Considering Product Family Assets when Defining Customer Requirements [1]

Günter Halmans, Klaus Pohl

University Essen, Software Systems Engineering,
Altendorfer Str. 97-101, 45117 Essen, Germany
Email: {halmans, pohl}@informatik.uni-essen.de

Abstract : *The success of a product family heavily depends on the degree of reuse achieved when developing product family based customer specific applications. If a significant amount of the customer requirements can be realized by using the communality and the variability defined for the product family, the reuse level is high; if not, the reuse level is low.*

In this paper we elaborate on the influence of customer requirements on the degree of reuse achieved within a product family. We argue that the level of reuse can be increased by considering the capabilities of the product family when defining the requirements for the customer specific application. We finally show how this can be supported by product family specific use cases.

1 Introduction

Software product families aim in decreasing the costs and time required to produce a customer specific product. In product family engineering one distinguishes between two development processes: domain engineering and application engineering (cf. Figure 1). In domain engineering the communality and the variability of the product family is defined. Shared assets are implemented such that the commonality can be exploited during application engineering while preserving variability [2]. During application engineering individual products are ideally being developed by selecting and configuring shared assets resulting form the domain engineering.

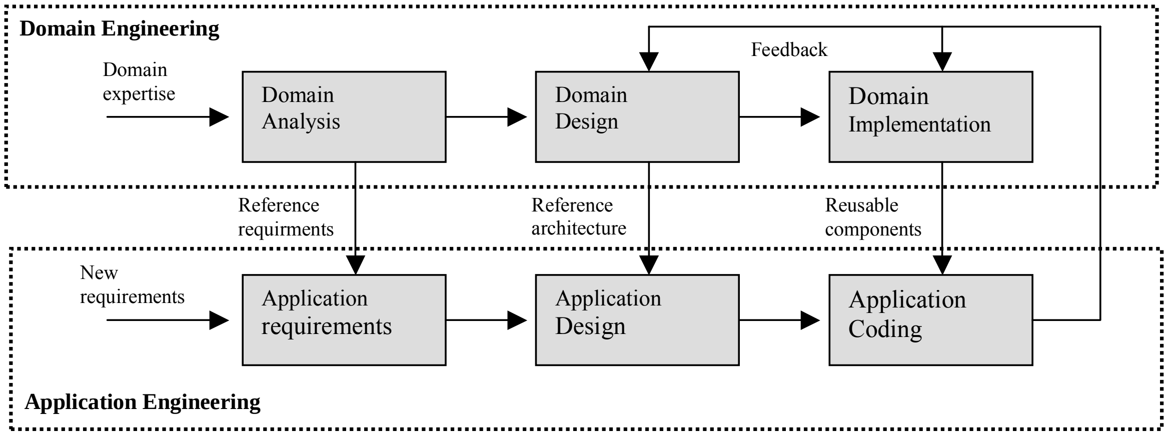

Figure 1: Domain and application engineering in product family development [1]

The success of a product family heavily depends on the degree of reuse achieved when developing product family based (customer specific) applications. If a significant amount of the customer requirements can be realized by using the communality and the variability defined for the product family, the reuse level is high; if not, the reuse level is low.

[1] This work has been funded by the ESAPS project (BMBF, Förderkennzeichen 01 IS 902 C/9 and Eureka Σ! 2023 Programme, ITEA project 99005).

As experienced in single product development, at the beginning of the requirements engineering process, customer requirements are often vague, i.e. the customer does normally not exactly know what she really wants. Requirements engineering is thus an iterative learning process in which the requirements engineer mediates the requirements definition among the stakeholders of the customers.

When making trade-off decisions during requirements engineering for a costumer specific application, the stakeholders should be aware of the consequences of their decisions with respect to capabilities of the product family. In other words, they should know which of the alternatives under consideration can be realized at low costs and in short time by exploiting the communalities and variability of the product family, and which of the alternatives conflict with the capabilities of the product family and thus lead to higher costs and longer development times.

Of course, an obvious approach to reach a very high level of reuse is to define a set of standardized product under the product family from which the customer can select his favourite. This works in situations where there is well-understood domain, and the understanding is shared by large parts of the potential market (customers). But even in those domains like financing (take SAP as an example), customer have specific requirements and want the standardized product to be adjusted to their needs. The need to develop products which do not dictate the requirements to the customer, but which satisfy the customers requirement is, according to our experience, increasing. As a consequence, also in product family development the customer has to be supported to identify and define his requirements – which from the product family provider point of view should largely correspond to the product family capabilities.

