An Industrial Case Study on Large-Scale Variability Management for Product Configuration in the Mobile Handset Domain

Krzysztof Wnuk, Björn Regnell, Jonas Andersson and Samuel Nygren
Dept. of Computer Science, Lund University, Sweden
{krzysztof.wnuk, bjorn.regnell}@cs.lth.se

Abstract

Efficient variability management is a key issue in large-scale product line engineering, where products with different propositions are built on a common platform. Variability management implies challenges both on requirements engineering and configuration management. This paper presents findings from an improvement effort in an industrial case study including the following contributions: problem statements based on an interview study of current practice, an improvement proposal that addresses the challenges found, and an initial validation of the proposal based on interviews with experts from the case company.

1. Introduction

Software Product Lines have already proven to be a successful approach in providing a strategic reuse of assets within an organization [9]. In this context, variability management is considered as one of the key for successful product lines and concerns in all phases of the software product line lifecycle [8]. We experience considerable growth of the amount of variability that has to be managed and supported in software assets. Inspired by the previous fact, we have conducted an industrial case study focusing on the process of variability management at one of our industrial partners in the mobile phone domain. The topic of our investigation was an established product line engineering process [9] in a company that sells over 50 products every year worldwide in millions of exemplars. Our goal for this study is to increase the knowledge of how the products are configured by studying current issues and if possible proposing and evaluating improvements. To address the goal we have formulated three research questions:

Q1: How are variability requirements and variability points managed in software product lines in practice?

Q2: What are the problems with managing variability requirements and product derivation?

Q3: What improvements can be made in managing variability?

The first two questions were addressed by an interview study, were we have investigated the process of *product derivation* [7] and the concept of *managed variability* [9]. By using managed variability we refer to defining and exploiting variability throughout the different life cycle stages of a software product line [9]. In total 29 persons working with requirements engineering, implementation and testing were interviewed in order to understand how the variability is represented, implemented, specified and bound during the product configuration. As a result, a set of challenges is defined and presented in this paper.

To address Q3, we have proposed and evaluated improvements to the current way of working. Our main proposal includes a new structure of variability information that aims at enable linking product configuration to the initial requirements. It includes splitting the configuration into two levels of granularity. Additionally, we propose to use a main product specification with entities that can be consistently applied throughout the whole organization and will address current documentation issues.

Finally, we have empirically evaluated our improvement proposals by applying them to the existing configuration structure in a pilot study. Additionally, we have conducted a survey by sending questionnaires about the potential benefits and drawbacks of our proposal. 28 out of 34 persons have answered our questionnaire. Most of the respondents expressed positive opinions about the proposal and did not express any major obstacles that may apply to it.

The reminder of this paper is organized as follows. In section 2, we describe the industrial context of the case study. In section 3, we provide a description of research methodology. In section 4, we discuss identified problems and issues. In section 5, we describe improvement proposals, which we evaluate in section 6. Section 7 presents related work and the paper is concluded in Section 8.

2. Industrial Context

The case study was performed at the company that has more than 5 000 employees and develops embedded systems for a global market. The company is using a product line approach [9]. Each product line covers different technologies and markets. The software product lines in our case are organized in clusters in two dimensions. The first dimension represents product segments or product technologies, and the second represents the code base that evolves over time. In each of the clusters there is one lead product built from the platform representing most of the platform functionality. The lead product is scaled down to create sub-products and new variants for other markets and customers. Some of the sub-products originating from the main product contain new features [9]. The platform development process is separated from the product development process as described by Deelstra et. al in [7].

Organization. There are three groups of specialists working with the requirements part of the platform project: *Requirements Engineers, Requirements Coordinators* and *Product Requirements Coordinators*. Each technical area in the products domain has a requirements engineers group responsible for covering the progress in the focused field. Their involvement in the projects is mainly focused on the platform project where they supply high level requirements derived from roadmaps, product concepts and customer requirements. They are also main responsible for the scoping process of the platform. Requirements coordinators work between requirements engineers and developers. Their main role is to communicate requirements to the developers and assist with creating detailed design documents and requirements. Product requirements coordinators are responsible for the communication of the requirements between the product planner and requirements engineers on the specific product level.

