Case study: Feature Models for Component Selection in Mobile Applications

Christer Thörn

ISSN 1404–0018
Research Report 09:1

SCHOOL OF ENGINEERING
JÖNKÖPING UNIVERSITY

Case study: Feature Models for Component Selection in Mobile Applications

Christer Thörn

Information Engineering Research Group
Department of Computer and Electrical Engineering
School of Engineering, Jönköping university
Jönköping, SWEDEN

ISSN 1404–0018
Research Report 09:1

School of Engineering
Jönköping university

Case study

Abstract

The mobile applications domain is variability intensive and characterized by deployment of applications on a variety of mobile handsets and operating systems, while maintaining a short development cycle. The circumstances lend themselves well to software product line approaches and similar qualified software reuse approaches. This report describes the experiences and results of using feature modeling to structure, document, visualize, and disseminate information about reusable assets with a mobile applications developer. It details the challenges met in finding tools and approaches supporting a large scale model, developed and used in an organization that is geographically distributed, and integrating the modeling benefits in the existing practices. Pairwise comparison is used as a means to prioritize and evaluate the quality of a feature model describing a component framework used for developing mobile applications. Indicators that give positive or negative contributions to the quality of the feature model are identified and lend support to the idea that the quality of a feature model can be influenced to better suit the intended uses and users of the model. The result of the efforts is a feature model, describing a component framework of reusable assets, contained in a wiki-style tool, and introduced in the organization's practices while being minimally invasive.

Keywords

Model-Based Software Engineering, Requirements, Software Modeling, Domain Engineering, Feature models

Table of Contents

1 Introduction **1**

2 Project Plan **3**

2.1 Study validity 4

3 Preliminaries **4**

3.1 Variability modeling and feature models 4

3.2 The mobile application domain 6

3.3 The company Tactel AB 7

3.4 Development of current approaches 8

3.5 The framework 8

3.6 Model quality 8

3.6.1 Changeability 9

3.6.2 Re-usability 10

3.6.3 Formalness 10

3.6.4 Mobility 10

3.6.5 Correctness 10

3.6.6 Usability 10

3.7 Pairwise comparison 11

4 Methodology for Feature Modeling **11**

4.1 The meta-model for feature models 14

5 Tool for Feature Modeling **15**

5.1 Scaling of modeling and visualization 15

5.2 Distribution of modeling and information 16

5.3 Tool requirements 17

5.4 Implementation 18

6 Modeling Activites **19**

6.1 Scoping 19

6.2 Training 20

6.3 Workshops 20

6.4 Seeding 21

7 Modeling Results **22**

8 Validation and Evaluation **22**
8.1 Quality indicators 24
8.2 Discussion and interpretation of results 25
9 Deployment and Integration **25**
9.1 Deploying the model in the organization 26
9.2 Integrating the model in practice 26
10 Conclusions **27**
11 Summary **27**
References **28**

1 Introduction

The mobile applications domain is typical for a business domain that can benefit from software product line approaches. It is characterized by a plethora of handsets offering various functionality and having specific constraints and limitations in terms of hardware and software capability. Carriers and subscription services delivered by content providers add further complexity, making development of mobile applications very variability demanding.

Software reuse and software product lines [1] are seen as potential approaches to alleviate the problems found in many software engineering contexts where there are many software variants and high levels of variable functionality in the software. Variability analysis and management is essential before applying a product line approach. Modeling has proven to be useful for these activities [2], and benefits of a variability model include, for instance, reduced risk of duplicated development efforts, higher communicatability and visibility of current capabilities [3].

Modeling reusable assets for software development is a highly recommended practice in software family and software product line approaches. These approaches are centered around constructing software assets for reuse and devising applications by assembling those assets. The idea is to get away from opportunistic reuse and instead achieve systematic and planned reuse, thereby reaching economy of scale benefits, and also higher product quality since the software assets are used in more products and thereby receive more practical testing and bug fixes. Having a software product line approach also makes it easier to create variants of the software products that satisfy different markets and customers and reduces maintenance since changes are propagated to all new products using the assets.

As the number of products using the core assets increase and the variants of the assets and applications grow, it becomes convenient to use models to describe this variability. A popular method for variability modeling is feature models that show the commonality and variability of a product line in terms of features, relations and feature groups. Feature modeling helps keeping track of the parts and components that are available in the product line, which parts that are dependent on each other in order to work, which assets that produce conflicts in the product, what alternatives that are available to implement a certain requirement, certain combinations of core assets that might cause unwanted interactions, etc.

Among often touted benefits of modeling we find the ability to test designs and solutions before devoting resources to implementation, abstraction of complex mechanisms, and visualization of the essentials of the modeled artifacts. Modeling techniques can thus be an important support for planning and structuring development efforts.

This report describes the feature modeling of an in-house component framework for mobile applications development, the approaches and tools used, and the experiences made. Although there is plenty of generic advise on feature modeling available and various tools offered, the context and demands of the company provide circumstances that are not catered to by the available methods and tools. In particular, this report looks at the demands that scaling of the feature model pose and the integration of the feature model and feature modeling activities in the organization's current development practices, and the evaluation of the produced model.

There are several published guidelines and methods for feature modeling in books, technical reports, conference proceedings, etc. Many of them do not regard the quality of the produced feature models, with the notable exceptions of [4] and [5], that mention the importance of usability in models. The quality aspect considered in most literature on software product lines is that of the products coming out of the product line. An often touted aspect of product lines is that the quality of the products increases when using a product line approach, since the quality of the constituent components increases. In software processes, quality in products is seen as emerging from having quality in the process and development activities. Taken together, it is therefore natural to assume that quality in the participating activities and models in software product lines should also be considered to improve the quality of the resulting products.

We used prioritization of quality factors as a means to determine what properties of models that influence the quality attributes of a feature model positively or negatively. The research questions we seek to answer in this work can be formulated thusly:

(Q1) What quality attributes should be compared?

(Q2) How can pairwise comparison be used for evaluating quality in feature models?

(Q3) What are the indicators of a quality in a feature model?

(Q4) What results from prioritization can be used as means of directing the modeling activities to reach desired qualities in a feature model?

We adress the research questions by applying prioritization and evaluation to a produced feature model and find, as reported in Section 8, that there are indicators that contribute positively or negatively to qualities in the model, and that the indicators are adjustable.

The project was expected to result in methods, tools, processes or artefacts that would contribute to more efficient management of variability aspects in the organization. A criterion used was that as many as possible in the organization should benefit from the work conducted.

2 Project Plan

This project was the result of an interest from Tactel AB and School of Engineering (Tekniska Högskolan i Jönköping, JTH) to find some common interests where advanced software engineering concepts could benefit Tactel, and at the same time provide industrial application cases for research conducted at JTH.