In this paper we identify what is required to empower the requirements engineer and the stakeholders to consider the capabilities of the product family when making trade-off decisions. (section 2). We argue that the level of reuse can be increased by considering the capabilities of the product family when defining the requirements for the customer specific application.

In section 3 we sketch two complementary types of support which help the requirements engineering in mediating between customer requirements and the product family capabilities. For supporting the trade-off decisions, we propose to classify each requirement with respect to its estimated realization effort. Therefore we propose seven categories. Each category represents a significant different way of realizing a customer requirements with the product family. Moreover, we propose use cases as means to provide product family specific information at the right abstraction level and discuss the extensions required to achieve this.

Finally in section 4 we summarize our contribution and provide an outlook on future work.

2 Requirements Engineer as Mediator between Customer Requirements and Product Family Capabilities

The first "phase" in the application engineering is eliciting, negotiating, documenting and validating the customer requirements for the desired customer specific application. Requirements engineering of product family applications differs significantly from requirements engineering in single product development. The main difference is that when performing requirements engineering within the application engineering process, one has to consider the capabilities of the product family based on which the customer specific application should be build. Reusing product family assets when realizing customer specific requirements reduces the costs and development time for the application and increases the return on investment for the product family provider; i.e. the more customer requirements fit with the product family capabilities, the higher the return on investment.

In the following we sketch the principle activities which empower the requirements engineering to provide product family information to the stakeholders during requirements engineering for a customer specific application (see also Figure 2).

First of all, the requirements engineer needs to know the capabilities of the product family (see (1) in Figure 2). Given that a industrial product families easy realize 1000 or even more requirements, and given that the realisation encompasses (several) hundred components. Moreover, existing variation points and their variants are not independent, i.e. there are dependencies between the variation points which have to be considered when exploiting the variation points for building customer specific applications. Making the requirements engineer aware of the product family capabilities is thus by far not trivial.

Second, in the discussion with the customer the requirements engineer has to transfer the knowledge about the product family to the customer (see (2) in Figure 2). A customer has a different viewpoint on the product family than the requirements engineer. For example, a customer typically does not care about variation points, variants or binding times. In general, the knowledge exchanged in the task (2) is more abstract than the knowledge in task (1). The requirements engineer must thus continuously mediate between the customer and the product family capabilities. The requirements engineer must thus continuously "transform" the knowledge of the product family in terms the customer is able to understand. We will discuss possible support for this task in section 3.

Third, after the customer requirements are established, the requirements engineer has to transfer the result to the product developer. To support this task, the requirements engineer ideally provides some kind of requirements classification by which the potential realisation effort and the effects on the product family are indicated. Besides the realisation of the requirements, she should also provide some indication for requirements which cause a change to the product family, e.g. since they require new variation points or new components to be integrated in the product family.

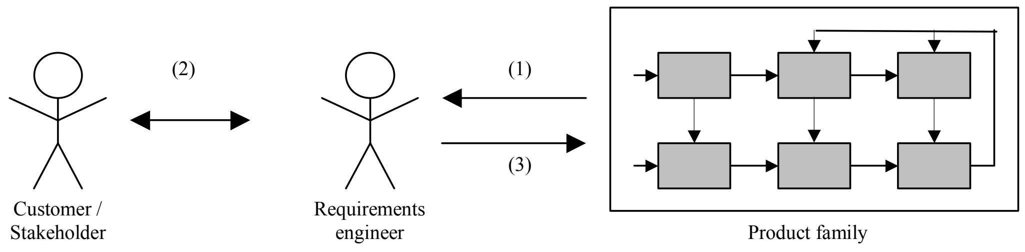

Figure 2: Requirements Engineer as Mediator

Due to space limitations, we focus on the second task (2), the mediation between the customer and the product family capabilities. However, the solutions described in section 3 provide an ideal basis to support the tasks (1) and (3) sketched above.

3 Communicating Product Family Capabilities to the Customer

We provide two types of support for the requirements engineer to communicate the influence of alternatives in trade-off decision to the customer:

- we categorize customer requirements with respect to their realization in a product family and sketch their use in supporting trade-off decisions (section 3.1);
- we describe how use cases which appropriate extensions could facilitate the mediation task of the requirements engineer (see section 3.2)

3.1 Realizing Customer Requirements under a Product Family: A Categorization

When realising (implementing) a customer requirement under a product family, one can distinguish between seven principle categories. Each of those categories stands for a significant different way of realizing a customer requirements with the product family. The effort required in terms of time and costs differs significantly for each of the seven categories defined in the following:

A: **Asset reuse:** Requirements which fall in this category can be realized by simply reusing a core asset of the product family; i.e. this category causes no implementation effort for satisfying a customer requirement;

