Requirement Reuse at Danfoss

Hauksdóttir, Dagný; Vermehren, Arne; Savolainen, Juha

Published in:
Proceedings of the 20th IEEE International Requirements Engineering Conference

Link to article, DOI:
10.1109/RE.2012.6345820

Publication date:
2012

Document Version
Early version, also known as pre-print

Citation (APA):
Abstract—Requirements engineering is an essential activity in creating embedded real-time systems. Companies that produce a number of partially similar products can reduce development time and cost, improve quality and simplify software maintenance by applying reuse practices. Requirements reuse is an essential enabler to achieve effective software reuse. This study describes two different approaches for requirements reuse at Danfoss. The first approach reuses those requirements that are envisioned to be common between two consecutive projects and allows changing and parameterization of parts of the requirements. The second approach organizes all requirements into a common model and explicitly manages variability and different requirement variants in this common model. The results show that both approaches can result in significant savings in reduced effort by reusing common requirements. The first approach was found to be effective when the domain maturity is low and the significant set of requirements were changed from project to project. The second approach allows high reuse potential and significant savings for stable domains, where most requirements tend to be small additions or minor changes of existing requirements.

Keywords—requirements; reuse; process; practice;

I. INTRODUCTION

Software reuse can only happen if one can identify what can be potentially shared among a set of products. Software requirements can express in detail both common and varying characteristics of a product family. A number of approaches have been proposed to enable requirements reuse. Common requirements can always be reused as-is without any modifications. The most challenging question in requirements reuse is: how are varying requirements represented and reused?

Feature modelling [1] has become a popular way to manage and express variability. Feature modelling aims to model high-level functionality of the domain as features and structure them into a feature tree. Variation points are added to guide the selection of the features and also the cross-branch dependencies can be modelled [2]. Requirements are typically used to define the features in detail. Some requirements engineering methods [3,4] support variability management directly in the requirements specification. These methods view product line requirements engineering as a family specification problem. They do not explicitly model individual product line variants; rather, they characterise the whole product line using commonality and variability.

An alternative approach is to have partially separate product specifications. That is, the product line is defined as an enumeration of product specifications (see e.g. [5]). One can also provide a basic product configuration and define all other products as the difference between the basic configuration and product line variants [6,7]. Parameterized requirements allow an easier way to represent for example, a continuous range of values as a part of a variable requirement. While a limited range of discrete values could be represented as a set of alternative features, a better approach is to combine variable and parameterized requirement representation in one approach [8]. [9] Provides general guidelines to enable systematic requirement reuse. It is identified that if requirements are too generic they might leave out important information and be of little value. It is therefore suggested to identify requirements that re-occur for all systems within a domain and insert pluggable parts where specific parameter values can be added.

In this paper two approaches that build on existing theory of requirement structuring for reuse are applied to two product families. The paper reports experiences from using the different approaches in an industrial setting.

II. INTRODUCTION TO DANFOSS

Danfoss Drives has produced Frequency Converters (drives) for 40+ years. Many product series and product families have been developed over time based on the clone & own concept – i.e. copy an existing product specification and modify it into a new product.

The traditional way of reusing requirements at Danfoss Power Electronics (PE) has been the direct requirements reuse. By direct requirements reuse we mean reusing requirements without any adjustments or modifications, thus keeping the project requirements identical to the reusable, company requirements. Company requirements are the complete set of reusable requirements that can be used in the project by applying the direct requirements reuse. A significant reason for adopting this process has been the requirements tool used for managing the company requirements. The currently used tool does not support representing explicit variability in the requirements model.

Direct requirements reuse happens when reusable company requirements are mapped from the company requirements repository to a project repository. This creates a clone of a reused requirement in the project repository. The cloned requirement is a new item and has a specific ID number but has an identical description field to the initial
requirement. The requirements now have a mapping relationship established, which means that when the company requirement is changed the project requirement is automatically updated as well (unless the mapping relationship is broken).

It is thus not possible to adjust project requirements without breaking this mapping link. In direct reuse the focus is on keeping this relationship and reusing requirements exactly as they are. To enable this method, it is necessary to separate requirements into parts that allow each project to select exactly the content and variation it needs.

When working in a project, if the user would like to adjust an existing requirement or enter a variation of it, he or she shall adjust the company requirement or create a new variable part in the company repository, and then map it to the project. However, the users tend not to do this. In practice, they only add new project specific requirements that do not end up being added to the company requirements.