After some initial meetings where the research focus and expertise of JTH was presented, and the current practices and development situation in Tactel was explained, the matter of variability management seemed to provide an interesting common ground. Tactel appeared to work in a very variation intensive domain and have a need for methods to manage the versions and variants offered to customers as well as internally. It was therefore decided to start a project to try out variability modeling on a set of products.

Initial meetings were held in September 2008, modeling activities started around December 2008 and were finished by May 2009. The following list shows the main activities of the project. They are further detailed in Section 6.

1. Setup
   (a) NDA.
   (b) Tool and methodology choice.
   (c) Rough scheduling.
   (d) Context inventory covering organization, domain, market, process etc.
2. Scoping
   (a) Choice of product set to cover.
   (b) Selection of sources/staff.
   (c) Scheduling and scope inventory.
3. Tool development
4. Quality targeting
   (a) Choice of usage scenario.
   (b) Choice of prioritized qualities of the model.
5. Modeling
   (a) Customization of methodology.
   (b) Tool and template preparation.
   (c) Modeling sessions.

6. Evaluation and validation
   (a) Model evaluation.
   (b) Process evaluation.
7. Deployment
8. Reporting

2.1 Study validity

As with all research based on industrial cases, the study reported in this report is hard to generalize to other contexts and organizations in the traditional sense of generalization. While researchers prefer experiments where as many variables as possible are eliminated, practitioners tend to favor case studies and experience reports, as the more complex settings in a case study makes it easier to relate to the practitioners own organization [6]. A case study is not intended to generalize to a population in the way that for instance a survey does. Instead, case study results generalize to a theoretical proposition, in our case lending supporting evidence that prioritization can be used to guide and evaluate the quality of feature models.

3 Preliminaries

This section describes some preliminaries of variability modeling and also accounts for the organization, business domain and the subject of the feature model described in this report. It also describes quality in models and the means used to evaluate the quality of the feature model produced in the project.

3.1 Variability modeling and feature models

The purpose of variability modeling is to structure, formalize and visualize the elements and relations of a domain or set of products. Commonality denotes the shared components or elements of a product set, while variability represent the variation of the products by means of optional or alternative elements. Variability modeling as we usually see it today originates in the domain and application engineering practices from the early 1990s.

A common method for variability modeling, and the one used in the case described in this report, is feature modeling with its roots in domain analysis. The commonality and variability are structured as features, each feature being a property of the product adding some value or being of interest to some stakeholder. Various

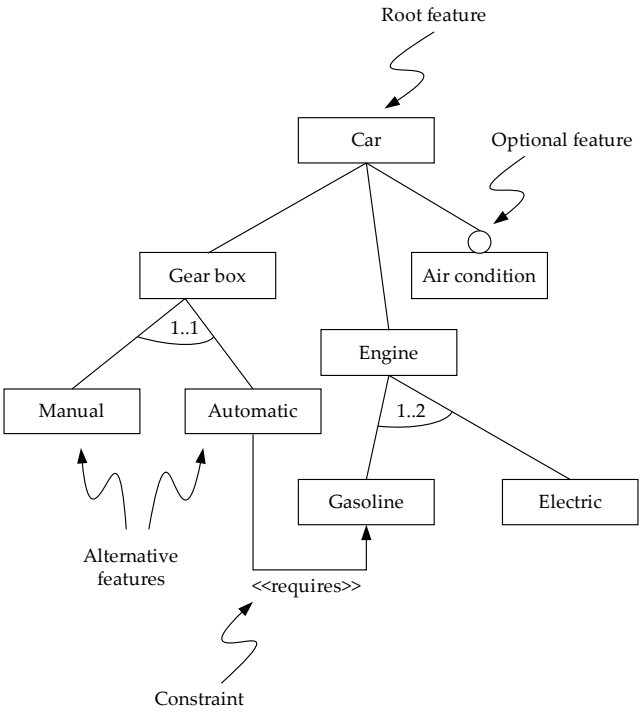

Figure 1: A simple example of a feature model describing a car.

authors and researchers have defined a feature as "a prominent or distinctive and user-visible aspect, quality, or characteristic of a software system or systems" [7], "a logical unit of behavior that is specified by a set of functional and quality requirements" [8] and "a software characteristic specified or implied by requirements documentation" [9].

The features are structured into a hierarchy of features and subfeatures, usually visualized in a feature diagram. The features can have combination qualifiers such as mandatory, optional or alternative. Mandatory features means that if the parent feature is selected for inclusion in the application, then its mandatory subfeatures must also be included. The optional and alternative features show the variability and choices of functionality that the product line offers.

Features can be grouped into feature groups with restrictions and multiplicities attached to the group. Besides the hierarchy relations in the model, there are additional relationships between features in the form of dependencies, mutual exclusions and other feature interactions. Figure 1 shows a commonly used example of a feature diagram.

The benefits of a properly structured and accessible variability model include, for instance, reduced risk of duplicated development efforts, higher communicatability and visibility of current capabilities, and systematic management of dependencies and constraints among development artifacts [3].

However, there are also challenges in feature modeling. Despite the large number of extensions and modifications to the original feature models, there is still no consensus on graphical notations and precise semantics for feature models. All methodologies for feature modeling recognize problems occurring in scaling of the approaches. Since the number of variants and possible combinations of features grows very fast in typical application families [2], overview and consistency management becomes problematic. Tool support is limited and generally the choice stands between unsupported research prototypes or very expensive tool suites for product line development.

Lastly, integrating variability modeling in existing development practices can be a challenge for organizations starting up organized reuse practices. While many methodologies used to propose a big bang-approach to introduction in the organization, changing all practices overnight has proved practically impossible in most cases. Today, the challenge lies in finding the best incremental introduction for a particular organization, and variability modeling is often the first step in such an approach.

3.2 The mobile application domain

As stated previously, the domain of mobile applications is demanding from a variability perspective. Mobile applications are expected to run on a wide range of handsets and the combinations of hardware, operating system, runtime environment and other packages result in a large variation space. The complexity of developing mobile applications has been receiving more attention recently, in part because of the domain to a great extent consisting of small and medium-sized enterprises (SMEs) [11]. Traditionally, SMEs do not utilize the same development processes as large enterprises [12]. Handling the variability in the mobile application domain in a manner that is practical for SMEs is therefore an interesting research area of great interest for practitioners.

Alves et al. [11] hypothesize that companies developing mobile games (a subset of mobile applications) often apply a software product line approach in order to cope with the variability demands. However, the application of the approaches is not explicit and the organizations are not clearly aware that their current practices qualify as a product line, and thus not aware of the opportunities for improving the practices based on other product line experiences.