B: **Binding a variant:** Requirements which fall in this category can be realized by choosing a variant provided by the product family for a given variation point and by defining the actual binding time of the requirement, e.g. during compilation, during runtime. To satisfy the requirement and adaptation of the configuration of the application might be required. However, the implementation effort required is almost zero;

C: **Creating a new variant:** Similar to B, the requirements in this category can be realized by binding a variation point. In contrast to B, the variant required to satisfy the customer requirement does not exist and must be realized. Depending on the complexity of the variant this could require some implementation effort; overall the implementation effort is low and does not effect the architecture of the product family. Moreover, the newly created variant can be used for future application development;

For realizing a requirement which falls into category A,B or C the variability of the product family is used and thus the implementation effort is very low. In contrast, the realization of requirements which fall in the categories D,E,F or G require a change of the core assets of the product family itself. Those, depending on the category and the change required, the implementation effort for realizing a requirement could be significant; in some cases it could be even impossible to satisfy the requirement with the product family (category G).

D: **Removing functionality:** Requirements which fall in this category require that some functionality or even components are removed for the product family and that this can not be achieved by a appropriated configuration of the variation points of the product family. In other words, the product family provides core functionality which is not desired by the application under development. If the functionality could be removed without changing the architecture of the product family (e.g. by replacing for the customer specific applications the product family components by customer specific components with reduced or restricted functionality) the requirements fall in this category. If the removal leads to a change of the overall product family architecture, the requirements fall in category F. The implementation effort required might differ significantly. However, in average a requirement in this category requires more effort than a requirement in category C, and more effort than a requirement in category E

E: **Adding functionality:** Requirements which fall in this category require that some functionality is added to the product family. Note that when adding the desired functionality could be achieved by using existing variation points or by implementing a new variant the requirements fall in category B or C respectively. Requirements which fall in this category those require the integration of new component(s) into the product family. In other words the implementation effort is the sum of the effort required for developing the component(s) and the effort required for integrating the components into the product family. . If the integration leads to a change of the overall product family architecture, the requirements fall in category F.

F: **Changing the architecture of the product family:** Realizing the requirements which fall in this category require a change of the product family architecture and those influence all existing product-family-based application. We do thereby not distinguish if a new functionality is added or if existing functionality is removed through the integration of new variation points and variants, or even a combination of both. The high development effort required to satisfy the requirements in this category mainly comes from the fact that the changes made influence already existing applications and lead to a new version of the product family architecture. The effort required to satisfy a requirement of this category is, in average, those significantly higher than for satisfying a requirement of the categories A-E:

G: **Impossible to realize**: Requirements which fall in this category essentially cause that the whole customer specific application can not be realized based on the product family. We call those requirements also “killer requirements”. In other words, the realization of a requirement of this category with the product family requires more effort than a realisation of the overall application without the product family. However, also in this case some assets of the product family might be reused when developing a product-family-independent application.

Figure 3 depicts the average level of reuse achieved when realizing a requirements of the seven categories defined above.

As already stated, the implementation effort required to satisfy a requirement of a given category might significantly differ depending on the “complexity” of the requirement itself. This holds especially for the categories D,E and F. For example, a requirement of category F could be lead to less implementation efforts than a requirement of category D; especially if the requirement in category F is of less complexity as the requirement of category D.

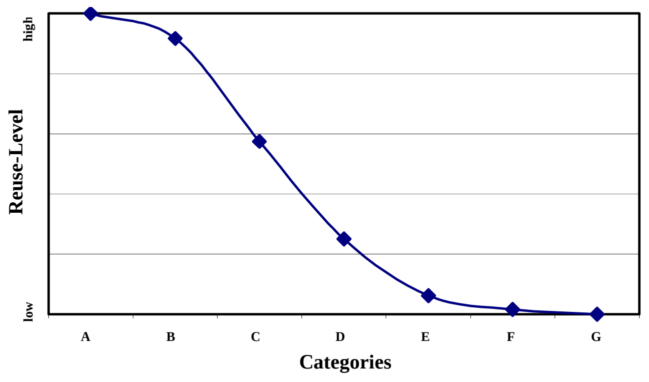

Figure 3: Categories and their Reuse-Level

However, when making trade-off decisions about alternatives for a given requirement, the alternatives under consideration are most likely of similar complexity and thus the difference of implementation effort within each category do not matter.

Classifying the requirements (alternatives under consideration) in the above categories is thus a good estimate for the realization efforts expected to satisfy the requirements. The categories provide excellent means to communicate the estimated realization efforts to the customer.

We thus use the categorizes for supporting trade-off decision during requirements engineering.

