Feature Scoping for Product Lines

Matthias Riebisch, Detlef Streitferdt, Ilka Philippow
Ilmenau Technical University, Ilmenau, Germany
{matthias.riebisch|detlef.streitferdt|ilka.philippow}@tu-ilmenau.de

Abstract

Product Line (PL) Engineering focuses on the development of complete system families as opposed to single systems. Systems are built of a reusable platform common to the whole family, and of specific parts extending it. The benefits of short time-to-market and lower development costs for each system within the system family are achieved by reusing the platform for each new system to be developed. Therefore the scoping of features for the reusable platform and the specific parts is crucial for PL success.
This paper proposes scoping with 4 priority levels and a decision-table based interpretation of the results. The interpretation is shown both for start and for evolution of product lines. The paper is based on experiences on large-scale reuse in industrial software projects.

1 Introduction

Reusability of software has been and is an important goal of software engineering researchers and practitioners. Experiences with different approaches have shown that both technical and organizational, economical, and psychological factors are crucial for success of s/w reuse. Product Line Engineering (PLE) focuses on the development of complete system families as opposed to single systems. By defining a reusable platform for all members of the family, a professional and planned way of software reuse is possible, based on domain analysis and other technologies. Additionally, PL offer a way for economic planning, e.g. for the return-on-invest and other key figures.

However, there are still various steps which are critical for economic success. Scoping is one of the most important steps, and one of the less formalized ones among them. Generally, scoping is the process of deciding about the effort of a software development task. This decision is influenced by the market situation, the customer needs and expectations, the strategic business goals, and the estimated effort for this task. Schmid's survey on scoping in PLE [Schmid 2000] classifies three ways of scoping:

- identifying products that should be part of a product line: product portfolio definition
- bounding the domains which are relevant: domain-centric scoping
- identifying the specific assets that should be part of the reuse infrastructure: asset-centric scoping)

In this paper, scoping focuses on the decision, which features should be implemented in the reusable platform on in a variable part of the PL. A feature is a property or a quality of a product. The set of features implemented in a PL are modeled in a so-called feature model (see [Czarnecki et al. 2000]). This way of decision is very similar to that supported by [DeBaud 2000], but extended by the decision about features which are supported by the reusable platform, even if they are outside the platform.

For the described step of scoping there are descriptions in recent publications, e.g. [DeBaud 2000] and PuLSE-Eco in [Bayer et al. 1999]. In practice however only sparse support for interpretation of scoping results as the basis for further development steps can be found.

This paper is organized as follows: After a short investigation of scoping in conventional software development there is a proposal for classifying scoping results in PLE by priority levels. An interpretation scheme of this priorities for deciding the next implementation tasks is described. In a separate section, additional propositions for further discussion during the workshop are listed.

2 Scoping In Conventional Software Development

Scoping in conventional software development – without PLs – is performed as part of the requirements engineering activities. Its predecessor activities are requirements elicitation and requirements modeling, which are followed by design and implementation phases. As mentioned above, scoping is influenced by the current market situation, the customer expectations, the strategic business goals, and the estimated effort for implementing a particular requirement resp. feature. Scoping results in the form of priorities are assigned to the list of features. Often these priorities occur in three levels:

1 - to be implemented (as part of the next development cycle),
2 - to be implemented in a later development cycle,
3 - not to be implemented (at the moment).

The task of assigning priorities consists of two main steps: the definition of business goals and their assignment to features, and the calculation of priorities. There are successful techniques for performing these two tasks: the Goal-Question-Metric technique GQM [Solingen et al. 1999], and the Quality Function Deployment QFD [Sullivan 1986].

3 Scoping In Product Line Engineering

In PLE, scoping is one of the most critical success factors.

- It influences the development effort during later changes of the reusable platform. Scoping represents the planning of reuse and therefore influences the return-on-invest.
- If performed according to the customer needs, it enables a short time-to-market and low development costs.
- By planning reusability, it leads to a higher process maturity for the reusable platform. Thus, software quality and evolvability are improved. Evolvability enables a longer usage time for the parts of the PL and therefore influences the return-on-invest, too.

In PLE, priorities are used for the assignment of features to development cycles as well as for their assignment to the reusable platform itself or to the variable parts respectively. Thus four priority levels are proposed:

1 - to be implemented for all systems in the PL, part of the reusable platform
2 - to be implemented for some systems in the PL, variable parts,
3 - to be implemented for some systems in the PL in a later development cycle, at the moment, however, no implementation
4 - not to be implemented (at the moment).