Mobile applications are increasingly using external data feeds or other communicating services. Examples are streaming media, RSS-feeds, stock tickers, weather and traffic information, maps, etc. This information is acquired from content providers, often via subscription based services that vary in form and terms between carriers and providers. The operating system and target platform also place constraints on

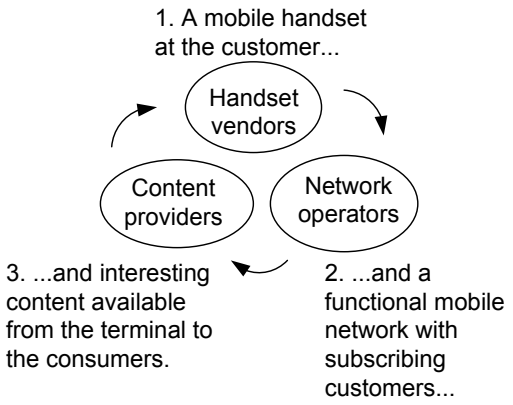

Figure 2: The company's customers and their business drivers.

available programming languages, binary compatibility, usable libraries and user interfaces.

Given the amount of variability in the mobile applications domain, issues of portability, cross-platform development and compatibility need considerable attention. Software product lines present a promising approach to solve or alleviate some of the issues. Since the cell phone industry is already offering their handsets in terms of product families and product lines, and have also made inroads to use SPLs for their device specific software [13], it makes sense for developers of mobile applications to look at software product lines as well.

3.3 The company Tactel AB

This report describes feature modeling experiences made in the mobile application developer Tactel AB, a Swedish SME with several development and sales locations distributed throughout Sweden as well as abroad. Each office has approximately 20-25 employees working on projects ranging from development of mobile television to on-device portals (ODP). Each location typically work with a particular competence such as programming language, operating environment, etc. The products are usually developed for the handsets/clients, while server and content delivery development is made by others. Customers include handset vendors, network operators and content providers, see Figure 2.

As the company grows, it adapts its development processes and methods in order to cope with increasing product complexity. Though being an SME, the company still faces challenging circumstances, such as coordination between development offices in different countries and time zones. The work in this project was carried out at the Huskvarna office.

3.4 Development of current approaches

The business domain that Tactel operates in poses high demands on effective and rational development. Given the requirement for short time to market, reuse is imperative in order to alleviate some of the constraints and problems that the organization faces. As a means to do this, the company has developed an internal framework of reusable components, called plugins. Over time, a large variety of plugins have been developed for various applications and projects. Thus, as hypothesized by [11], we also find the company has already applied advanced reuse and product line concepts without explicitly terming it so.

As the number of developed components that are spread across various projects is becoming too large to overlook using trivial methods (simple lists, spreadsheets), the risk for duplicating efforts grows. If the company would lack efficient and applicable methods to locate and reuse components, they would also miss out on the potential increase in quality that one get from using tried and tested components. Feature models meet the demands for a solution.

3.5 The framework

In order to achieve higher levels of reuse, Tactel has developed a framework of reusable components, called plugins. Plugins are elements of a mobile application principally working like XML tags in a document running on a browser, interpreting the application. Applications are thus written as XML-documents and presented to the user by the runtime system. The platform independent XML runs on top of any of the supported operating systems and handsets, see Figure 3.

A common set of functionality that is used for most of the developed applications and projects is maintained in the core framework. In order to extend the framework with logics, presentation, interface or communication, new plugins are developed within each project. A large variety of plugins have been developed for various applications and projects, including variations of previously existing plugins.

3.6 Model quality

Trying to measure, evaluate or estimate the quality of software products have by now become prevalent in most methodologies and is the subject of lots of research as well as standards and certifications. The matter of quality in models is receiving less attention, despite the fact that software models are often integral and important parts of the early requirements engineering and design activities in a software development project. However, quality work on models has a running start, thanks to the prospect of transferring concepts used on other software artifacts to models.

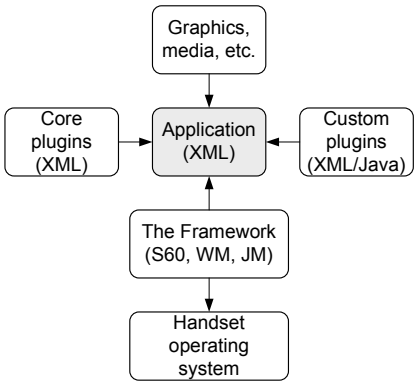

Figure 3: Using the framework is a matter of writing XML using resources, core plugins and custom plugins. The framework interprets the XML and runs the application on the target platform.

Metrics for determining qualitative properties have been used to measure and compare usability and understandability of models and modeling languages. Metrics for models have mostly been looked at for conceptual data models [14] and UML-models [15]. Devising metrics for variability on model level has begun to receive some attention, such as a basic complexity measure [16].

A quality represents an important aspect that can emerge in a feature model. Quality models, methods for quality evaluation and attributes or metrics affecting external quality have been published, both regarding software in general [17], product lines [18] and software related models [14, 15, 19]. The selection, description and rationale used for the six qualities used in our case study are partly based on the experiences from previous work in quality attribute determination, a selection based on what qualities that are transferrable to feature models, and a consensus among the stakeholder and modeling leaders on what the most important emergent properties of a feature model are (addressing research question Q1 from Section 1). We define here the quality factors that were used in the prioritization and evaluation of the feature model.

3.6.1 Changeability

is the ability of the feature model to evolve as the modeled product line evolves, while maintaining the same purposes and applications as previous iterations of the feature model. A feature model describing a product line that is expected to change rapidly or regularly must have the flexibility to accommodate those changes without requiring too extensive efforts for re-design.

3.6.2 Re-usability

encompasses the properties of the feature model necessary to enable or facilitate the reuse of the current feature model when evolving or developing other models. The level of re-use can be on different levels of granularity such as entire feature model or partitions of it.

3.6.3 Formalness

is related to the ability to manage the feature model in a formalized manner, notably for the purpose of machine management, tool support and automation. Conformance to notations, redundancy and consistency are examples that may affect the level of formalness in a feature model.

3.6.4 Mobility

concerns the feature model's ability to be installed and made available to the users, integrated with tools, interact with existing processes, moved or duplicated to different systems or tools.

3.6.5 Correctness

is the quality of the feature model's veracity and appropriateness for the modeled scope. The accuracy in modeling the real artifacts, the level of completeness and coverage of the modeled domain and the reliability of the usage of the feature model resulting in accurate solutions are aspects of correctness. While having a correct feature model would seem to be a top priority, it could also mean that the learnability, re-usability or changeability are negatively affected.

3.6.6 Usability

is the feature model user's ability to employ usage and receive beneficial effects of the feature model. A usable feature model should be easy to learn, accepted by the stakeholders as a suitable representation, accessible and visible to the users, and understandable for the stakeholders. In order to reach high usability one might have to sacrifice correctness and formalness.