3.2 Extended Use Cases

The application of use cases has proven in practice and research to achieve better customer involvement and thus better requirements specifications. Use cases set requirements into a certain usage context and thereby reduce complexity and make requirements easier to understand.

Given this positive feedback from single product development – which is less complex than product family development – we suggest use cases for transforming information about functional capabilities, and their variability (variation points; variants) of the product family to the customer. To be able to apply use cases during application engineering in a product family context, however, some extensions are required. In other words, the use case templates proven successful in industrial praxis (cf. e.g., [3; 4; 5]) have to be extended to

- ***Capture variability:*** The variability within a use case has to be documented. All involved variation points associated with the actual use case have to be registered. For example in a use case template a section which points out the number of variation points can be added. Further more the variation points have to be represented in the use case diagrams. Also the activity diagram has to be extended, so that the consequences of binding a variation point in the dynamic of a process can be documented. Numbering all variation points enables the requirements engineer to check the possible alternatives within a given context which is important for the understanding by the customer.
- ***Capture binding times:*** In software product lines, variability is made explicit through variation points which are bound to a particular variant at a certain time (cf. [2] for details). For empowering the discussion of the requirements engineer with the customer it is essential to document these possible variants within a given context. So for every variation point which is numbered in the use case the possible variants and the binding conditions have to be listened and to be described. That means for every variant in the use case a main path scenario has to be described. Also for every variant main path an activity diagram can be defined and the related constraints have to be documented. With these information the requirements engineer is able to describe the particular alternatives within a use case.

- ***Document the consequences of the variation point binding:*** The several variants which can be realized for example by binding one variation point effects other use cases or other variants from other variation points within the use case. For describing a potential behavior of the product by concretization a variant it is necessary to know these effects. So in the use case template these associations have to be documented. This can be done by using extensions in the use case-diagram.
- ***Capture the categorization***: As mentioned in section 3.1 to categorize the requirements in different ways of realization support the requirement engineer and the customer in trade off decisions during defining the product. So these categorization of the requirements have to be documented. Normally several requirements are associated with one use case and therefore it may occur that the use case capture requirements from different categories. In an use case template, which numbers the associated requirements, the category of each of them should be documented by an additional attribute. In other cases the occurred categories have to be evaluated and a categorization of the use case may depend on which category the main requirements have.

Use cases are ideal facilitator for empowering the requirements engineer in the discussion with the customers stakeholder because they define a context with for example a special goal and given actors. This enables the requirements engineer to explain the variability in special scope. This is essential because of the complex product families. As mentioned in section 2 the categorization of the requirements makes it possible to give a first estimate of the related costs

4 Conclusion

Requirements engineering during application engineering has to consider the product family capabilities if the customer specific application should be realizable with a high degree of reuse. In this paper we have briefly discussed the challenges requirements engineering for customer specific applications faces in a product family context.

To support the requirements engineer in "mediating" the product family to the customer – for example when making trade-off decisions about possible alternative requirements - we have suggested

a) to classify the requirements with respect on their realisation efforts and times, i.e. the level of reuse achieved
b) to use apply extended use case to transfer knowledge about the product family, e.g. alternatives provided by variation points, to the customer.

We have thereby concentrated on one – out of three potential tasks – the requirements engineer has to perform when defining with the customer the requirements for a product-family-based applications. The solutions sketched in this paper, however, also support the other three tasks, e.g. if the product family is defined based on use cases, changes to a use case required by a customer can be captured as "deltas" and thus the realisation of the changes can be supported (see [6] for details).

We will apply the support sketched in this paper in an industrial setting. Although a validation appears to be non-trivial, we hope to be able to report on first results of this experiment soon.

References

[1] ITEA: 4th ESAPS Workshop on Derivation of Products and Evolution of System-Family Assets;2-4 April, 2001 Bad Kohlgrub

[2] Jan Bosch, Gert Florijn, Danny Greefhorst, Juha Kuusela, Henk Obbink, Klaus Pohl: Variability Issues in Software Product Lines, Fourth International Workshop on Product Family Engineering (PFE-4), Bilbao, Spain, 2001

[3] Alistair Cockburn: Writing Effective Use Cases; Addison Wesley, 2001

[4] Hans-Erik Eriksson, Magnus Penker, Buisiness Modeling with UML, Business Patterns at Work; OMG Press, 2000

[5] Putnam P. Texel, Charles B. Williams: Use Cases combined with BOOCH OMT UML; Prentice Hall, 1997

[6] Pohl, K.; Brandenburg, M.; Gülich, A.: „Scenario-based Change Integration in Product Family Development", 2nd ICSE Workshop on Software Product Lines: Economics, Architectures, and Implications, May 2001