The *Development Teams* are responsible for implementing the software in the platform. They review the requirements and estimate the effort needed for implementation. Each new functionality is assigned to a primary development team which is responsible for its implementation in the software modules. Newly implemented functionality is later tested before final delivery to the platform. The different modules need to be integrated and compiled to a full system. This stage is done by the *Product Configuration Managers (PCMs)* team which manages the different variants and versions of the products created from the platform. The compiled system is tested by a product focused testing organization, *Product Software Verification.*

Requirements Management Process. The company is using two types of containers to bundle requirements for different purposes: *Features* and *Configuration Packages (CPs).* As a *feature* we consider in this case a bundle of requirements that we can estimate market value and implementation effort and use those values later in the project scoping and prioritization. Configuration packages are used to differentiate the products by selecting different packages for different products. The company is using the similar approach to CPs as described in [10], where a configuration package is a set of requirements grouped to form a logical unit of functionality. Every requirement has to be associated with one or more CPs. The requirements engineers list the changes and CPs in their area of expertise in the *Configuration Package Module*. These modules have dependencies between each other and some of them are mutually exclusive [10]. CPs that are common for all products in a cluster are marked with an attribute stating that these packages cannot be removed from a product configuration. Hardware dependencies, which make individual requirements valid or invalid for different products, are also specified by the use of *Configuration Dependencies* on the requirements level. The model is similar to the Orthogonal Variability Model proposed by Pohl et al [9].

Product Planning. *Product Planners* are responsible for defining products from the platform available in a cluster. They belong to the marketing division in the company so their task is to create an attractive product offer [3] rather than to perform the actual configuration of it. The product planers establish a concept of a new product which induces commercial overview, price range, competitor analysis and gives an overview of the high level requirements. This document serves as a basis for the *Product Configuration Specification*, which specifies the product based on capabilities offered by the platform. The product configuration specification specifies the configuration of a product concerning both software and hardware using the configuration packages defined in the configuration package modules including configuration dependencies. This model is also similar to the Orthogonal Variability Model proposed by Pohl et al [9]. The product configuration specification corresponds to the application variability model of the Orthogonal Variability Model.

Product Configuration Management. Product Configuration Management teams are responsible for integrating, building and managing variants in the cluster. When configuring a new product in the cluster, the product configuration manager uses hardware constraints derived from a hardware specification for each product in a cluster to set and configure the software. At this stage, the traceability from the configuration parameters to the requirements is crucial. This part of the context is the subject for the improvement proposal in section 5.

3. Research Methodology

In order to get a comprehensive picture of how variability management is performed at our case company, we decided to conduct a set of interviews with various employees in various positions within the company. The requirements management tool architecture was also explored to understand how variability is defined at the requirement level. During this phase the persons involved in process improvement for the requirements process were interviewed and consulted with during the exploration of the requirements management process.

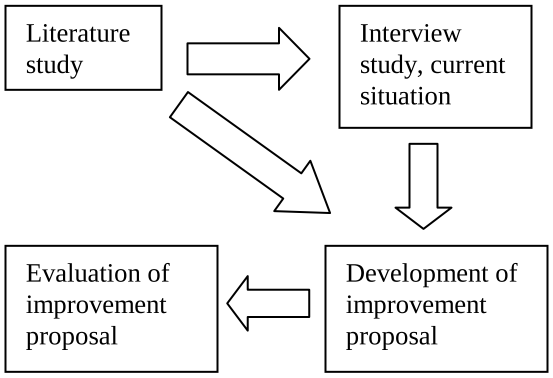

Literature study
Interview study, current situation
Evaluation of improvement proposal
Development of improvement proposal

Figure 1. Research methodology.

The next step was to select key personnel to interview in order to get as many different perspectives how variability is managed and how products are configured as possible. By analyzing the case company's product configuration interface, the amount of variation for different development groups was established. One group with a large amount of product variations and one group with a small amount were selected for further investigation. To cover the whole process of variability, we have involved Product Planners, Requirements Engineers, Requirements Coordinators, Developers and System Testers in our study.

The interviewed persons were selected based on their position in the company. Some persons were recommended by already interviewed. In some cases the person that was asked to participate in our study suggested a colleague as a replacement with the motivation that he was more familiar with the area. In total, 27 persons were interviewed. The interviews were semi-structured in order to allow the interview to change direction depending on the interviewee's answer, and adapted for the different roles and the progress of the interview study. This approach balances between early interviews that were more focused on the general aspects with later more specific interviews. The interviews took approximately one hour. During this time interviewers took notes continuously which were later summarized. During summarization, discrepancies between interviewers interpretation were discussed and, if needed, formulated as questions that were later sent to the interviewee. Apart from the summary, the interviewee also received a model of how he or she perceived the process of variability management. After interviewee approval, which sometimes was done after some minor changes, the data was ready to be analyzed. After interviewing 27 persons, it was decided that the received overview of the current process was satisfactory to proceed with analysis and propose improvements. Sample questions used at the interviews and distribution of interviewed personnel can be accessed at [15].