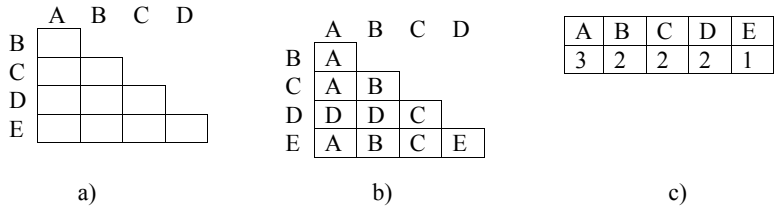

Figure 4: Comparison matrix, filled comparison matrix and ranking using pairwise comparison.

3.7 Pairwise comparison

In this study we use pairwise comparison for determining a ranking of important and prominent qualities in a feature model. Pairwise comparison is a common method of comparing choices and arriving at a preference. It is a method that is easy and fast to administer and provides a straightforward way to gather the collective ranking of a group of developers or other stakeholders.

Given a set of options or items that are to be compared, in our case qualities, a matrix is constructed similar to the example in Figure 4a. Each stakeholder in a group fills out the matrix by selecting which item in the grid that it considers the more important (Figure 4b). Finally a summation is made of how many times each option has been selected among the stakeholders and a ranking order is found (Figure 4c).

A common element of quality assurance for software products is goal selection, where the stakeholders of the product get to negotiate and select which non-functional requirements on the product that are regarded as the most important to accommodate. The development of the product is then directed and governed by the goal to ensure that the result meets the quality demands posed.

We first use pairwise comparison to get a ranking of what qualities that the stakeholders of a feature model regard as the most important. We then use pairwise comparison for evaluation of the finished feature model where a stakeholder is presented with the complete feature model and is asked to use a comparison matrix to appreciate which quality that prevails in the model. We detail how the study was conducted in the next section.

4 Methodology for Feature Modeling

The organization did not have any previous experience with feature modeling. In this section we describe the rationale and reasoning of the scope selected for the feature model, the training and modeling activities. Out of the many guidelines published concerning domain analysis and domain modeling [4, 20, 21] most are far too

extensive for an SME to apply. Others are very specific in the sense that they pay particular attention to a special aspect of the modeling. In the first methods prescribing a product line approach there were emphasis on introducing the product line mindset organization-wide in an almost revolutionary manner. More recent descriptions and recommendations acknowledge that it is difficult to make such stirring changes from a cold start. Variability inventory is commonly the first step.

The general approach to feature modeling could be abstracted to:

- collecting information sources,
- identifying features,
- abstracting and classifying features into a model,
- defining the features, and
- validating the model.

In preparation of the modeling activities, a suggested workflow for eliciting features was constructed, showed below. This methodology was refined and specialized to fit the circumstances in this project. Section 6 has more details on the activities carried out and the resulting artefacts.

ITERATE OVER

- Elicitation of terms and features
- Composition and structuring of features
- Refinement and re-modeling
- Consistency checking

DEFINITION OF MODEL BOUNDRY

- Name systems inside and outside boundry
- Formulate modeling completion criterion

DEFINITION OF DICTIONARY

- Define a domain vocabulary containing non-trivial words

LIST FOR PRODUCTS IN THE SCOPE...

- Customer/system specific features
- Additions, subtractions or changes due to product evolution
- Non-functional attributes or qualities
- Product specific development aspects

…FROM ANY OF THE CATEGORIES…

- Entities
- Events
- Operations
- Scenarios
- Interfaces
- Visible functions
- Domain specific technology
- Theory, algorithm, data
- Implementation techniques
- COTS
- Existing components

…AND RECORD THESE PROPERTIES FOR EACH FEATURE

- Name
- Source
- Short description
- Rationale
- Stakeholders
- Exemplar systems
- Constraints, ranges
- Availability and binding time

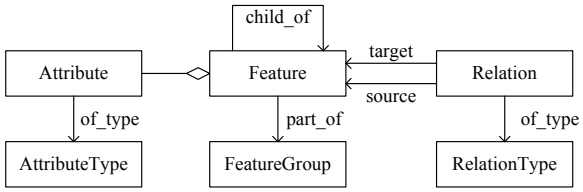

Figure 5: Informal illustration of the meta model used for the tool FMWIKI.

- Priority
- Top/lower-level domains
- Input/output, needs/provides service

DESCRIBE RELATIONS BETWEEN FEATURES

- For example include, exclude, depend, produce, consume, etc.

ABSTRACT FEATURES

- Specialization, generalization
- Classification, partitioning, logical vs. physical boundry

4.1 The meta-model for feature models

The meta-model used in the modeling project is depicted in Figure 5. It allows the modelers to define attributes and relations to be attached to features in a very flexible manner and is easily implemented in a relational database. The default feature hierarchy is tree-formed, and other relations are solved by adding further relationship types and relations. The default relationship in our models' hierarchical structure is optionality.

A feature is part of a feature hierarchy, where each feature has a parent feature. The model does not allow for more than one parent per feature, but pragmatically there is no need to accommodate such a construct. Any such needs can be solved using suitable dependency relations, while the ensuing tree structure of the current meta-model is easy to understand. Considerable effort has been spent in feature modeling research trying to determine what types of relations that should exist in feature models, and also what additional information about a feature that is meaningful to store. The meta-model used in our work allow the modelers to create any type of attribute describing a relevant property with a feature, while not presenting a fixed set of possible relations. Since there are few publically presented cases in variability

modeling where the situation of the modeling organization has been fully accommodated using the same setups as previous projects, it makes more sense to avoid imposing restrictions.

A feature in our case can come in various types. Using terminology used by other authors, the features can be

- Abstract, indicating a concept that must be realized by a subfeature in the form of a plugin. Principally, this is a categorization.
- Concrete, which in our case means a plugin.
- Collection, that acts as a container for a feature group.
- Parameterization, which in our case almost always is realized as a parameter to a plugin.

5 Tool for Feature Modeling

This section describes the requirements for a tool designed to accommodate the model produced in the project. While there are a few options for commercial tools that could have been used in the project, it was decided that a different tool was needed. This section first discuss the particular circumstances that Tactel face and then formulate the requirements on the tool and the implementation of FMWIKI.

5.1 Scaling of modeling and visualization

Almost all authors and methodologies in the field of variability management and analysis recognize the problem of scaling modeling and visualization of variability [22]. When the complexity of a product line gives rise to many interdependencies and relations between features, and when the number of features in the model increases, most notations for visualizing models break down and become very hard to use. Keeping track of all the data and the changes that go into the model becomes impractical using trivial methods and approaches, like text documents, schematic drawings, list, etc. Some form of dedicated tool support becomes a necessity in order to handle the large amounts of information that go into the models.

Most of the early tools for feature modeling as well as most of the research oriented prototypes and demonstration tools that are available, try to use a modeling interface that emulates the use of modeling and notations that were used in the initial efforts of feature modeling. This means that a workbench or workspace surface is used to model, edit, rearrange and compose the features, relationships and dependencies. This approach quickly results in the same incomprehensible and tangled

structures as had one used pen and report. More modern commercial tools use more effective approaches to structuring the feature models, mostly using tree structures and outlines.