### III. DANFOSS SOLAR INVERTERS

#### A. Background

The Danfoss Solar Inverters (SI) organisation has recently been merged with Danfoss PE. After the merger Danfoss SI needed to adopt many of the processes and tools from the PE organization. Requirement documentation and management was one of the processes that the SI organisation needed to implement.

The Solar Inverter technology has been around for a number of years. However, there are still many uncertainties in the market environment. Laws and standards applying to the product are evolving and it is not clear which types of product will be preferred by the customers. It can therefore be expected that the market will be volatile for a number of years. To manage this volatility, there is a need for flexibility in the adjustment of existing requirements and the specification of new requirements in a very agile way.

#### B. Adjustable Requirement Reuse

The implemented approach at Danfoss SI is called *adjustable requirement reuse*. The approach has been inspired by the work of Lam et al. [9]. In adjustable reuse the requirements are mapped from the company repository to the project repository. If the users documenting requirements need to change the requirements they are allowed to un-map the requirements and make adjustments. The adjustments can either be improvements in the wording of the requirement, adjustments to specific values and parameters or the users might add comments regarding the specific requirement or market. It is therefore not necessary to split the generic and specific part of the requirement into separate requirement nodes. This means that independent variants of requirements are not always collected in the requirement repository.

To replace the mapping relationship, an origin attribute, that specifies the ID of the company requirement where the project requirement originated was introduced. The origin attribute remains even if the requirement is unmapped. Therefore, if a company requirement is changed, the user can make a search for the project requirements that have reused it. This also enables downstream reuse, such as reuse of test specifications.

To support this approach, it was decided to identify *adjustable parts* of the requirements. The general and the variable part of the requirement were identified. The variation points are highlighted with specific colours and fonts, which indicate that the text can be adjusted. The variable parts therefore become adjustable parts, where users should enter the project specific values. If information and values are not highlighted as adjustable parts, it indicates that the users should not adjust them in the project. Figure 1 shows an example of a requirement containing an adjustable part.

![Figure 1. Representation of a general and a variable part.](image)

Using the adjustable approach the quality and readability of each requirement is improved since it is not split up to a general and a variable part. Since it is not required to document every variation in a separate node it keeps the structure of the requirements much more simple. Furthermore, not being bound by a rigorous change process should enable the users to make continuous improvements to the requirements.

A possible risk to this approach is that there is little control over how the users adjust the requirements. Incorrect changes will need to be captured through review processes. This also might require more effort in reviewing the requirements, since if they were not adjusted in the current project, they would have been reviewed before in a previous project. However, the requirements are expected to continuously improve using this method, so the review effort should decrease in the long run. By allowing requirement adjustments in projects the reused requirement becomes generic since it can be used to specify different variants or values. It is therefore not possible to document specific dependencies within requirements in the reusable structure. It is possible to identify that two requirements affect each other, but not precisely how. Finally, by allowing the projects to have their own specific definitions, it can happen that a project might specify requirement variants, which have been already fulfilled by another project. Figure 2 shows the data structure of requirements in adjustable requirement reuse.

In adjustable reuse, projects are allowed to make requirements “their own”. It is vital that the users can rely on the availability of correct and updated information in the company repository. Therefore the main focus will be on keeping the company requirements alive and updated.
The changes and improvements in the projects therefore need to be consolidated. To ensure this, a consolidation approach was implemented based on the concept of requirement owners.

Prior to the documentation all the involved employees received training in requirements management. During the project there was constant support from the requirements specialist, who reviewed the requirements and participated in discussions regarding how to structure sets of challenging requirements.

The first project was able to create a good initial asset of requirements. When the first requirements specification was accepted, the requirements were copied without any changes to the company repository. The second project to document requirements started with the initial requirement asset, mapped from the company repository. The requirements often contained specific information from the first project and still had some room for improvements. The users were allowed to un-map the requirements and make adjustments to improve the requirements or adjust them to the specific product. After the second RS had been accepted, all the requirements in both of the projects were un-mapped. A representative person from the second project and the requirement specialists then reviewed the requirements specification from the perspective of consolidation and generalization of company requirements. Finally, the company repository was updated. In some cases the structure was adjusted, the requirements were made more general and adjustable parts were specified.

