maintenance QAs can be filtered to analyse whether all requirements concerning maintenance have been included and are consistent. QAs should have a different content and a specific writing stile. Specific metrics should furthermore be specified for each QA which could support authors to define a measurable criterion for the requirements.

5.6 Hierarchical decomposition of requirements

Under each sub-group a set of requirements is structured. A similar hierarchy logic as specified in [15] with an AND tree, where the child requirements are used to define the meaning of the parent requirements is suggested here. This involves decomposing high level requirements to more detailed requirements in an aggregation hierarchy and breaking general requirements down to more specific requirements in Generalisation/specialisation hierarchies (see section 3.5). As described in section 5.3, sub-groups are broken down to high level objects which are broken down to FRs, NFRs, solutions and constraints, which are specified in further detailed in each level.
Models such as the Use case diagram and the NFR/AUC can be useful to identify and analyze requirements. These techniques model separate use-cases or objectives. They have until now been lacking a high-level structure for organizing the models in a sufficient way. If such models are created they can be used to create a corresponding breakdown of requirements in the hierarchy structure. The use-case would be an objective type broken down to FRs and NFRs (See figure 1 and figure 5). Stakeholders working with such models would therefore be able to present them in the structure and use them directly to document requirements. It is however not expect that all objectives will be structured in this way.
When decomposing the requirements for a reusable structure, it is important to consider the life time of the structure. The structure needs to be flexible to add additional requirements without moving altering the initial structure. To accomplish this sometimes it is necessary to create additional nodes in the system when it is expected that other more similar requirements will be added later.
In some cases it might be appropriate to bring the variability up to a high level in the tree structure. Domains which are only present in the environment of some product members might thus be included or excluded on a sub-group level. Variability within the same domain would be more appropriate on a lower level.
To enable easier comprehension of the structure similar decomposition logic should be applied within similar sub-groups. This will create consistency and familiarity in the system, making it easy to understand the dynamics in the structure. Following these guidelines should enable a more stable structure. The case study discusses some practical experiences regarding creating a hierarchical structure.

5.7 Contribution to the reusable requirement structure

With the proposal presented above a way of structuring a RS with the objective of reuse has been suggested. In section 4 previous modelling and structuring techniques are evaluated against the identified criterions in section 2. Some of the criterions identified to create a good reusable requirement structure were considered to be sufficiently addressed by the existing theories. Those were: overview and context, coverage analysis, reflection of the solution domain and user friendliness.
The proposed structure builds on the existing techniques utilizing their contributions. It uses the viewpoint logic for sub-groups in the structure and it suggests a cross categorization build on the definition of QAs. Furthermore the requirement modelling and the definition hierarchy are used to support the hierarchical decomposition of requirements. Combining the current contribution in this way is a new proposal. It is important, since each of the existing proposals only partly fulfil the criterions.

Combining them within the same proposal therefore enables a fulfilment of more criterions by the same technique. This proposal however, additionally seeks to close some of the gaps left open by the current theory as follows:
Coverage of all requirements: A grouping logic presenting 5 main groups is suggested. These groups expand the scope of the structure, and accomplish and promote coverage of all requirements within the same structure. This is important in order for all requirements to be available in the same RS.
Consistency: The groups are also believed to be consistent for all embedded products which would be useful for those companies developing more than one embedded product families. Having a consistent classification might support re-use of requirement between product families.
Maintenance: Considering the characteristics and purpose of the requirements helps to understand the management and maintenance of the requirements in each group. For example, the business requirements and the standards and law requirements shall have different management approaches. This is a new identification that is important for the life time of reusable requirement structure.
Representation to stakeholders: Representation of the requirements to different stakeholders has also been considered in the grouping as each group is focused on a special stakeholder group. Sub-groups and cross categorization should also make it efficient for stakeholders to view information they are interested in and more importantly to identify stakeholders that have knowledge about each sub-group.
Downstream reuse: Finally, a consistent structure in addition to including a requirement group for design requirements should support downstream reuse.
The structure should be applicable and consistent for embedded products and should be ready to be applied by companies that seek to document and reuse requirements. The following section introduces the case study of this paper.