4. Results

In this section we present the results from our interview study. We describe the different perspectives on the configuration process, configuration activity measurements, and finally the problems that were identified.

4.1 Perspectives on the Configuration Process

Most of the stakeholders have a common view of how products are created. The product projects create a product concept, which is then used by requirements engineers in defining platform requirements. Later in the process the product planners are involved in creation and configuration of new products by creating change requests issues regarding both new and existing functionality. When previously created formal change request is accepted, it is send to the assigned developers team which performs implementation or configuration changes. The differentiation achieved in this manner is not explicitly documented in product specification but only in the minutes from the change board meetings. In the next section, the deviation from this common view is described, as well as the differences from the documented process model.

Product requirements coordinators, requirements coordinators and requirements engineers have limited knowledge about how variability is achieved due to their focus on the platform. They also state that developers do receive most of the configuration instructions through bug report issues from product planners, customer responsible and testers. We discovered that some variability is stated in the requirements' text in an implicit way creating problems with recognition and interpretation at the development phase. Product planners' knowledge about configuration packages is limited and they have not experienced the need for a better product documentation than what is delivered in the concept definition.

The developers express the opinion that information regarding variability is not communicated in a formal way. Instead, they get information about variability through their team leaders in a form of change requests at the late stages of development. These change requests are often used to configure products. The creation of new variation points is done in the platform project, and is therefore often based on assumptions made by the developers out of the previous experiences and informal communication with people involved in the process. The main opinion is that the information about what value that should be assigned to a variation point is possessed by individuals. The information is also not documented sufficiently in formal documents. Requests for new variation points or values are forwarded to the product configuration managers.

Product Configuration Management Perspective. We discovered that the product derivation process is iterative and similar to the one described by Deelsta et al [7]. When a main product from a cluster is created from the platform, it is based on the existing configuration of the previous similar product. This configuration is adjusted to the new hardware specification for the platform. Since the amount of configuration parameters in the configuration file has increased significantly, and they are not sufficiently documented product configuration managers are unable to keep track of all changes.

When a new product has been set up, it is built and sent to the product testers. Their task is to test the product and to try to discover software errors and functionality that might be missing. At this stage it is often difficult for the testers to determine whether errors depend on faulty configuration or software errors. Therefore they create a bug report towards the developers to initiate investigation of the reason of the failure. The errors are corrected by developers and new source code is later sent back to the product configuration manager, which is merging the delivered code from all development groups.

When the sub-product is created, the most similar product configuration is copied from the previous products. Next, the configuration manager responsible for the sub-products is trying to configure the product by checking product structure documentation and other relevant information. The required information is gained from multiple sources, which leads to the double maintenance problem described by Babich [11], where uncertainties about the values of variation points are concluded by comparing with other projects. As a result a time consuming investigations have to be perform and very often influences the speed and correctness of the product configuration.

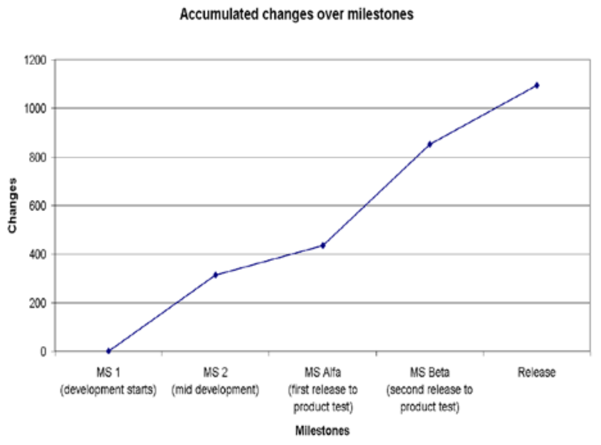


Figure 2. Accumulated changes to the configuration over milestones.

4.2. Configuration Activity Measurements

In order to understand how the configuration is changed over time, change related measurements were defined. The configuration file was chosen for each label of the code base in the cluster. Labels are used to tag revisions of files produced by developers that are to be used by product configuration manager. The differences between each configuration file were calculated in order to get measurements describing how many parameters that were added, deleted or changed. The results are visualized in figures 2 and 3. Note that over 60% of the configuration changes are done after the software has been shipped to the testers (MS Alfa).