The consequences of the priority assignment depend on the stage of the PL development. Either it is just started or it is to be evolved. The next section deals with these issues.

The new criteria for deriving priorities are a refinement of those in the preceding section. Differences are described as follows:

- The market situation and the customer expectations are analyzed to rate the current situation and to predict future trends, since the future has much more impact on PLE than on single system development.
- Strategic business goals correlate with investments in the PL's reusable platform. They have to be consistent with the marketing strategy and the organization of the company.
- The effort for implementing a particular requirement resp. feature is usually estimated by experts. Due to the higher complexity of a PL such estimations are more complex, too. However, in the case of an existing PL with existing feature models, cost estimations can be partly replaced by assessing traceability links. Such links offer possibilities for automating several development steps (see [Philippow et al. 2001]).

For the calculation of priorities, methodologies similar to QFD are used. In TrueScope [DeBaud 2000] and in PuLSE-Eco [Bayer et al. 1999] these calculation schemes are called product map.

Interpreting Scoping Results For Successful Product Line Development

The results of scoping are priorities assigned to features in the feature model. To decide the next steps of development these priorities have to be interpreted. In literature, there is no systematic way for decision-making. The interpretation of the priorities depend on the development stage of the PL. In order to achieve a more systematic way of interpretation, the use of decision tables [Kohavi 1995] for architecture, design and coding decisions is proposed. Decision tables are in use in several branches of engineering, e.g. automation engineering. A decision table consists of scheme of input values as columns and resulting actions in the rows with conditions in the body of the table. In one step, all actions are performed, where the conditions are met by the input values. If in this step there is no matching action, an optional default action is performed. Using the decision table, a decision about the implementation of each feature of the PL is made. Here, we discuss three cases: the start of a PL development from scratch, the start of a PL development from existing assets, and the evolution of an existing PL.

Start of a Product Line Development

The starting of a PL development from scratch is the simplest case. The decision table is very straightforward (Tab 1): Prio-1 features are implemented within the reusable platform. Prio-2 features are

implemented by variable parts of the PL. The interfaces of these parts have to be supported by the reusable platform.

Under some conditions prio-3 features influence architectural decisions for the reusable platform: The architecture of the reusable platform is prepared for later implementation of these features, if the effort for later refactoring is significantly higher than the preparation of the reusable platform at this moment, and if the complexity of the platform's architecture is not significantly increased by this decision. This way, decisions are influenced by the intentions of the Extreme Programming approach [Beck 1999]. Although these considerations have not been discussed in current papers, they might lead to economic advantages.

All other prio-3 and prio-4 features are neither considered for decision-making nor for implementation.

Tab 1: Decision Table for the Start of a Product Line Development

...is of prio 1	…is of prio 2	…is of prio 3	…leads to significant increase of complexity	…leads to higher refactoring effort later than now	Feature in question … / Resulting action
yes	no	no	don't care	don't care	To be implemented as part of the reusable platform
no	yes	no	don't care	don't care	To be implemented as variable part
no	no	yes	no	yes	Platform architecture to be prepared for future support
					Default: no change

Start of a Product Line Development with Existing Products

In this case existing products need to be integrated in the PL. They will have to be refactored in order to divide them into parts according to the features. The refactoring effort partly leads to economically-driven decisions for re-development of large portions of the products. This effort has to be considered when estimating the total implementation effort. All other decisions are identically to the section above.

Evolution of an Existing Product Line

Here, the resulting decision table is more complex (Tab 2). Features with priorities of 3 or 4 are not implemented. Prio-1 and prio-2 features are implemented in the reusable platform or by variable parts of the PL, respectively. If the priority of a former prio-1 feature decreases resp. prio-2 feature we have to distinct, if its priority is increased or decreased, compared to the previous development cycle. In these cases a refactoring has to be done.[1]

Tab 2: Decision Table for Evolution of an Existing Product Line

…was previously part of the core	…was previously in a variable part	…was previously not implemented	...is of prio 1	...is of prio 2	Feature in question … / Resulting action
no	no	yes	yes	no	To be implemented as part of the reusable platform
no	no	yes	no	yes	To be implemented as variable part
yes	no	no	no	don't care	Reusable platform to be refactored in order to remove this feature; variable parts to be adjusted accordingly
no	don't care	don't care	yes	no	To be included in reusable platform; variable parts to be adjusted accordingly
					Default: no change

4 Propositions for Discussion

The following issues came up during discussions in our research team. Our intention is to put the given statements as "requests for comment" during the workshop.

