Variability Modelling Throughout the Product Line Lifecycle

Christa Schwanninger[†], Iris Groher[‡], Christoph Elsner[†], Martin Lehofer*

[†]Siemens Corporate Technology Erlangen,
[‡]Johannes Kepler University Linz, *Siemens VAI Linz
{Christa.Schwanninger, Christoph Elsner.ext, Martin Lehofer.ext}@siemens.com, Iris.Groher@jku.at

Abstract. This paper summarizes our experience with introducing feature modelling into several product lines within Siemens. Feature models are used for solving various tasks in the product line lifecycle, starting with scoping the reusable asset base up to support for actual product configuration. Using feature models as primary artefacts for managing variability early in the lifecycle, we could improve the efficiency and transparency of scoping activities considerably and made the development efforts way easier to schedule. On the other end of the lifecycle, feature models lowered the engineering efforts in solution business in supporting product configuration and instantiation.

1 Introduction

Product line engineering [1, 2] denotes a collection of engineering techniques supporting the efficient reuse of a common set of core assets when developing similar products. There are three main measures to achieve this reuse: proper scoping of the domain and deriving platform scoping decisions from business considerations, managing variability, and building up a reuse culture. Siemens business groups have a lot of domain knowledge and many success stories to tell; nevertheless staying competitive requires constant improvement and a product line approach is very promising to decrease time-to-market for those business groups developing similar or successive products in the same domain.

Feature modelling [3] was introduced as part of the domain analysis and domain modelling phase to systematically describe the common and variable features shared among the products of a product line. We found that feature modelling supports several areas of product line engineering very well, especially *scoping* [4] and the *configuration and derivation* [5] of products from the reuse infrastructure, but also activities like project planning and tracking, testing and customer negotiations.

We introduced feature modelling as a concept together with appropriate tool support in several business groups within Siemens, mainly to support either scoping and project planning or (partly) automatic product configuration and derivation. In this experience paper, we will describe the introduction processes in two business groups together with the improvements achieved, and the lessons learned.

Originally published in the Proceedings of the ACM/IEEE 12th International Conference on Model Driven Engineering Languages and Systems. Vol. 5795. Berlin/Heidelberg, Germany, Springer, 2009. ISBN 978-3-642-04424-3. The original publication is available at http://www.springerlink.com.

2 Experiences with Feature Models for Scoping

The first group we report on comprises one platform unit developing reusable core assets and several application engineering units. This distribution of responsibility requires considerable effort to communicate the platform scope and support for transparent tracking of asset development. Application units add features to each product and want to know exactly what the platform will deliver when. Feature modelling supported all steps for setting up a product line approach described subsequently.

2.1 Structure the requirements and build up a domain vocabulary

Why? The business group had structured their requirements mainly in use cases before. While this made the requirements easily understandable, it was hard to determine if they were complete and what the commonalities and variations were in the platform. Moreover, many of the 5000 requirements were not included in the use case descriptions because this was not really feasible for some parts of the overall domain, e.g. UI frameworks or frameworks for data management.

How? The feature model was built in a top-down and a bottom-up manner. Top down a couple of sub-domains were identified with two of them being workflow-driven. These workflows are kind of standardized, so they can easily be utilized to check for the complete coverage of these sub-domains. In a bottom-up approach the existing requirements, which partly used to be assigned to use cases, were grouped underneath the top-down features. The feature model thus lead to a rearrangement of existing requirements, giving the opportunity to identify missing areas and to make the whole requirements base easier to understand through hierarchical decomposition. Overall, the user visible features became top level features, while internal features either ended up in the lower level of the feature model or in separate, more technical sub-domain feature models. Consequently, we classified the feature nodes into different types with a different set of attributes depending on their characteristics. The feature modelling tooling [6] is integrated with the requirements management tooling. The requirements meta model resembles most of the feature modelling meta model. This allows for importing the feature models into the requirements management tool and adding additional information and traces there.

2.2 Use feature modelling for the platform scoping negotiation process

Why? The business group is split into a domain engineering and several application engineering units. Requirements for the reusable asset base are not mined from customer contracts, but come from application engineering. Negotiations about which functionality should be a commonality and should therefore be supported by the platform had traditionally high conflict potential. Every application unit tried to get as much of their specific functionality into the platform as possible because platform development was pre-funded by application units. The challenge was then to consistently de-scope from all the requirements that had no or only low reuse potential.

How? The use case structure of the requirements had made a commonality/variability analysis among all involved units very hard. With the feature model that consists of user visible features on the top level and getting more detailed with features that reflect functional specification decisions a good communication basis is set up for negotiation. The application engineering units are interested in this detailed information about platform internals because they partially extend the platform features with product specific features or variants. The feature model is used as central repository for feature negotiation. First of all it makes it a lot easier to identify commonalities among applications because it forms a common vocabulary. Second, information about the value of a feature for each customer (i.e. how important is this feature to support a product and estimations how often this product will be sold) together with cost estimations of the platform development unit are the basis for prioritizing features. The decisions on what should be part of the business group became more transparent, decreasing the conflict potential considerably.

2.3 Trace features to the architecture

Why? For safety reasons, collecting tracing data is an important issue when developing medical software. Before the feature model was created, single requirements were traced from market requirements down to design specifications. However, this is very work-intensive, error-prone, and inefficient to maintain and even not required by regulation organizations.