The results support our previous observations derived from interviews, where developers admit that they configure the products based on bug reports and change requests. At the time this study was performed, the configuration had over one thousand different parameters available at the product level, spread across a configuration file of thousands of lines. These parameters were controlling over 30 000 variation points in the source code with different levels of granularity. Further analysis showed, that one configuration parameter controls an average of 28 variations points, which suggests that most of the

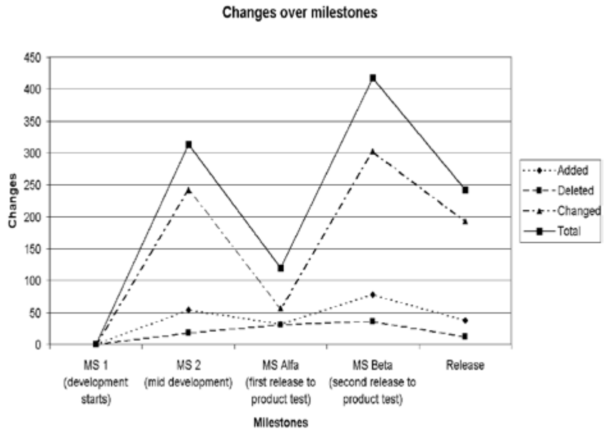


Figure 3. Changes to the configuration over milestones.

variability is quite fragmented. The source code consists of millions of lines of code in more than 10 000 files, giving an average 250 lines of code per variation point.

4.3. Problems Identified

According to Van Der Linden et al [3], the configuration manager should be responsible for maintaining the configuration of all variants and ensuring that the functionality for all products is covered. In our case it remains unclear who is responsible for binding the variation points of the platform to create a specific products. As a result, we experience creation of variation point that have no specific owner. Furthermore, since most of the development and architectural activities are platform focused and a role such as Application Architect or Product Architect responsible for binding variation points of the platform to create specific products is not present in the current organization [9]. The lack of clear responsibilities results in an absence of clear, specific and strategic goals and long term improvements.

The configuration of new products is achieved in an iterative manner between developers, configuration management and testers [7]. Due to the lack of a specific ownership, the configuration is not always properly reviewed, which is often a reason for missing functionality. As a result, testing and maintenance efforts may increase. The knowledge about product derivation and variability is not formalized [7,10].

As mentioned previously, the unrestricted rules for creating and managing variation points results in their excessive creation. Many variation points become obsolete either due to the fact that they were not created for product configuration purposes or because of the complex dependencies. It is undefined who is responsible for removing these obsolete variation points from the configuration file. This fact makes the configuration file hard to manage and overview.

In our case, the flexibility that needs to be copied by standardization of the product line [9], in the sense of amount of variation points is too great and offers many more configuration capabilities than is needed for product configuration and differentiation. The number of variation points, and their structure is too complex to be managed by the people responsible for the product configuration and differentiation. The variability capabilities need to be more standardized and less detailed to handle the costs associated with the flexibility.

The biggest challenge throughout the organization turned out to be the lack of complete product specifications, which may lead to the following problems:

- Time consuming "detective" work where information is gathered through informal communication and unofficial documents.
- Faulty bug reports.
- Double maintenance of fragmented product information that exists in different documents and versions throughout the organization.
- Faulty configuration.
- Critical knowledge about variability configuring products possessed by individuals.
- Increased effort in verifying the configuration of a product.

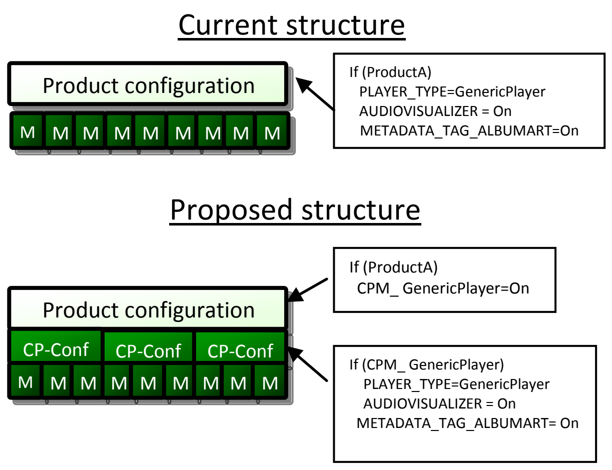
These problems is tackled by the use of unofficial documents specifying the product characteristics for both hardware and software. The documents are created in an informal way and are neither reviewed nor a part of the formal approval process, but still used throughout the organization. These documents and the related process can be improved with respect to configuration management, as uncontrolled documentation procedures may result in unintended product configurations.