D. Results of Change

The main result of the change was that it was practical for the users to document requirements in a more flexible way. The users did not hesitate to reuse requirements even though they didn’t precisely fit in their initial version. They were able to document specific information they had noted from customers or other stakeholders. The employees were furthermore active in improving the requirements.

The first project to document requirements, documented about 530 requirements. The RS was ready in 21 weeks and it is estimated that around 6000 person hours were used for the job. The second project documented about 480 requirements. The requirements specification was ready in 15 weeks and it is estimated that around 4000 person hours were used for the job. The time and resources saved in the project by reusing requirements are therefore considerable. Table 1 shows results regarding the level of reuse in the second project.

<table>
<thead>
<tr>
<th></th>
<th>Mapped</th>
<th>Adjusted</th>
<th>New</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basic Spec.</td>
<td>45 (24%)</td>
<td>111 (61%)</td>
<td>27 (14%)</td>
</tr>
<tr>
<td>Requirement Spec.</td>
<td>162 (54%)</td>
<td>71 (24%)</td>
<td>62 (21%)</td>
</tr>
<tr>
<td>Total</td>
<td>207 (43%)</td>
<td>182 (38%)</td>
<td>88 (18%)</td>
</tr>
</tbody>
</table>

The table shows the number of requirements and the percentage of requirements reused. The main result is the second project was able to reuse in total over 80% of the requirements they documented. 43 % of the requirements are directly reused but 38 % are reused with adjustments. By allowing adjustable reuse we enable a higher level of requirement reuse than would have been possible through previous practice.
Other results suggest that since the structure of the requirements was reasonably stable, it was possible to improve the planning of documenting and reviewing requirements and provided a better overview of the task at hand. The quality of the requirements furthermore improved in the second project. It is certain that as the quality continues to improve, the review efforts will be reduced and the requirements will become more usable in the projects. The task of consolidating the requirements took in total approximately 3 days for two persons, which is not as much effort as was expected. This initial experience shows that resources used for the consolidation can be justified by the savings of resources and time, in the projects, due to reuse of requirements.

IV. DANFOSS DRIVES

A. Background

In 2005 a new Danfoss development center was founded in Beijing with the purpose of developing highly cost optimized and mass produced drives, and one new product family was defined as the initial task. All development and production activities were to be controlled by the new unit. Four technology experts in different fields were stationed in Beijing and local staff were hired in for the twofold task of creating a new product family and building up an independent development site.

From a software (SW) point of view this meant that a number of people who had little knowledge of the drives domain would have to build up competence and make independent decisions over a short span of time. Focus was therefore on specifications and evaluation of existing products, and a very high focus on the RS of the new product. More than half of the SW development effort was spent on documenting the product features and on reviewing the new specification with expert teams from the Danish headquarter. Due to product constraints for memory consumption and performance, it was not possible to reuse much of the existing software implementation. We believe that the success to re-implement with high speed and quality was mainly because of the detailed requirements specifications work.

Having high quality specifications allowed for formal inspections with domain experts from other development sites and enabled good mapping to test specifications and controlled and measurable test coverage. Keeping the RSs as live documents during the full development cycle has created a path to revise decisions, to preserve experiences, and to efficiently handle customer modifications.

Today the department has 100+ employees and is working on their 5th product family. New products have much in common with their predecessors. Basic strategy for new products is still the “clone&down” reuse – i.e. define a new product based on changes compared to an existing product, but behind the scenes a lot has changed.

B. Direct Requirement Reuse

Based on the good experiences with requirements focus, the team has developed a common repository for all requirements of all its products. Any product is defined as a 100% subset of that repository – meaning that the product does not own any requirements by itself, but is instead a “limited view” into the common repository. If a product needs new features adding, they are added to the company repository. If a product needs to modify existing functionality it is done for all products (typical for defect corrections) – or the new/changed feature is added as a requirement variant into the company repository. This is known as Direct Requirements Reuse. The design center has managed to institutionalize this by having all requirements in the common repository and has had sufficient governance to make it work. This approach has significant advantages over the old way of working.

Figure 3 shows the requirement structure in the company Drives repository. Since the requirements are directly reused they must be broken down to smaller parts that enable the user to select precisely what they want. General requirements and variable parameters need to be divided and separate variances of similar requirements need to be presented.