6. Case study

The case study is performed with a global company developing electrical inverters. The experience with working with the proposed structure will be described and some practical insights that surfaced when structuring requirements will be discussed. Since the structure was implemented two projects have reused the structure (i.e. three projects have used the structure in total). The projects belong to the same product family but have different scopes.

6.1 Introduction to Case

The company has previously used a requirement tool to document and manage requirements and focuses highly on requirement reuse. The structure of the reusable requirement repository has developed casually and is not believed to be the best practise. The company wants to improve the quality of the requirements and increase the efficiency and ease of the reuse process. The company creates two RSs; a customer specification, presenting high level stakeholder requirements, aligning the over all scope of the product and a, technical specification which presents more detailed, technical design requirements belonging to the solution space of the product. The RS is used as input for design specifications. The company is now implementing a new product, solar inverters, into the requirement tool and wants to implement an improved way of structuring and managing requirements.
Solar inverters are a type of electrical inverters developed to convert Direct Current (DC) electricity from a Photovoltaic (PV) array into Alternating Current (AC) that is, in this case, supplied to a power utility grid. The product families of solar inverters are roughly divided between single- and three phase, transformer and transformer less, low- and high power inverters. The inverters differentiate on their efficiency, operating range and features. The solar inverter on one hand shall intake current form the PV array where the main goals are having a flexible working range and keeping high efficiency by tracking fluctuations in the current due to solar radiations, shadows etc. On the other hand it shall supply current to a utility grid which requires following regulations for how to react to imbalance on the utility grid and other market standards for electrical products. The inverter has different user groups such as residential users, commercial users and power plants which have different demands for the data output and features. Additionally stakeholders such as government and grid tie owners have interest in the additional electrical supply.

6.2 Requirement tool

The company uses a requirement management tool called CaliberRM provided by Borland. The tool allows users to create groups for requirements and then create a hierarchical structure of requirement entities. The tool provides some predefined attributes and knowledge fields to define requirements and manage project related issues. The tool also enables users to create specific attributes that fit their specific needs. It enables users to define dependencies between requirements, but does not enable further definition of the kind of dependencies.

Reusable requirements are collected in a special reusable requirement repository, company requirements and requirements for products are documented in a project requirement repository. When the scope for a new project has been analyzed the project members use the company repository and look for already existing requirements that can be reused. If the user identifies requirements that can be reused, they map each requirement, from the company repository to the project repository. Mapping means that a clone of the company requirement is created in the project repository where it is given a unique ID number and has a life of its own.

When a project requirement is mapped, it is not possible to change or remove parts of the text field of the requirement without breaking the mapping link (un-mapping the requirement). Therefore when reusing requirements it must be possible to map the entire text field of the requirement without change. When the text of the mapped-from requirement in the reusable structure is changed it automatically changes in the mapped-to requirement in the project. If the reusable requirement changes it is important to evaluate if the project requirement should also be updated, or other vice un-map it.

6.3 Grouping

Since the highest level of structuring, the groups are pre defined they created the initial structure for the RS. The concept of the groups has been well received and understood by the users.

The 5 groups have been successful in creating a location for all requirements. Sometimes there were requirements that could be located in more than one group. For example there might be standards for how the products should be disposed, which can therefore be located in both the *standard and laws* and the *life phase requirements* groups. Requirement regarding tool interfaces for implementation might include physical standard design requirements and could be defined as *design requirements* as well as *life phase requirements*. It is sometimes a matter of judgement where it is best to locate the requirements and of managing conflicts.

The trigger for the consolidation or maintenance activities is different between the groups. Maintenance of product properties requirements will traditionally occur when projects create new requirements or change existing ones. On the other hand maintenance for standard and law requirements will occur when market regulations are changed. Furthermore a change in standard and law requirements is more likely to result in a update of previous products. The company has implemented maintenance processes where the trigger and frequency of the maintenance process is different for the different groups. Finally the split between the groups has been valuable to identify responsibility for maintenance.