Since there is no standard, consensus, or de facto notation used for variability models the matter of unclear visualizations is relatively easily resolved by reducing the number of visual modeling constructs used. Many approaches to variability modeling suggest that one should not illustrate the interdependencies between different parts of the feature model in the main view of the visualization, but instead use a table or separate list for such information. This highlights the difficulty of communicating the variability of core assets effectively, trying to avoid overwhelming information while still capturing the essential aspects of the variability.

However, it is not only the visualization that poses problems to the up-scaling of variability modeling. Many of the examples in texts about feature modeling use constructed examples that are not very large and could probably be managed in a rather simple manner. In practice, the variability models contain many more features and have more information tied to them than what is visible in the overviews of feature diagrams. Management and maintenance of the models become as hard as it is vital, if the tools and methods used are not equipped for continuous modeling and updating.

5.2 Distribution of modeling and information

As previously mentioned, Tactel has development and sales offices in several geographical locations. Development of reusable assets is conducted in different sites and in several projects and the assets are used by different projects, meaning that a model of the assets needs to be constructed and updated by, and communicated to, several locations and staff in many different roles.

Software engineering tools with support for distributed and cooperative design and development have become more common in recent years, although the large tool suites usually only have support for team-based development in the high-end versions. When it comes to supporting teams separated in location and time zone, versioning, version merging and differencing, and concurrency become important aspects of the development approach [23].

Aside from the modeling and updating of the feature model by staff in different locations, there is the matter of distributing the model to the end-users. Collecting changes and updates and making them available in a timely, consistent and convenient manner pose demands that can be hard to meet using desktop applications.

Grundy and Hoskins [23] review several aspects and mechanisms for collaboration support in software development, and separate the tools to general-purpose tools for communication or document management and special purpose-built tools for the

software development activities. The advantage of general-purpose tools, such as shared repositories, conventional e-mail, etc. is that they are readily available and tested. However, they may lack useful or essential features for developers.

5.3 Tool requirements

Gray et al. [24] bring up a list of considerations for constructing software engineering tools including (i) host platform and implementation language, (ii) standards conformance and reference models, (iii) user interface, and (iv) mechanisms for repository, integration and interoperability.

With regard to (i), the environment at the company is quite agnostic with regard to supported platforms and languages. However, a preference was made towards open and portable solutions. This is related to the context that the feature model would have to be possible to access and edit in diverse locations and by various people working in different environments.

As for (ii) there are, as previously mentioned, no standards or conventions to adhere to when considering the tools. All variability modeling tools have inherent limits in the types of models that they can contain, since the tool is built to support some type of meta-model for variability modeling. The meta-model dictate the constructs that are available for the modeler, for example restricting the model to a tree structure. In the next section the scoping activity will be further described, during which the demands on what constructs that the feature model had to support were elicited. The implemented meta-model is described below. The general preferences were that the meta-model should be relatively extensible so that the features could be annotated with additional information about them with ease, and so that different forms of relationships and dependencies could be expressed. Aside from that, the demands on particular modeling constructs were quite basic and could be accommodated by most methodologies and approaches available in literature and tools.

Aspects (iii) and (iv) are related to each other since the user interface is the view into the repository where the model is stored. Again, the need for scaling and the distributed nature of development and dissemination of the model sets the main requirements. Obviously the choice of constructs to be available for the modelers is also influencing the requirements for the repository. When considering the usage scenarios for the feature model and the information that would be contained in it, the preferences for user interfaces and storing of the model leaned towards a server-client solution with an uncluttered and – to the greatest possible extent – simple user interface to a browsable, searchable feature model in a central location.

Rech et al. [25] have suggested that wikis could be a suitable basis for reuse-oriented tools. The benefits they found in their application of a purpose-built wiki in development projects, of interest in the context of this report, include the use of a

single point for storage and retrieval of information and a low usage threshold. The main drawback found was the possibility of opening up for generally low quality documentation due to incompleteness and inconsistencies.

The collaborative aspects of wiki-like tools have been brought up in several other works. Ferreira and da Silva [26] describe a wiki designed for facilitating requirements engineering. Romberg [27] describes a wiki-like desktop tool for integrating data sources relevant for software engineering projects. Xiao et al. [28] propose using wikis for collaborative, remote software development including coding, compiling and debugging.

5.4 Implementation

A purpose-built wiki-style modeling tool would fulfill several of the requirements posed on a tool for feature modeling in Tactel. It would be easily distributed via a web interface and support concurrent modeling and immediate distribution of changes to all users. Thusly, a web-based tool for feature modeling (called FMWIKI) was implemented based on a flexible meta-model and a practical notation decided on during the scoping activity.

The meta-model is implemented in an SQLite database, and the web interface is made in PHP. While the meta-model, and thus the repository database, is very pragmatic and flexible, the web interface is tailored to correctly interpret the information about features, associated attributes and relations to support the aims of the organizations.

The graphical notation is controlled with CSS-templates and can be changed to match other preferences. FMWIKI supports user administration, model administration, defining attribute types and relationship types, browsing the feature tree and relationships, adding, editing and removing elements of the feature model, filtering views and full text searching in the model.

The implementation consists of 23 PHP-scripts, CSS and graphics, some Javascript, and an SQLite database for users of the system. Each feature model project is stored in a separate SQLite database. At the time of this writing, the implementation is feature complete and functional, but there are many ideas for improvement. The agreement between Tactel and JTH assures that the tool is fully accessible by both parties.

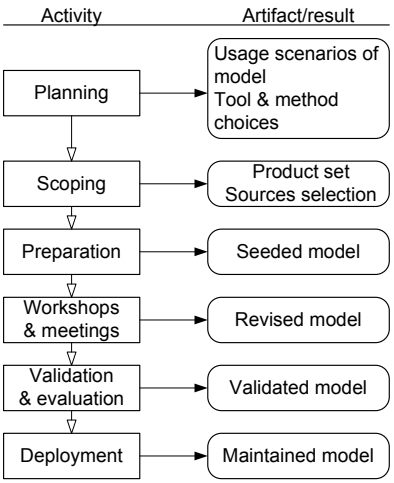

Figure 6: The activities and artifacts of the feature modeling effort.

6 Modeling Activites

Figure 6 shows the general order of activities carried out and artifacts produced during the feature modeling project. This section describes four aspects or activities of the feature modeling efforts carried out and the impact and results they led to.

6.1 Scoping

The scoping activity aims at finding the boundary of the model and setting the goals for the outcome of the modeling activities. The input used to decide on scope usually includes the organization's business and product strategy, technical suitability, access to domain expertise and sources, etc. Ideally, the scope of the model should end up encompassing enough to provide the most value for the organization, but still be manageable and prone to be realized. The recommendations found in various guidelines and methodologies are to pick a scope where the included parts are well understood, relatively small and not subject to pressuring schedules which would mean that the analysis would be rushed.