<table>
<thead>
<tr>
<th>Max Reference and Min Reference</th>
<th>Different variants to a Req.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reference handling</td>
<td></td>
</tr>
<tr>
<td>Max/Min Reference and Reference Range</td>
<td>Different parameters of the same Req.</td>
</tr>
<tr>
<td>Max Reference and Min Reference</td>
<td></td>
</tr>
<tr>
<td>Value of REF_Macros (HVAC)</td>
<td></td>
</tr>
<tr>
<td>Value of REF_Macros (Industry)</td>
<td></td>
</tr>
<tr>
<td>Reference Range Choices</td>
<td></td>
</tr>
<tr>
<td>Fixed reference range</td>
<td></td>
</tr>
<tr>
<td>Reference in Local Control Panel (LCP)</td>
<td></td>
</tr>
<tr>
<td>LCP with Potentiometer</td>
<td></td>
</tr>
<tr>
<td>LCP Without Potentiometer</td>
<td></td>
</tr>
</tbody>
</table>

A parameter “MAXIMUM REFERENCE” to define the max reference must be available.

Max reference must be adjustable in the range MinRef to REF_LIMIT.

It must follow the selection in parameter "CONFIGURATION MODE":

Open Loop control/Speed Closed Loop control/Torque control: Max. reference is \( FMAX_1 \) for VVC+ control and \( FMAX_2 \) for U/f control – so the maximum reference is max(\( FMAX_1, FMAX_2 \)).

Process control: External references are scaled in range +/-200% of REF_LIMIT.

Figure 3. Requirements in the Drive approach.

C. Results of the Change

Table 2 shows the number of SW requirements per product family (PF) and the number of hours spent on review (not development) of the SW requirements for these product families.
As shown in table 2, the effort spent per PF is decreasing, while the number of requirements per PF is levelling out, as more and more requirements are reused directly. PF 2 was the product, which created the company repository, and PF 3 was the first one to use it. The products have similar relative complexity, although they are developed for different business segments.

Reusing requirements without adjustments gives a clearer definition of new products. When Word documents were “clone&owned”, it was easy to fix a little here and there and everywhere in specifications, code, tests, but no cross-product compatibility could be guaranteed. With present focus, each feature is evaluated, and only the defined, value-adding features are updated.

Separating the general part and the defined variance of the product family, enables building specific dependencies in the reusable requirement model, identifying how a specific requirement variant is related to another. This allows for capturing engineering knowledge into the model and prevents users from making invalid selections.

Direct reuse furthermore enabled structured coding and testing. When SW requirements are reused, the underlying architecture and code can also be reused, and it makes more sense to automate unit- and system-testing, when the same asset can be used untouched for a number of products. In other words, for the same effort, more features and test coverage can be obtained. Based on our experience, one can achieve much faster time to market and more reliable quality as a direct result of the approach taken.

D. Future Scenario

The present scenario has been implemented for SW requirements only, but needs to be spread to all requirement types including business, design, and lifecycle requirements and various technology areas such as electronics, mechanics, and others. This is especially important for areas, where a given feature could be implemented with different technologies in different products.

Present focus was kept on the product specification as the “source of work”. As the number of product families grows, the effort of maintaining all product specifications based on a common company repository is becoming a bottleneck. Therefore, global teams to maintain features on company level will be assigned. The next step will be to introduce a configuration management system which allows describing a product family as a configuration of company requirements. Then a single product specification can be generated on demand from the common repository.

V. Discussion and Lessons Learnt

One of the preconditions of requirements reuse is that the product line has previously specified a specific requirements variant, which is considered to reasonably fit the needs of the customers. This variant has previously been implemented in an existing product and the cost and risk of reusing that variant is therefore considerably less than creating a new variant. Reusing requirements should guide the customers towards the existing solutions and reduce the complexity of the product family.

In the case of the Solar Inverter products, the correct requirements variants that will best fulfil the customer needs are still being analyzed and identified. At this point it does not make sense to direct the customers to specific requirement variants but rather to find out which requirement variants should be offered. The current asset is expected to expand and each product will differ considerably from the last one. The adjustable reuse can be applied to products within the same domain. For the direct reuse to be applicable, the conditions are stricter. It applies when there exists a defined product family with a predefined scope and dedicated variance.