6.3.1 Sub-group identification

The process of implementing the suggested structure to the solar inverter product began by indentifying and modelling the relevant sub-groups. Experienced employees were gathered to come up with the right sub-groups. After a few work shops the employees agreed on a common view of the main domains for each group. The result is shown in Figure 6**.**

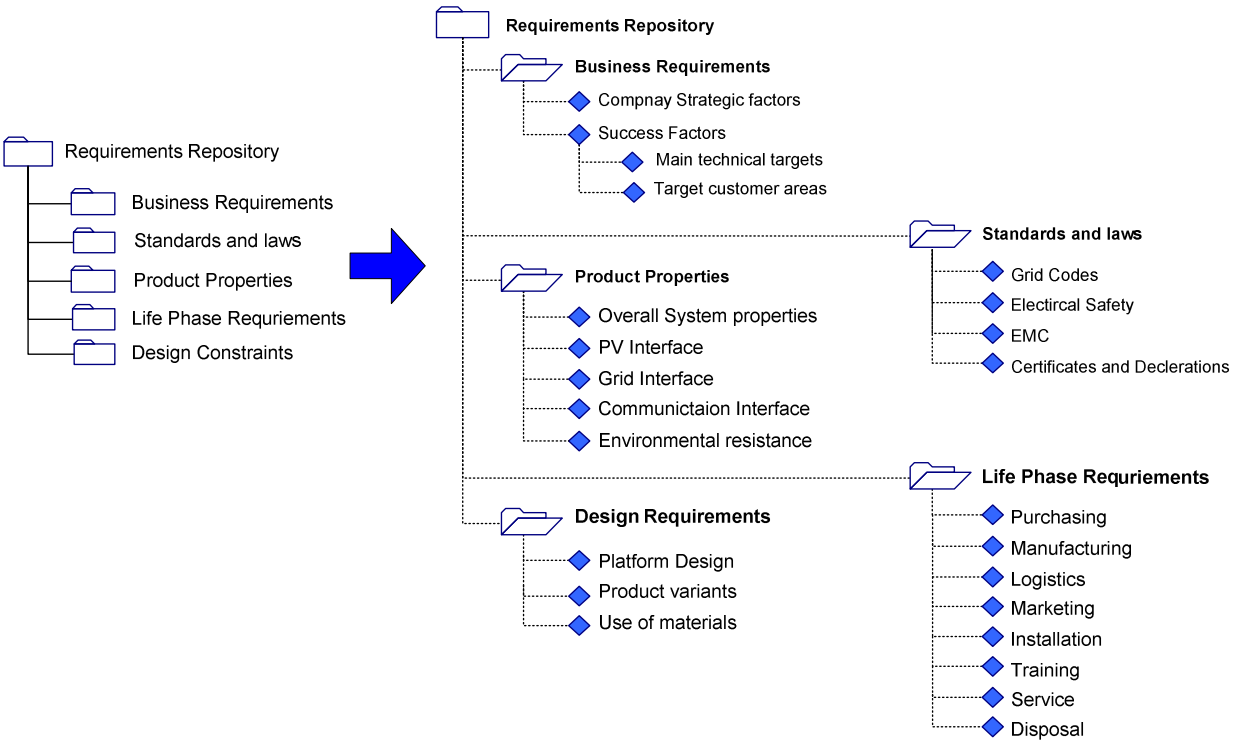

Figure 6: Sub-groups created by the company.

It was clearly advantageous that employees created the structure themselves using their own words and interpretation of the product. It is important that the employees chosen to create the structure are experienced in order to get a correct structure and also to get a buy in from other users. When the sub-groups had been created it was easily realised where it was appropriate to locate most requirements topics since the groups were harmonized with the sources of the requirements. Creating sub-groups improved the structure compared to previous experience of the company. Applying this structure creates sub categorizing and overview of the requirements and reduces the risk of the structure having a huge expansion on single levels in the hierarchy.