In the case reported in this report, the selection of the framework and the various plugins as subject of feature modeling was chosen since the plugins generally represent components suitable for reuse and are often made in different variants providing somewhat different functionality. Having a proper and common structure and overview of the plugins would benefit the possibilities to reuse previous efforts, reduce the risk of duplicating functionality in different locations and projects, and benefit a large number of people in the organization. It also had the additional benefit of being easy to formulate in measurable progression, i.e. keeping track of how many plugins and projects that had been included in the feature model.

Practically, the selection of the plugins to be analysed for commonality and variability, and be included in the model, was based on the documentation level that existed for the plugins. While, obviously, it is desired that all plugins and derivations that are produced are also documented, time pressure and available developer resources result in varying levels of elaborate documentation. For the initial round of analysis and modeling, four projects that produced plugins and variants of plugins were chosen. Three of the projects were managed in the same company office, which also made it easier to gather expertise concerning the components.

6.2 Training

Since the project was carried out in cooperation with staff from academia with experience and expertise in feature modeling, the training of the modelers was mainly carried out during the workshops based on various presentation material. Feature elicitation techniques could be kept rather straight-forward, thanks to the well-defined scope of the feature model. The modeling notation was introduced using generic examples, based on the capability of the tool used for the modeling.

6.3 Workshops

Initial meetings to decide on the scope and size of the feature model were held on a project manager level, gradually including more staff from the organization as the scope was narrowed down and the necessary expertise was needed. The requirements mentioned in Section 5 were mainly formed during these initial meetings, based on the managerial view of what the model scope and deployment should be.

Once the scope was determined and the participant projects were selected, workshops were arranged. An initial workshop introduced the project and initiated the modeling, gathering almost all the team members from the different projects. While the projects were selected based on the adequacy of the documentation available, practically all developers in the projects had been part of producing the plugins, so each one had unique expertise to bring to the table when it came to analyzing the plugins in terms of features that they provide. Before commencing with the modeling sessions, each stakeholder filled out a pairwise comparison of the six quality factors. This resulted in a ranking of what the stakeholder group saw as the most important qualities in the eventual feature model, see Section 7.

Initial analysis and modeling of the assets was made using pen and report, in order to get the developers thinking in terms of features and quickly get started on the modeling notations. After the first workshop, a basic structure of the assets had emerged and could be used as a basis for further modeling. The assets originating from the projects were then added to the model, based on the general structure.

Once all the plugins from the projects had been added to the model, a series of smaller workshops were held in order to validate and evaluate the resulting model.

After the resulting feature model was validated, the stakeholder group was asked to do an evaluation of the model using pairwise comparison. For each comparison, the stakeholders were asked which quality that they believed was more prominent in the feature model. The result was thus a ranking of what qualities that the resulting model had, for comparison between the desired qualities and the resulting qualities (addressing Q2).

While doing the evaluation of the resulting model, the stakeholders were asked to indicate what properties, characteristics, or traits of the feature model that led them to believe that a quality took precedence over another (addressing Q3). Composing and comparing those indicators resulted in a matrix of what properties of a feature model that indicate the qualities of it. Relating this information to how the indicators emerge in the model leads to the potential of guiding the modeling activities towards achieving the prioritized qualities (addressing Q4).

6.4 Seeding

In preparation of the first workshop the modeling leaders had reviewed the documentation for one of the participant projects. This initial review resulted in each of the existing plugins being assigned a number of single-word tags, describing characteristics, main functionality, or variant functionality of the asset. The tags were then used to seed the model with a basic structuring, as several plugins were identified having similar functionality or being variants of similar functionality.

Undoubtly, this initial analysis of one project, made by the project leaders, affected the rest of the analysis by the participating projects. Although the model was entirely open for changes in the workshop stage, most of the analysis and structure from this first seeding remained through the rest of the modeling. The question is if this had a positive or negative effect on the overall resulting model. While a positive effect was a running start for the first workshop, it could also lead the developers into tracks that were inappropriate for the domain and scope.

In the end (as also noted in Section 8) the end-users of the model were satisfied with the structure and content of the model, so we would still recommend this practice, provided that it is performed by persons with adequate understanding of both variability modeling and the domain scope.

Table 1: Some properties of the resulting model.

Property	Value
Number of features	305
Number of top-level features	9
Maximum depth of hierarchy	5
Number of variation points	62
Number of relations	11
Number of relation types	3

7 Modeling Results

For comparison with other feature models, some numbers describing the general properties of the resulting model can be seen in Table 1. The validated model contains 305 features, distributed over 9 top-level features and 62 variation points, and has a maximum depth in the hierarchy of 5 levels. The model is thus quite shallow with many variation points. Intuitively, this corresponds to a structure of a model with fewer common features and more focus on the variability. The model in this case does in fact have few mandatory features, but many optional features and feature groups.

Once all projects had added assets and variants to the feature model, each project validated the feature model together with the modeling leaders, clarifying modeling choices and loose ends. Each feature was annotated with a description of the main functionality in the feature, the origin of the feature for tracing which project it was developed in, restrictions, and external dependencies.

During the modeling sessions and the validation sessions, several benefits of having a variability model became apparent. It turned out that different projects had developed assets with the same functionality, unaware of the existence of plugins with the desired functionality. There were also several relations and dependencies among the assets that were not previously explicit.

8 Validation and Evaluation

The model was also evaluated by the participants, in order to see if there were any major qualitative arguments against the current state of the model. The participants were satisfied with the structure and found it to adequately describe the current state of the scope, be usable for the purposes intended and contained appropriate information. However, a proper evaluation of the model will require that it is put into use for some time.

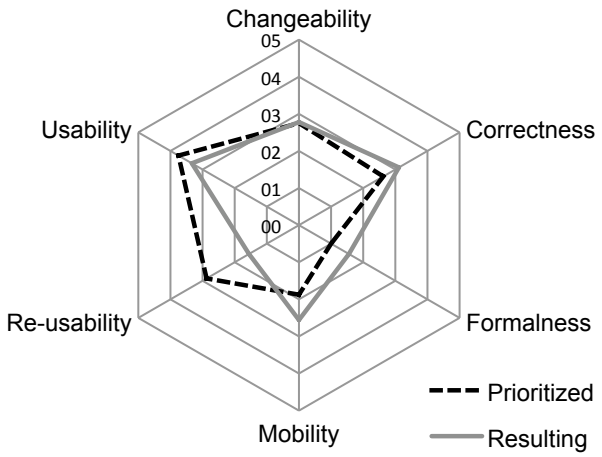

Figure 7: Radar graph comparing the qualities that the stakeholders prioritize in a feature model, and the evaluation of the resulting feature model.

Table 2: The average number of tokens given each quality during prioritization and evaluation.