- Scoping is one of the most critical steps for economic success in requirements engineering of PLs. While there are formalized ways for the elicitation of business goals (GQM) and for the computa-

[1] As the attentive reader will notice, the 1st and 4th row of the decision table could be joined in a normalization step, because the corresponding actions differ only slightly.

tion of scoping priorities (QFD), there is no systematic way for interpreting the results and deciding about the next steps. Decision tables offer such a way.

- Similar to the Extreme Programming (XP) philosophy all development task should be simplified as much as possible. Especially no effort for future requirements is to be invested to keep the reusable core simple and to assure a high evolvability.
- The thresholds for the priority levels 1 to 4 are defined on cooperation of management, marketing and development. They can be verified in every development cycle.
- In some cases there is a need for an additional priority level between 2 and 3 for requirements, which are to be implemented in a later product, but resources for them have to be implemented or to be prepared in the reusable platform in this development cycle.
- Effort estimation for scoping purposes is usually carried out by developers based on their expert knowledge. If the PL development is documented using feature models and traceability links, then effort for changes can be derived by tools in a more systematic way. Traceability links could be collected in a way similar to e.g. TrueScope's Increment Control List.
- Marketing and customer relations have adopt the PL philosophy. Their support for the evolution of a PL is more critical for long-term success than technical development.
- Similar to the experiences with software reusability, software quality requirements for the reusable platform are stronger than for other parts. This fact has to be considered in planning development cycles.

References

[Bayer et al. 1999] J. Bayer, O. Flege, P. Knauber, R. Laqua, D. Muthig, K. Schmid, T. Widen, J.-M. DeBaud: PuLSE – A Methodology to Develop Software Product Lines. Symposium on Software Reusability, Los Angeles, CA, USA (SSR'99), 1999, pp. 122-131.

[Beck 1999] Beck, Kent: Extreme Programming**{** XE "Extreme Programming" **}** Explained: Embrace Change. Addison Wesley Longman, Reading/Massachusetts, 1999.

[Czarnecki et al. 2000] Czarnecki, K., Eisenecker, U.W.: Generative Programming. Addison Wesley, Reading, MA, 2000.

[CQM 1993] -: Kano's Method Special Issue. Center for Quality of Management Journal, ISSN 1072-5296, Vol. 2, No. 4, Fall 1993, http://cqmextra.cqm.org/cqmjournal.nsf/issues/vol2no4

[Clements et al. 1998] Clements, Paul, Northrop, Linda M., et al.: A Framework for Software Product Line Practice – Version 1.0. SEI, CMU, Sept. 1998. http://www.sei.cmu.edu/plp

[DeBaud 2000] DeBaud, Jean-Marc: TrueScope – A Full LifeCycle Approach to Develop Software Product Lines. In: [SPLC 2000], Tutorial 6.

[Kohavi 1995] R. Kohavi: The Power of Decision Tables. In the European Conference on Machine Learning, 1995. http://robotics.stanford.edu/users/ronnyk/ronnyk-bib.html

[Philippow et al. 2001] Ilka Philippow, Matthias Riebisch: Systematic Definition of Reusable Architectures. In Proceedings of the 8th IEEE International Conference and Workshop on the Engineering of Computer Based Systems (ECBS 2001), April 2001, pp. 128-135.

[Schmid 2000] Schmid, Klaus: Scoping Software Product Lines. In: [SPLC 2000], pp513 – 532.

[Solingen et al. 1999] R.Solingen, E.Berghout: The Goal/Question/Metric Method. McGraw-Hill Publishing Company, 1999.

[SPLC 2000] Donohoe, Patrick (Ed.): Software Product Lines – Experiences and Research Directions. Proc. 1st Software Product Lines Conf. (SPLC1), Aug. 28-31, 2000, Denver, Colorado, Kluwer Acad. Publ. 2000. http://www.sei.cmu.edu/plp/conf/SPLC.html

[Sullivan 1986] L. P. Sullivan: Quality function deployment. Quality Progress, 19(6), 1986. pp. 39-50.