5. Improvement Proposal

In order to improve the issues presented in section 4.3, we have developed a set of improvements regarding variability documentation, granularity and management. In order to improve the variation point granularity we propose to introduce an abstraction layer in the configuration interface allowing a clear separation between product configuration and feature configuration. Regarding documentation issues, we propose to use the product configuration specification as the only source for the product differentiation specification, which should be used by all stakeholders involved in the product creation process.

Improved traceability between requirements and variants. Our proposal will reuse the configuration package concept, described in section 2, to associate the configuration parameters with the requirements. The configuration packages should be used by the product planners to configure the products. By associating the configuration packages with the configuration parameters, traceability links to both requirements and configuration parameters will be established. The division into configuration packages should be done in cooperation between developers and requirements engineers to fully capture all possible aspects of variability. Newly created variation points should be explicitly documented and spread across all stakeholders. This approach will result in a more complete traceability between the configuration packages and the configuration interface, and can be a step towards the automatic generation of a product configuration directly from the product configuration specification in the future.

Abstraction layer. The overview of the proposed abstraction level is described in figure 4. In the current structure the configuration file contains all detailed feature configuration on a very low level for all products defined. The file is edited by both product configuration managers and developers and because of its size and granularity it is vulnerable and subject to merge conflicts. Our proposal introduces a new abstraction layer, *CP-Conf*, between the product configuration interface and the software modules. The low level configuration is moved into the lower layer, and a high level product configuration based on the configuration packages is used on the product configuration level. This solution clearly separates the responsibilities between the developers and the product configuration manager, where the developers are



```
If (ProductA)
  PLAYER_TYPE=GenericPlayer
  AUDIOVISUALIZER = On
  METADATA_TAG_ALBUMART=On
```

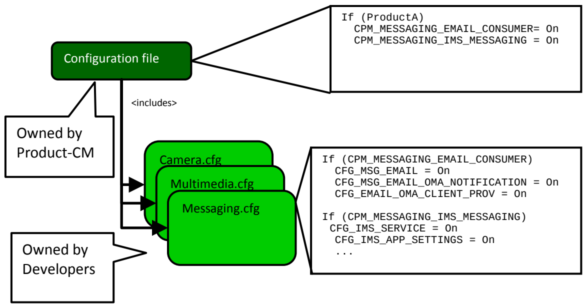
```
If (ProductA)
  CPM_ GenericPlayer=On
```

```
If (CPM_ GenericPlayer)
  PLAYER_TYPE=GenericPlayer
  AUDIOVISUALIZER = On
  METADATA_TAG_ALBUMART= On
```

Figure 4. Overview of the proposed abstraction layer.

becoming responsible for the *CP-Conf* layer and the modules associated with it. The product configuration manager is only responsible for the high level product configuration. To be able to introduce an abstraction level, configuration parameters in the configuration file need to be moved to a separated files where a parameters belonging to a certain development team reside. The specification of selected modules needs to be in these separated files too, since it depends on the selected configuration packages. However, the definition of the module versions is a configuration manager responsibility, associated with the integration of the delivered software modules, and has to be separated from the respective development team's configuration files. This division should remove many of the false merge conflicts. Also, when this abstraction layer is introduced and the parameters are named according to the configuration packages, there should be no need to change the existing variation point naming since the parameters will be moved out from the main configuration file. The solution is described in figure 5.

New configuration parameters. Today the naming of the configuration parameters includes a feature description indicating what functionality the parameter affects. However, the features in the configuration parameters are not mapped to the requirements by including an identifier connected to a specific



```
If (ProductA)
  CPM_MESSAGING_EMAIL_CONSUMER= On
  CPM_MESSAGING_IMS_MESSAGING = On
```

```
If (CPM_MESSAGING_EMAIL_CONSUMER)
  CFG_MSG_EMAIL = On
  CFG_MSG_EMAIL_OMA_NOTIFICATION = On
  CFG_EMAIL_OMA_CLIENT_PROV = On

If (CPM_MESSAGING_IMS_MESSAGING)
 CFG_IMS_SERVICE = On
  CFG_IMS_APP_SETTINGS = On
  ...
```

Figure 5. Configuration is distributed into configuration files according to the concept of Configuration Packages.

requirement. Since the feature names originate from two sources, traceability is based only on human reasoning. We propose a new standard for configuration parameters where four types of parameters are available:

- The existing low level parameters which are presently used for product configuration. To remove or change these parameters is an infeasible work.
- The existing parameters which define the hardware properties of the product should be assigned a prefix CFG_HW. Today many of the parameters created are hardware dependent and could therefore be removed by using the hardware properties instead of creating new parameters. The syntax of the parameters should include the serial number from the hardware requirements specifying its value.
- A new type of parameter for configuration dependencies. The name should include the dependency type (HW/FormFactor/Customer/Market). The syntax can e.g. be CD_<TYPE>_<NAME>.
- An internal binding should be used when software varies non-significantly.