The structure has supported the requirement documentation and reuse activities by the groups and sub-groups remaining stable. The number of sub-groups has increased from 42 in the first project to 50 in the third and most are consistent. This has created a certain familiarity with the structure. It has also helped the projects to plan the requirement documentation activity. Having a stable structure allows for easily identifying which domains will be similar and which domains will have variants and allocate employees with the relevant knowledge to write and review the sub-groups. Some domains do not exist in the problem domain of all products. For example a residential inverter will not have operations with a power plant. The domain can be selected or excluded entirely. Thus in some cases the variability is on a higher level in the tree- structure enabling an efficient selection. Other domains such as the PV module are always included in the operation of a solar inverter.
The second and the third projects have had RSs consisting of over 80% reused requirements all though the scope and the architectural drivers of the products have been different for each project. The structure has furthermore remained consistent and is believed to remain so. Having such stable structure enables a better communication for the users.

The focus on the problem domains has helped to keep the focus on the working environment and stakeholders of the product. This makes it easier to identify where to find the relevant information for the requirement analysis and which stakeholders to involve. Structuring based on the domains makes it transparent which technical specialist to contact for a review of the requirement group.

6.3.2 Requirement types

The different requirements types helped to ensure that the structure is correctly implemented and that the content of the requirement is according to its type. As described in section 6.3.1 including sub-groups is important to accomplish a quality break down and categorization in the structure. It is furthermore important to always include an objective to ensure that the actual needs and goals of the product are captured.
There is a risk that the authors with technical background describe the need on a too low technical level. Defining the objective first ensures that requirements are identified in the correct order, finding the general need before discussing software or hardware implementation. It can be valuable to identify the technical disciplines of some of the more detailed requirements. However, the different technologies should not be separated in the structure and it is important to establish an overview of how general objectives are broken down to technical requirements. Combining the different technologies in the structure increased the quality of the requirements and helped users to look further than their technical area.

6.3.3 Requirement QAs

The specified QAs used by the company are based on [8]. They are the following:

- *Look and Feel:* The spirit of the product's appearance. Example: Colours, softness, shape and overall aesthetics.
- *Usability:* Human factors related to the products ease of use and any special usability considerations. Example: understandability, consistency and documentation.
- *Reliability:* To what extent can the product be expected to perform its intended function satisfactorily. Example: recoverability, predictability, accuracy, and mean time between failures.
- *Performance:* How well does the product perform? Example: Processing speed, response time, resource, consumption, throughput and efficiency.
- *Operational:* The operating environment of the product and what considerations must be made for this environment.
- *Maintainability:* Ease of identifying what needs to be changes as well as ease of modification. This also includes requirements related to making it easy to changing the product to accommodate a new environment and with other configurations.
- *Security:* The security and confident ability of the product.
- *Cultural and Political:* Special requirements that come about because of the people involved in the product's development and operation.
- *Legal Requirements:* Laws and standards that apply to the product.

QAs were defined for all requirements in the structure. A learning is that for the definition of QAs to be valuable it is important that they are defined correctly for all requirements. If some requirements are wrongly defined the suggested categorization does not provide the value it should. This did not become as problematic as expected and the requirement types created an understanding of the purpose of the requirements and how they were related to the goals of the product.
As the structure gets more mature these identification becomes valuable. It allows users to filter out specific QAs and evaluate if they are complete and consistent. Usually QAs are general and include adjectives which are difficult to quantify. Currently the company is implementing a listing of standard QA aspects and metrics that should support users to specify the QAs in a more concrete and detailed manner. Reliability is a example of a QA that can have many different aspects that might be difficult to specify and quantify. Having a standardised listing of reliability aspects and metrics would be useful for a user defining reliability requirements in testable way and to evaluate if all relevant aspects of reusability, that the product needs to satisfy, to accomplish this QA have been defined.

6.3.4 Hierarchical decomposition structuring

The project members were ambitious in creating a good and a flexible structure for reuse, while at the same time considering alignment with test and the structure of the RSs, it was often difficult to find the best way to decompose the requirements. It was challenging to predict how the requirement structure would evolve in the future and to implement the needed flexibility in the structure.