	Desired quality	Resulting quality
Changeability	2,8	2,8
Correctness	2,6	3,1
Formalness	1,0	1,6
Mobility	1,9	2,6
Re-usability	2,9	1,6
Usability	3,8	3,3

The ranking of the qualities that the stakeholders saw as the most relevant in a model, and the evaluation of the resulting model is shown in Figure 7. The rank value is calculated as the total number of tokens placed on the quality divided by number of stakeholders, giving the average number of times the stakeholders chose the quality in the pair-wise comparison, see Table 2.

Usability is clearly the quality that is most desired with re-usability, changeability and correctness trailing close together. The highest ranked qualities that were found in the resulting model are usability and correctness. The largest difference between the desired and achieved quality is found in re-usability where the produced model is not reaching the preferred level.

Generally, the prioritized qualities are higher on ease of use and re-use at the expense of formalness and correctness. Most of the users are thus more interested in having a model that is oriented to practical and pragmatic qualities, rather than technically oriented, formal aspects. The resulting model, however, exhibit more of

correctness and formal adherence. It is important to note that the evaluation was made on the model in itself, but that there could be confounding factors affecting the outcome, such as the user-friendliness of tools, etc.

8.1 Quality indicators

In the evaluation of the resulting feature model we see five principal indicators for how the quality factors affect the model.

Supported modeling elements Several qualities of the model are affected by the meta-model used and thusly what modeling elements and constructs that are available. As stated previously, the meta-model for the tool is simple and straight-forward, but flexible. Even though the model could contain a variety of feature attributes and relationship types, it turned out to be quite constrained in this respect. The resulting model thus has few constructs to learn and use, which improves usability. It also positively affects formalness, since fewer modeling elements are easier to adhere to.

Placement of features in hierarchies The placement of the features in the hierarchies has a positive impact on changeability and re-usability, but negative effect on correctness, in the sense that more freedom of choice on where to place the features results in easier changes and updates of the model. However, the placement of the features is sometimes difficult and the solution that results in the more comprehendible model, might to a lesser extent reflect the circumstances in the modeled product line. During the validation activity of this case, there were several features that could be placed in different parts of the model, since the features had cross-cutting functionality.

Structure and amount of dependencies The evaluation shows that having few dependencies in the model is beneficial for changeability and re-usability. As could be expected, few dependencies in the model makes it easier to modify, and avoiding complex entangling of features via dependencies improves the re-usability. The difference between dependencies and relationships is important in this case. There could be soft relations between features that are not dependencies, but of a more informative character. Dependencies are a hard type of relations, where the structure of the dependencies affects how many features that are involved in each change.

Obviously, the structure and nature of the dependencies in a model are closely related to where the features are placed, and in turn, how the modeled product line is structured. However, in our experience during the modeling activities, one can experiment with trade-offs between feature placement and dependency structure, meaning that it can very well be possible to affect the structure and amount of dependencies in the model, and thus the qualities that emerge as a result.

Amount of supplemental information A feature could have any type and amount of information attached to a feature or relation in the model [21]. In the case de-

scribed in this report, the supplemental information was kept brief, with only half a dozen attributes for each feature. Restricting the amount of supplemental information to the essentials was found to be positive for the changeability, since it reduces the effort needed for making changes and adding new features to the model. Limited amounts of information was on the other hand also found to be negative for the correctness of the model, since it made it harder to convincingly determine whether the model was properly structured when first encountering it.

Overview and correspondence to actual artifacts The resulting model is quite shallow in that it does not have very deep hierarchical structures. This was found to provide easy overview and improve the usability of the model. It also contributed to changeability since it was easy to find the appropriate place in the model to enter new features. Additionally, related to the placement of features in the model, the features were found to correspond well to the structure of the actual artifacts and assets. This was considered to contribute to the mobility of the model, since the model could be used for integration with other systems, models or tool.

8.2 Discussion and interpretation of results

There are several factors that could lead to the results found in the evaluation, that are not related to the indicators mentioned. Among others the tool, the material used for the modeling, and the context in the company could influence the results. Obviously, the only way to identify and attribute appropriate weight to those factors is to keep applying the methods in further cases.

There is also a more subtle factor as to why the results of the evaluation lean towards the formal aspects of a model. It could be the case that the feature modeling method and the style of the models that results from feature modeling, inherently emphasizes the qualities related to correctness and formality, while the expectation of the users of the feature models is to have a more pragmatic model of the components. While generative approaches to feature modeling are pursuing increased formalism, there appears to be room for research into the pragmatic side of modeling as well.

The indicators that were identified during the evaluation are quite intuitive and affect the qualities in relatively predictable ways. However, it is interesting to note that we have found all of the indicators found to be adjustable and thus the quality of the model is possible to influence towards the priorities made.

9 Deployment and Integration

Although the requirements for the tools and integration have already been touched upon in previous sections, we provide some more details regarding the disseminat-

ing of the model to the organization and the possibilities for integration with other practices and systems already in use.

9.1 Deploying the model in the organization

Making the model available to the end-users is technically a matter of providing the URL to the wiki and add the user to the wiki user list. This is of course a very convenient solution to the requirement of providing the tool across different locations and countries. It also means that installation problems are kept to a minimum. The wiki tool is currently tuned towards the web browsers most commonly used, but is relatively stable across platforms.

Practically, however, the developers that have not been involved in the modeling efforts so far, will need to be informed of the purpose and availability of the model and have training material available. The modeling wiki itself does not currently provide training material or tutorials, although it is easy to add a model to use as sandbox.

Since the model is already populated by plenty of features and have a structure in place, it is relatively self-explanatory. A new user that needs to interact with the model can fairly easily discern from existing material, how to add new features. The level of training needed for new users is therefore not much more complex than a brief presentation of the tool and some guidelines for how to use it. All of that is contained in a single document that is distributed to new users.

9.2 Integrating the model in practice

The intention of the model is to assist developers in finding suitable, reusable assets. Obviously, this is important in the beginning of the development lifecycle. Software product line methodologies emphasize the strategic use of variability models to determine current capabilities and potential new products and market possibilities. Aside from use in the initial product and project planning activities for finding the current capabilities, the model is also intended to serve as a coordination point for finding reusable assets throughout a development project.

The feature model also serves as the minimum level of documentation for new development assets produced in the organization. This means that when new plugins, extensions or derivations are developed, they are to be added to the model. Although different projects may require different levels of elaboration for the documentation, the information contained in the feature model now represents the common denominator for all new assets.

This in turn means that further development and maintenance of the model is very much in the hands of the developers themselves. Getting the model into the normal

development workflow of the developers is not much more complicated than having a bookmark in the web browser and search the model when considering the required functionality for the application. Integrating the model in the documentation workflow might prove a bit more bothersome, although the model is made as accessible as possible to not discourage keeping it updated.

The wiki tool was designed to not be integrated with other development systems, such as version and configuration management tools, source code repositories, etc. There are thus no integrated links from features to other systems. This decision was made based on the existing structure of binaries and source codes, as well as the existing development policies, making it sufficiently straight-forward to find the assets from the feature model in the other systems, without explicit integration.