Documenting variability. Currently, the documentation of variation points is not mandatory and resulting in its incompleteness. Since developers in our proposal will be responsible for the lower levels of variability, the documentation process will be simplified by responsible stakeholders' constraining. By introducing traceability between the product level configuration interface and the configuration packages, no further documentation is needed on the higher level. The name standard will be descriptive and in line with the configuration packages. It will enable stakeholders to find more information in the requirements management system, where the configuration packages are defined, described and associated with requirements.

Managing obsolete configurations. Many parameters in the configuration file are obsolete. Because of that we propose that the configuration file should be locked for changes. Parameters that do change but have the same value for all products should be moved to the development team's specific file, and should not be a part of any configuration package. Similar to the configuration parameters, obsolete configuration packages that are not used in any product should be moved out from the software product line. If a configuration package is used in any product it should be incorporated into the platform and removed from the set of CPs.

The values of hardware parameters should be determined by analyzing the higher level hardware parameters. In the same fashion as the configuration packages, the high level hardware parameters should be left at the product configuration level, while its associated low level parameters should be moved to the proposed low abstraction layer and owned by the developers.

Availability of product specifications. All available configuration packages in the platform should be included in the product configuration specification, and a connection to the previously mentioned abstraction layer should be made. By applying this approach, the task of configuring a new product will be simplified and could possibly be automated in the future. The automatic configuration file generation can be based on the configuration packages defined in the requirements management tool.

6. Evaluation of the Proposals

The evaluation of the implemented proposals was carried out as a desktop pilot [12], where the new structure was applied to the existing structure. The desktop pilot was run on a subset of the configurations belonging to two development teams. Two developers from each team, chosen based on their knowledge about configuration parameters, have participated in the redefinition part of the evaluation. The configuration packages defined by requirements engineers were used to group the existing low level configuration parameters, as described in the proposal. This was done in cooperation with the developers. When parameters could not be linked to a certain existing configuration package, the developers had to consider defining a new configuration package, configuration dependencies or hardware requirements. From these lessons learned we can conclude that:

- Packages need to be complemented with a more complex version for greater differentiation possibilities
- Some packages need to have defined dependencies to other packages
- The differences between some of the similar configuration packages need to be described by requirements engineers
- One package may in the future need to be split into several packages that contain end-user functionality and one common package that does not offer any end-user benefits. This one package is dependent on others previously described.
- Problems may arise when new configuration packages need to be created instantly. In this case the bottleneck will be the communication with requirements engineers.
- There are packages that can be removed from the product due to strong dependencies. In this case, product planners should not be allowed to deselect these packages.

After the redefinition of the configuration, the developers were asked to fill in the evaluation form [13], answering questions concerning the improvement proposal and its possible benefits and drawbacks. To get as many answers as possible, the information was held short and concise. The evaluation form was also sent out to all members in the first development group and to half of the members in the second group, totaling with 34 persons. 28 out of 34 persons have answered and the detailed results are accessible in [14].

From the evaluation it can be seen that the participants have been involved in the product configuration. They also see problems with how it is handled today. The proposal was considered as easy to understand and implement.

Some responders mentioned that customer specifications were not addressed enough. One participant also addressed a need for training in variability management. Most of the participants thought that the responsibilities and the separation of product and feature configuration is easy to understand. In the qualitative part of the results, it was confirmed that the workload will be reduced by improved division of responsibilities.

Most responders strongly agreed to that our proposal should increase the quality of products. On the other hand, a few responders claimed that the quality of the products is now high enough and that our proposal will not make any significant difference. The question addressing improvement in the configuration efficiency scored above average, which indicates that this proposal would have a significant effect on efficiency in the way of working rather than end-product quality. This was emphasized by some people who stated that the configuration would become more manageable and less time consuming.