Use case modelling has been used by some developers at the company for analysing and modelling system requirements. An approach, for transferring these models into a structure for decomposing use case objectives into a hierarchical requirement structure, was developed. Linking the use case diagram to the structure was well received by the users working with use case diagrams. This motivated them to transfer the use case knowledge to the RS. It was therefore a useful connection between the requirement analysis and the documentation of the RS. The Use case model also provided a good overview of the requirements in the hierarchical structure. Finally, reusing use-cases could be an efficient way to reuse requirements. However, if there is much variation it could become problematic since it requires an update of the use case model on a high level in the structure.
It often occurred that several high level requirements shared the same requirement on a lower level. In this case it could be necessary to repeat the requirement several times. This presents some challenges when maintaining the structure. This can be solved by either moving the requirement to a higher level in the structure or making references between the duplicate requirements.

6.5 Results from using structure

The structure has been accepted and has had positive response from its users. They are able to understand the logic of the structure and the categorization of the requirements. Considering where the requirements evolved from helped the users to understand where requirements should be located and why. This by it self is a critical issue when structuring information and will lead to increased reuse and reduce the lack of duplication in the structure. Both the projects that have used the structure to reuse requirements have had RS consisting of over 80% reused requirements. This shows that the first project seams to have been successful in identifying requirements and that it has been possible to reuse the requirement content.
Analysis of the group requirement types shows that the structure has remained stable. The number of sub-groups has slightly increased between the projects, from 42 in the first project to 50 in the third project. These are consistent groups except for a few that have been included or excluded as a result of the scope of each product. This stability has made it possible to better plan the documentation and maintenance of the structure by allocating users to the sub-groups.
It can also be seen that they reflect the different stakeholders as each group usually has 1-2 authors assigned to it. There is a clear split in which authors assigned to each of the 5 groups. Stakeholders outside the project such as service and maintenance people and other specialists have also been called in to gather further information for groups outside the operating range of the products.
The presented structure has been successful in meeting the criterions for a good requirement structure in general and for reuse. By the experiences with the case study it has been proven to provide an overview and context for the RS. The structure has provided a location for all requirement topics that have come up during the case study and the focus on the problem domain and a classification of the QAs has supported coverage analysis of the RSs.
It is assumed that when the structure matures the stability of the structure will enable downstream reuse.

7. Conclusions

The purpose of this paper has been specified to identify criterions that shall be met to accomplish a good structure for a reusable RS and to find a structure that meets these criterions. A number of quality criterions for a reusable requirement structure have been identified. Analysing previous contributions it was found that some of the existing techniques sufficiently fulfil a few of the criterions. However, individually they each only fulfil a few criterions and some criterions have not been successfully met with any of the existing techniques. According to this a definition of a good requirement structure for a reusable RS is missing.
Parts of the previous contributions have been utilized, in addition to new contribution to create a holistic way of structuring an reusable RS that can meet all the identified quality criterions.
It is concluded that it has been sufficiently argued that the suggested structure fulfils the criterions identified, better than the other identified modelling techniques and is therefore a more qualified structure for RSs than has been suggested previously.

The proposed structure has been used to document and reuse RSs for a company developing electrical inverters. The case study confirms that the structure was sufficient for a reusable requirement specification. The structure remained stable, supporting reuse and the planning of the documentation activity. It also supported the consolidation and maintenance process of the requirements by focusing on the different characteristics of the requirement groups and the stakeholder that should maintain it.
Recently the structuring technique has been applied to a different embedded product family in the company. The high level structure and focus on the environment of the product was well received by the users. The sub-groups of the structure remained stable between the product families. Only a few sub-groups in the product properties and standards and laws groups changed. In fact, it has this consistency highlighted requirement groups that could be reused between the two product families. The structure has also been used for documenting a RS for a PC software product. The 5 high level groups remained stable but the subgroups changed considerably compared to the embedded products. For further research it would be interesting to study further how the structure works for other product types and whether they require different categorization.
By this it is concluded that the proposed structuring technique should be useful and applicable for companies creating a reusable requirement structures that should be used to document RSs for embedded product families.

Acknowledgements
The authors of this paper would like to thank Danfoss Power Electronics for the founding of this article work and the employees at the Danfoss Solar Inverter business unit for their time and participation in the project. Furthermore, we would like to think Torkild Folmer Pedersen for sharing his knowledge and experience regarding requirement management.