The Danfoss SI approach allowed efficient work in projects. When there are not many available resources to build up a product family requirement structure, the adjustable method does not require much pre-work to build up a structure, and the requirement structure can evolve and improve continuously through projects. The structure becomes simpler since each requirement is not divided into different variations. However, there is less control over the variability that users introduce during the projects, and there is a risk of non-value adding variability being entered to the product family. This can increase the overall complexity in the product family, and also limit the cross-product compatibility.

The processes of reviewing and consolidating project requirements therefore become essential to manage the adjustable reuse. The changes made to the company requirements during the second consolidation at Danfoss SI were quite extensive. In the future the implementation of the requirement owner concept will be important to maintain the company repository.

The drives product family is mature and guiding customers and designers to existing requirements becomes logical. The drives methods are more restrictive for users working on projects since they can not enter any specific project information or improvements without adding it to the company repository first. However, due to the level of maturity in the product family, the user documenting requirements in a project should be able to create a complete RS without many changes.

It still is necessary to enter new requirements, especially new customer requirements. It therefore becomes essential to implement processes that allow the users to enter new requirements to the company repository in a practical way. Governance of projects changing or adjusting requirements on project level might also be necessary.

When documenting every variable parameter and the difference in a special requirement, the requirements might become split into atomic parts causing the number of requirements and the complexity of the structure to increase. When the lifetime of the structure extends, processes to
review and continuously improve the company repository for maintenance will become necessary.

This method of reuse enables reusing test specifications and solutions. However, if a requirement has a complicated combination of varying parameters and has many variations and versions, this should be reflected in the tests. Additional findings from building the reusable company repositories are the following:

**Rationale matters:** Maintenance of company and product requirements will be the future task of requirement management. One important issue is to define a process for maintaining and cleaning up legacy requirements. One valuable aspect of this is to document the rationale for each requirement. Based on our experience, it is difficult to evaluate the relevance of requirements, without knowledge of why they were originally introduced. This is especially important in the drives domain where the customers expect backwards-compatibility for all products.

**The scope of the requirements:** Some requirements are product or business area specific, whereas others are company- or even world-wide defined and updated. Examples could be reliability measures and certification versus domain specific application and features. It would make sense to structure the company repository in a similar way, so that some requirements can be shared by the whole company or business segment, whereas others are only shared between similar product lines.

**The level of detail:** One approach to handle variability is to break larger requirements into smaller pieces. Fine-grained requirements allow facilitating a large range of variability, but are hard to maintain. However, combined requirements need rework if a variant requires one part of the requirement to be changed. One needs to find a suitable compromise between these aspects. Allowing adjustable reuse in combined requirements is one way to allow variability in parts of the requirements asset where we do not know enough to create a stable fine grained structure of requirements.

**The role of constraints in the requirements model:** When having a requirements repository of all project requirements, one can combine the requirements in any way desired. Many of the possible combinations are illegal in a way that such combination does not make any sense in terms of the application domain, marketing or even basic physics. While in theory one could create a number of constraints in the requirements model to prevent a user to select illegal configurations, in practice this is rarely done. That is because the people deriving the project specific requirements are experts in the application domain and they know the relevant constraints that restrict selections. Second, if one would model all the constraints so that only current products can be derived from the model, it would become fragile, and break when facing evolution. This would significantly increase the evolution cost for a limited derivation time benefit.

**Tailor the requirements reuse approach to the application domain:** The volatility of the domain has a profound effect on how requirements reuse initiatives should be done. The adjustable reuse approach allowed us to introduce requirements reuse already from the second project operating in the same domain. In practice, the approach allows doing domain analysis incrementally, where each project improves the understanding of the domain allowing better reusable assets to be created.

By identifying the adjustable parts, in the Danfoss SI, it has been defined what is generic and what is specific in the requirements. The structure has therefore taken the first step towards being prepared as a proper product family model. When the product Solar Inverter family matures and the requirement structure achieves sufficient quality, it will become relevant for the product family to introduce the explicit variability as in Drives.

We believe that the future common solution for Danfoss will be a combination of the two approaches presented in this study. Meaningful variability will be collected into the requirement model and offered for direct reuse. However, variable parameters will be defined as attributes that the users can specify without creating a new requirement. Finally, heavily varying requirements will be managed through adjustable reuse. However, implementation of such an approach will require advanced tool support going beyond the tools currently used for requirements management. To conclude, we believe that the key message arising from the work is that one needs to choose the methods for requirements reuse based on the maturity of the domain.

**REFERENCES**