On the question regarding drawbacks there were concerns that the configuration packages may get too large and fail to offer the needed from market perspective detailed level of configuration. It was also mentioned that there will be a stabilization period until the CPs are clearly defined. One responder expects that quick fixes will be hard to handle using CPs, and that there therefore could lead to the *“quick and dirty”* solutions which are hard to maintain. There is a risk that the number of CPs will increase and that the same problems will arise again. Some responders were also worried about customer specific configurations, which the proposal does not specify in detail. Most participants stated that their work will not be affected negatively. Moreover, they stated that there will be less work for the developers with the proposal. The developers would have fewer responsibilities and for some participants their responsibility for product configuration will be completely removed. Overall, the proposal was considered as a better solution than the current way of working.

In the evaluation with the configuration management strategists the responses were positive. Among the positive comments were the possibilities to define a clear process with unambiguous responsibilities, to automate product derivation and verification and to improve the product derivation process. The concerns regarded the need for a streamlined process for managing the configuration packages, including exception handling. Possible dependency problems when a configuration package spans many development teams were also discussed. The overall impression was very positive.

Threats to validity. The way how people were chosen to participate in the interviews can lead to insufficient results. By getting recommendations to which people to interview the risk of getting a subjective picture increases.

The results can be biased by continuous communication with the contact person in the company or by the fact that some concerned stakeholders might have been overlooked in different parts of the case study.

When performing these kind of evaluations, it is difficult to cover all aspects. We are aware that this evaluation only takes a few of the affected development teams into account, and therefore some important information may not be reached. Furthermore, the amount of variation points that each development team is responsible for or shares with other groups varies. Therefore, the scale of affection of the proposal on each development team may vary.

We have not yet performed any evaluation among other stakeholders, like product planning and requirements engineers. Although they are not involved in the technical parts of the proposal, they are part of the process associated with the proposal and it is therefore a drawback not to have these stakeholders represented in the evaluation.

We also see some challenges concerning the ability to maintain the new way of working. It is important that the configuration packages only reflect the current needs for variability and that the configuration packages are not created proactively in the same manner as variation points are created today. It is also important to educate people in order to consistently convince them of the gains achieved about the new praxis.

7. Related empirical work

Industrial case studies in existing literature [1,2,3,4,5,7] describe the process of introducing product lines. These studies report similar problems to those reported in this paper appear. For example, in the Thales case [7] documentation of the platform has deviated from the actual functionality as the platform has evolved. In other cases [1,4] the enormous amount of low level variability in software was reported. Clements et. al [5] reported that the variability was present only on the files and folders level. In the Philips case [3], the problem of too many dependencies between components, resulting in much time spent on integration problems, was reported. Patzke et. al [6] discovered that many of the differentiation point were actually obsolete and not used any more. The company was also struggling with outdated documentation that was not updated regularly.

In most cases a product line approach was introduced in an evolutionary way, apart from one example [4], where all ongoing projects were paused and the resources were moved to the introduction of the product line project. In some cases, the product line was developed around a new architecture, while assets were derived from an existing base e.g. [3]. Sometimes, a new product line was based on the most similar product from the most recent project. Some cases, like [1], claim that their main success was achieved in the architecture and reorganization, and resulted in the change of the hardware to software cost ratio from 35:65 to 80:20.

8. Conclusions

As mentioned in introduction, software product lines improves the quality of the products and reduces the time spent on a product development. However, managing a product line and its variation points efficiently requires a consistent way of working and clear responsibilities. In this case study it has been found that new products are derived by copying the most similar configuration from previous products and iteratively configuring the product between developers, CM and testers. The variability is neither clearly specified nor documented. The responsibilities are unclear. There is no connection between the requirements and the configuration possibilities in the product line. These aspects affect negatively the possibilities to verify the configuration and the time spent on product configuration.

To be able to cope with these issues, improvement consisting of an abstraction layer in the configuration interface have been proposed. This abstraction separates the low level feature configuration from the high level product configuration, and establishes a traceability from requirements to configuration. To clarify the product configuration and ensure that everyone is working consistently, we propose that a product specification, based on these configuration packages, is used throughout the company. Below, we summarize identified problems and corresponding possible improvements:

- *Large number of variation points with an unmanageable granularity.* Variation points are encapsulated into configuration packages, separating the high level configuration from the low level configuration, and resolving the granularity issues.
- *Unclear responsibilities and unstable process for the product configuration.* By dividing the configuration into different layers and proposing responsibilities are clarified.
- *No clear traceability between configuration parameters and initial requirements.* By introducing an abstraction layer based on configuration packages, the configurations are directly linked to the initial requirements.
- *No complete product specification available.* A new and managed product specification

based on configuration packages are spread throughout the organization and used by all stakeholders.
- *Products are configured in an inefficient and iterative process without using the initial requirements.* By the use of a complete product specification and a configuration interface based on the same configuration packages, the configuration can be done at early stage.