10 Conclusions

During the workshops, it became apparent that a variability model was a suitable addition to the development processes in the organization. Different projects had independently developed similar functionality and derivations, unbeknownst of each others' efforts. The acceptance and utility of a variability model, based on feature models was thus high and provides concrete applicability in the organization. Using a web based tool reduces the efforts needed for installation and integration, and has collected previously unclear and dispersed information in a common format, increasing the communicability of existing assets.

The possibility of using a wiki-like environment offers interesting possibilities for further development. Since the available capabilities are now up-front and visual, the tool could encourage and facilitate discussion around the assets. By keeping track of the most common searches, the most central functionality could be found. Wishlists and rankings could be used to identify gaps in the capabilities and weaknesses in the current set of assets. The advantage is that many users are already used to these sort of functions from other online services, reducing the learning threshold for new functionality.

We are not able to give a definitive list of guidelines and indicators for qualities in feature models at this point, but we have demonstrated the potential of using pairwise comparisons of qualities in feature models for prioritizing and evaluating models, and found support for the qualities and indicators detailed in this report.

11 Summary

As indicated by for example [11], SMEs are often applying software product line concepts, especially those working in domain close to embedded systems where

product families and product lines are commonplace. Introducing suitable mechanisms and workflows in SMEs can lead to rather immediate benefits as they allow higher utilization of the efforts that are already in place.

Because of the limited resources and limited time available in an SME for revising development procedures, it is imperative that the efforts spent are targeted to reach the results that are most suitable for the context. Variability modeling is a useful way for leveraging the existing resources in an organization, and relatively simple and common techniques can be used to reach higher quality and usability in variability models.

The results of this project meet the expectations of Tactel and JTH. For Tactel, there is a feature model covering the plugins for the framework, described in a tool and deployed in the organization. For JTH, empirical evidence has been gathered to support the pairwise comparison approach to prioritizing and evaluating the quality of feature models.

References

[1] Clements, P., Northrop, L.: Software Product Lines: Practices and patterns. Addison-Wesley (2002)

[2] Sinnema, M., Deelstra, S.: Classifying variability modeling techniques. Inf. Softw. Technol. **49**(7) (2007) 717–739

[3] Fey, D., Fajta, R., Boros, A.: Feature modeling: A meta-model to enhance usability and usefulness. In: SPLC2. (2002)

[4] Chastek, G., Donohoe, P., Kang, K.C., Thiel, S.: Product Line Analysis: A Practical Introduction. Technical Report CMU/SEI-2001-TR-001, Carnegie-Mellon University (2001)

[5] Simos, M., Creps, D., Klingler, C., Levine, L., Allemang, D.: STARS Organization Domain Modeling (ODM) Guidebook. Technical Report STARS-VC-A025/001/00, Lockheed Martin Tactical Defense Systems (1996)

[6] Zelkowitz, M.V., Wallace, D.R., Binkley, D.W.: Culture conflicts in software engineering technology transfer. In: NASA Goddard Software Engineering Workshop. (1998)

[7] K.Kang, S.G.Cohen, J.A.Hess, W.E.Novak, S.A.Peterson: Feature-Oriented Domain Analysis (FODA) - Feasibility Study. Technical Report CMU/SEI-90-TR-21, Carnegie-Mellon University (1990)

[8] Bosch, J.: Design and Use of Software Architectures. Addison Wesley (2000)

[9] IEEE Standards Board: IEEE Standard Glossary of Software Engineering Terminology. Technical Report lEEE Std 610.121990, IEEE (1990)

[10] Griss, M.L., Favaro, J., d'Alessandro, M.: Integrated feature modeling with the RSEB. In: ICSR'98: Proceedings of the 5th International Conference on Software Reuse. (1998) 76–85

[11] Alves, V., Câmara, T., Alves, C.: Experiences with mobile games product line development at meantime. In: SPLC '08. (2008) 287–296

[12] von Wangenheim, C.G., Weber, S., Hauck, J.C.R., Trentin, G.: Experiences on establishing software processes in small companies. Information and Software Technology **48**(9) (2006) 890 – 900 Special Issue Section: Distributed Software Development.

[13] Maccari, A., Heie, A.: Managing infinite variability in mobile terminal software: Research articles. Softw. Pract. Exper. **35**(6) (2005) 513–537

[14] Genero, M., Poels, G., Piattini, M.: Defining and validating measures for conceptual data model quality. In: CAiSE. (2002)

[15] Lange, C.: Assessing and Improving the Quality of Modeling: A Series of Emperical Studies about the UML. PhD thesis, Technische Universiteit Eindhoven (2007)

[16] Lopez-Herrejon, R.E., Trujillo, S.: How complex is my product line? The case for variation point metrics. In: VaMoS. (2008)

[17] International Organization for Standardization: Software engineering — product quality. International standard; ISO 9126 ISO/IEC 9126-1/2/3:2001 (2001)

[18] Etxeberria, L., Mendieta, G.S.: Quality assessment in software product lines. In: ICSR. (2008) 178–181

[19] Moody, D.L.: Theoretical and practical issues in evaluating the quality of conceptual models: current state and future directions. Data Knowl. Eng. **55**(3) (2005) 243–276

[20] Lee, K., Kang, K.C., Lee, J.: Concepts and Guidelines of Feature Modeling for Product Line Software Engineering. In: Proceedings of The Seventh Reuse Conference. (2002)

[21] Czarnecki, K., Eisenecker, U.: Generative Programming. Addison Wesley (2000)

[22] Djebbi, O., Salinesi, C.: Criteria for comparing requirements variability modeling notations for product lines. In: CERE'06: Fourth International Workshop on Comparative Evaluation in Requirements Engineering. (Sept. 2006) 20–35

[23] Grundy, J., Hosking, J.: Software tools. In: Encyclopedia of Software Engineering. John Wiley & Sons (2002)

[24] Gray, J.P., Liu, A., Scott, L.: Special issue on constructing software engineering tools. Information and Software Technology **42**(2) (2000) 71 – 72

[25] Rech, J., Bogner, C., Haas, V.: Using wikis to tackle reuse in software projects. IEEE Software **24**(6) (2007) 99–104

[26] Ferreira, D., da Silva, A.R.: Wiki supported collaborative requirements engineering. In: Wikis4SE: Third Workshop on Wikis for Software Engineering. (2008)

[27] Romberg, T.: Wiquila - a wiki rich client that mixes well with other sources of software project information. In: Wikis4SE: Third Workshop on Wikis for Software Engineering. (2008)

[28] Xiao, W., Chi, C., Yang, M.: On-line collaborative software development via wiki. In: WikiSym '07: Proceedings of the 2007 international symposium on Wikis, New York, NY, USA, ACM (2007) 177–183