References
References
1. B. Gumus, A. Ertas, Requirement management and axiomatic design, Journal of Integrated Design & Process Science 8 (4) (2004) 19-31.

2. E. Hull, K. Jackson and J. Dick, Requirement engineering, Springer, United States of America, 2005.

3. S. Barlas, Anatomy of a Runaway: What Grounded the AAS, IEEE Software 13 (1) (1996) 104–106.

4. GAO, Contracting for Computer Software Development – Serious Problems Require Management Attention to Avoid Wasting Millions, US General Accounting Office (1979)

5. W.W. Gibbs, Software's Chronic Crisis, Scientific American (International Edition) 271 (3) (1994) 72–81.

6. I. Sommerville, P. Sawyer, Viewpoints: principles, problems and a practical approach to requirements engineering, Annals of Software Engineering 3 (1997) 101-130.

7. S. Andriole, The Politics of Requirements Management, IEEE Software 15 (6) (1998) 82 – 84.

8. P. Massonet, A. van Lamsweerde, Analogical Reuse of Requirements Frameworks, In: Proceedings of the IEEE International Symposium on Requirements Engineering, (1997), 26-37

9. S. Roberstson, J. Robertson, Mastering the requirements process, Addison-Weslay, London, 1999.

10. C. López, L.M. Cysneiros, H. Astudillo, NDR Ontology: Sharing and Reusing NFR and Design Rationale Knowledge, in: Proceedings of the first international Workshop on Managing Requirement Knowledge (MARK ´08), 2008, pp. 1-10.

11. A. van Lamsweerde, Requirements Engineering, From system Goals to UML Models with Software specfiications, John Whiley and Sons, Ltd., Chichester, 2009.

12. S. Supakkul, T. Hill, E.A. Oladimeji, L. Chung, Capturing, Organizing and Reusing knowledge of NFRs: An NFR pattern approach, in: Proceedings of the 2009 Second International Workshop on Managing Requirements Knowledge (MARK 2009), 2010.

13. I. Sommerville, P. Sawyer and S. Viller, Viewpoints for requirements elicitation: a practical approach, in: Third international conference on requirements engineering proceedings, 1998, pp. 74-81.

14. D. Samadhiya, D. Chen, S.H. Wang, Quality Models: Role and Value in Software Engineering, in: proceedings of 2nd International Conference on Software Technology and Engineering(ICSTE), 2010, pp. 1320-1324.

15. J. Kuusela and J. Savlainen, Requirement Engineering for Product Families, ICSE, 2000, pp. 61-69.

16. J.H. Hausmann, R. Heckel, G. Taentzer, Detection of Conflicting Functional Requirements in a Use Case-Driven Approach A static analysis technique based on graph transformation, in: Proceedings of the 24th international conference on software engineering, 2002, pp. 105-115.

17. K. Alghathbar, Enhancement of Use Case Diagram to Capture Authorization Requirements, in: Fourth International Conference on Software Engineering Advances, 2009, pp. 394-400.

18. U.I. Hernández, F.J.Á Rodríguez, M.V. Martin, Use Processes – Modeling Requirements Based on Elements of BPMN and UML Use Case Diagrams, in: Proceeding of the 2010 2nd International Conference on Software Technology and Engineering(ICSTE), 2, 2010, pp. 236-240

19. M.A. Laouadi, H. Seridi-Bouchelaghem, M.A. Laouadi, F. Mokhati, A Novel Formal Specification Approach for Real Time Multi-Agent System Functional Requirements, in: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence (LNAI) and Lecture Notes in Bioinformatics) 6251, 2010, pp. 15-27.

20. C. Atkinson, J. Bayer, C. Bunse, E. Kamsties, O. Laitenberger, R. Laqua, et al. Component-Based Product Line Engineering with Uml., London, Addison-Wesley. 2000.

21. G. Halmans and K. Pohl, Communicating the Variability of a Software-Product Family to Customers, Software and Systems modelling Journal 2(1) (2003) 15-36.