The evaluation of our proposal shows that the developers are coherently positive to the suggested improvements. To validate out proposals, the changes were simulated together with two development teams. The results showed no major obstacles, but emphasized the importance of cooperation between the requirements engineers and the developers in the definition of the configuration packages. The expectations of this proposal are as follows:

- to reduce effort and time spent on iterative configuration,
- to ensure a higher product quality by improved product verification,
- to state more clear responsibilities among stakeholders,
- to make the information concerning variability within the company more accessible.

It is stated in [9] that explicit documentation of variability can help to improve making decisions, communication and traceability. Following [9] we can also conclude that introducing abstraction levels for variation points and variants improves understanding and management of software product line variability. As a result, we conclude, that our improvement proposals may be relevant for other contexts by addressing the general issue of variability in software product lines with abstraction mechanisms on both requirements and realization level [8].

This paper contributes in a detailed investigation on product derivation from a large software product line, which addresses research question Q1. Question 2 is addressed in section 4.3 as a set of challenges in practice. Finally, Q3 is addressed by the improvement proposals, decribed in section 5 that may increasing product quality and decreasing the effort needed for product defiviation.

Acknowledgements. This work is supported by VINNOVA (Swedish Agency for Innovation Systems) within the UPITER project. Special acknowledgements to Per Åsfält for valuable contributions on problem statements and research direction.

9. References

[1] L. Brownsword and P. Clements, “A Case Study in Successful Product Line Development”, Technical Report no. CMU/SEI-96-TR-016, Carnegie–Mellon Software Engineering Institute, Pittsburgh USA, 1996.

[2] A. Jaaksi, “Developing mobile browsers in a product line”, *IEEE Software,* IEEE Computer Society, 2002, pp. 73-80.

[3] Linden, Frank J., K.van der Schmid and E. Rommes, *Software Product Lines in Action The Best Industrial Practice in Product Line Engineering,* Springer-Verlag, Berlin Heidelberg, 2007.

[4] Clements P. and L. Northrop, *Software Product Lines: Practices and Patterns.* Addison-Wesley Professional,2002.

[5] Clements P. and L. Northrop, *Salion, Inc.: A Software Product Line Case Study,* Technical Report CMU/SEI-2002-TR-038, Carnegie Mellon Software Engineering Institute, Pittsburgh , 2002.

[6] T. Patzke, R. Kolb, D. Muthig and K. Yamauchi, “Refactoring a legacy component for reuse in a software product line: a case study”, *Journal of Software Maintenance and Evolution: Research and Practice,* John Wiley & Sons, UK, 2006, pp.109-132.

[7] S. Deelstra, M. Sinnena and J. Bosch, ”Product Deriviation in software product families: a case study”, *The Journal of Systems and Software,* Elsevier, New York USA, 200, pp.173-194.

[8] J. Bosch, G. Florijn, D. Greefhorst, J. Kuusela, J. H. Obbink, K. Pohl, “Variability Issues in Software Product Lines”, Software Product-Family Engineering. 4th International Workshop, Springer-Verlag, Bilbao, Spain, 3-5 Oct. 2001, pp. 13-21.

[9] Pohl, C., G. Böckle, and F. J. van der Linden, *Software Product Line Engineering: Foundations, Principles and Techniques,* Springer-Verlag, New York USA, 2005.

[10] Bosch J., *Design and Use of Software Architectures Adopting and evolving a product-line approach,* ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 2000.

[11] Babich, W.A, *Software configuration management: coordination for team productivity.* Addison-Wesley Longman Publishing Co.,Inc., Boston, MA USA, 1986.

[12] R. L. Glass, “Pilot Studies: What, Why and How”, *Journal of Systems and Software,* Elsevier Science Inc, New York USA, 1997, pp. 85-97.

[13] Evaluation form can be accessed at http://www.cs.lth.se/home/Krzysztof_Wnuk/VaMoS_2009/EvaluationForms.pdf

[14] Results of evaluation can be accessed at http://www.cs.lth.se/home/Krzysztof_Wnuk/VaMoS_2009/ResultsOfTheEvaluation.pdf

[15] The interview’s instrument and participants distribution can be accessed at http://www.cs.lth.se/home/Krzysztof_Wnuk/VaMoS_2009/InterviewInstumentAndDistribution.pdf