How? The detailed tracing is replaced by tracing of features, which are an order of magnitude less than requirements, to architectural entities. In parallel to feature models, an architectural entity model reflecting the static structure of the architecture is built. This model is hierarchical like the feature model, only with subsystems, components and classes as the elements of this hierarchy. Features trace into the architectural elements in a many-to-many relationship. From this model it is then possible to investigate the effect of requirements on single architectural building blocks either in design specifications attached to building blocks or in the code.

2.4 Support project and iteration planning and project controlling with feature modelling

Why? After using the feature model for scoping, it is only consequent to use it for project planning and controlling as well. The development process is an agile, iterative one, therefore features are ideal items to be put into backlogs and be planned in iterations.

How? The features of the feature model are used as first class artefacts for project planning and controlling. They are augmented with attributes regarding the acceptance criteria for each feature, development status, and schedule. Therefore, the common vocabulary is not only present in product management and development but also in project planning and controlling. Furthermore, the linkage to the architecture models allows tracking the degree of completion of each feature. For iteration planning the features are further decomposed into iteration features that can be implemented in a single iteration. The iteration features are the smallest units for planning, but they are always seen in the context of their parent feature and are planned in a way that iteration features belonging to one feature are assigned to consecutive iteration steps.

2.5 The feature model as product derivation support

Using the feature model for platform configuration and derivation is a long term goal. To achieve this it is not sufficient to establish links form features to architectural building blocks. All variations have to be linked to the concrete variation implementations in solution space, e.g. to configuration parameters or removable application code building blocks. A derivation infrastructure has to be developed that evaluates the links and configures the application, e.g., by setting the parameters or by omitting building blocks according to the feature selection.

3 Experience with Variability Modelling for Product Derivation

Siemens VAI is the world's leader in the domain of engineering and building plants for the iron, steel, and aluminium industry and uses variability modelling techniques for product derivation in its CC-L2 product line. The product line provides process automation to continuous casting plants in steel mills and consists of several applications on different technical platforms like C++, Java and .Net, at a total of about 2 MLoC. Modelling techniques are used heavily in the server core, which consists of more than 800 components. To the average customer, about 600 selected components are delivered and custom extensions to the product line are made.
With their academic partner, the Christian Doppler Laboratory for Automated Software Engineering they developed the DOPLER approach [5]. Based on detailed sales support documents and the problem space knowledge of product management, the features and the variability of the product line were mined and consolidated into a model. This model has extended product derivation capabilities, as the features are attached with questions in natural language. During application engineering, answering the questions in close cooperation with the customer leads to decisions triggering the feature selection and therefore to a concrete product configuration. The resulting models are used as domain-specific language (DSL) to resolve the problem space variability together with the customer based on concrete product requirements.
The solution space of the CC-L2 product line comprises a component-based architecture. Because of the clearly defined mapping between problem space and solution space variability it is possible to automatically select and configure the assets required to build the desired product.

In the last years, the product line approach helped Siemens VAI to deliver more than 150 projects on schedule and on budget. Before, they had serious problems with code changes causing problems during start-up of plants. They were able to significantly reduce project execution time and travel times. Through defining a PLE evolution and planning process, Siemens VAI was able to reduce their development efforts and increased the reuse of software components

4 Lessons Learned and Conclusion

Important lessons learned while using feature models for scoping are:

- Early involvement of solution space knowledge: It is necessary to consider solution space knowledge early when identifying features and variants of a product line. In our examples products or systems were already built before the migration to product line engineering was started. The structure of existing systems helps to identify meaningful sub-domains. Linking features to existing solution space assets, or at least to architectural entities that are under design, helps to estimate cost early and keeps the whole effort grounded.
- Co-development of feature model: Development should be integrated early in building the feature model. There is considerable knowledge about past products or systems in development that helps to establish parts of the feature model with its variability quickly. The communication between product management and development furthermore leads to a common understanding of the requirements on the one hand and of the cost to implement those requirements, especially variability, on the other.
- Sub-domain division: Covering the whole problem domain with one feature model is too complex, if the goal is to model not only variability but to cover the whole system including all commonalities. Therefore, domains should be divided into sub-domains modelled in separate feature models.

At the other end of the life cycle feature models are very well suited to build DSLs for supporting automatic product derivation. The vast majority of variability in our domains is configurative variability. The hierarchical form of feature models makes them easy understandable by all stakeholders, not only the customer.

We did not do a project yet that combined feature modelling at both ends of the product line lifecycle. However, within the first described business group, we want to augment the feature model built for scoping to support product derivation.

5 References

1. Clements, P., Northrop, L.: Software Product Lines: Practices and Patterns. Addison-Wesley (2001)
2. Pohl, K., Böckle, G., van der Linden, F.: Software Product Line Engineering: Foundations, Principles, and Techniques. Springer (2005)
3. Kang, K.C., et al., Feature-oriented domain analysis (FODA) feasibility study. Technical Report CMU/SEI-90TR-21, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA, USA (1990)
4. Schmid, K.: A comprehensive product line scoping approach and its validation. In: 24th International Conference on Software Engineering, pp. 593--603. ACM (2002).
5. Rabiser, R., Gruenbacher, P., Dhungana, D.: Supporting Product Derivation by Adapting and Augmenting Variability Models. In: 11th International Software Product Line Conference, pp.141—150. IEEE (2007)
6. pure systems GmbH. Variant Management with pure::variants. Technical Whitepaper (2006)