22. H. Gomaa, Designing Software Product Lines with Uml – from Use Cases to Pattern Based Software Architectures, Addison-Wesley. 2005.

23. S. Bühne, G. Halmans and K. Pohl, Modelling Dependencies between variation points in Use Case Diagrams, In: Ninth International Workshop on Requirements Engineering: Foundation for Software Quality (REFSQ'03), 2003, pp. 59-70.

24. L.M. Cysneiros and J.C.S. Do Prado Leite, Integrating non-functional requirements into data modeling. In: Proceedings of IEEE International Symposium on Requirements Engineering, 1999, pp. 162-171.

25. http://www.cs.utoronto.ca/~alexei/pub/Lapouchnian-Depth.pdf

26. R. Darimont, A.van Lamsweerde, Formal Refinement Patterns for Goal-Driven Requirements Elaboration, in: Proceedings of the ACM SIGSOFT Symposium on the Foundations of Software Engineering 21 (6) (1996) 179-190.

27. A. van Lamsweerde, R. Darimont, P. Massonet, Goal-Directed Elaboration of Requirements for a Meeting Scheduler, In: Proceedings of the IEEE International Conference on Requirements Engineering, 1995, pp. 194-203.

28. A. van Lamsweerde, Requirements Engineering in the Year 00: A Research Perspective, In: Proceedings of the IEEE International Conference on Software Engineering, 2000, pp. 5-19.

29. X. Liu, S. Liu, X. Zheng, Adapting the NFR Framework to Aspectual Use-Case Driven Approach, in: Proceedings - 7th ACIS International Conference on Software Engineering Research, Management and Applications, 2009, pp. 209-214.

30. J. Mylopoulos, L. Chung, B. Nixon. Representing and Using Non-Functional Requirements: A Process-Oriented Approach, IEEE Transactions on Software Engineering 18 (6) 1992.

31. D.M. Weiss and Lai, Software Product-Line Engineering – a family-Based Software Development Process, Addison-Wesley. 1999.

32. H. Gomaa and L. Kerschberg, Domain Modeling for Software Reuse and Evolution, in: Seventh International Workshop on Computer-Aided Software Engineering, 1995, pp. 162-171.

33. Klingler, The Reuse-Oriented Software Evolution (Rose) Process model. Preston, Chichester, Wiley, 1993.

34. J. Coplien, Multi-Paradigm Design for C++. Reading, Addison-Wesley, 1999.

35. G. Kotonya, I. Sommerville, Requirements engineering with viewpoints, Software Engineering Journal 11(1) (1996) 5-18.

36. A. Goedicke, A. Finkelsetin, B. Finkelstein, J. Nuseibeh, L. Kramer, M. Finkelstein, Viewpoints: A Framework for Integrating Multiple Perspectives in System Development, International Journal of Software Engineering and Knowledge Engineering 2 (1) (1992) 31-57.

37. P. Darke and G. Shanks, User viewpoint modelling: understanding and representing user viewpoints during requirements definition, Info Systems Journal 7 (3) (1997) 213 – 239.

38. P. Darke & G. Shanks, Stakeholder Viewpoints in Requirements Definition: A Framework for Understanding Viewpoint Development Approaches, Requirements Engineering 1 (2) (1996) 88-105

39. G. Kotonya, I. Sommeiville, Requirements engineering with viewpoints, Software Engineering Journal 11 (1) (1996) 5-18.

40. J.M. Neighbors, The Draco Approach to Constructing Software from Reusable Components, IEEE Transactions on Software Engineering 10(5) (1984) 564-574.

41. S. Easterbrook and B. Nuseibeh, Using Viewpoints for Inconsistency Management. Software Engineering Journal 11(1) (1996) 31-43.

42. A. Hunter and B. Nuseibeh, Analysing Inconsistent specifications, in: Proceedings of the Third IEEE International Symposium on Requirements Engineering, IEEE, 1999, pp. 78-86.

43. M. Mannion, B. Keepence and D. Harper, Using Viewpoints to Define Domain Requirements, in: IEEE Computer Society Press, 1998, pp.453-462.

44. http://midwestgreenenergy.com/images/PV-GridSystem2.jpg