A Cost-Based Approach to Software Product Line Management

Holger Schackmann, Horst Lichter
{schackmann, lichter}@informatik.rwth-aachen.de
RWTH Aachen University. Software Construction Group. Ahornstr. 55, 52074 Aachen, Germany

Abstract

The evolution of a software product line requires different product management practices compared to a single product since the diverging requirements of different customers must be coordinated to preserve the common product line architecture. Allowing too much variability leads to substantial follow-up costs during the lifecycle. This paper describes the challenges of product management for an evolving software product line. A costing approach is proposed to enable more transparency on the costs of variability, support sound decisions on the appropriate amount of variability and gain control on the development process.

1. Introduction

The development of a software product line (SPL) aims at exploiting commonalities between the products in all phases of the development process. What are the differences between product management for an SPL and product management for a single product?

The management of a product within an SPL can not be detached from the management of the other products due to dependencies between development artifacts and use of the same resources. Thus product management in SPLs is characterized by a large number of stakeholders that directly or indirectly exert influence on product management and can be affected by its decisions.

Furthermore, products may be situated in different stages of their lifecycle. While a set of products will be established as product line members with regularly maintenance releases, there will be ongoing development of new products and possibly the integration of formerly independent products.

Thus product managers must constantly aim at aligning the diverging needs of different products towards the SPL approach; otherwise the promised synergies can not be achieved.

The remainder of this paper is organized as follows. Section 2 depicts the difficulties imposed upon management of an SPL. Section 3 draws comparisons to variability management in production engineering. Section 4 then proposes an approach to variability management for SPLs based on the application of activity based costing. Section 5 gives an outlook on further work and section 6 provides a summary.

2. Challenges of product management for software product lines

In the following we will depict a list of challenges for SPL management, also based on observations with industrial cooperation partners.

2.1. Insufficient lifecycle scoping

Scoping is a part of domain analysis for SPLs. According to Schmid the task of scoping is planning and bounding what should be made reusable [1]. Based on an analysis of the commonalities and variabilities of the underlying domain, it must be decided which features should be present in all products, which should be implemented as optional or alternative parts, and which features should be developed in a product specific way.

While existing scoping approaches are focused on initial scoping (see [1] for an overview), the problem of scoping during maintenance and evolution is rather unexplored. In practice we encounter that SPLs are introduced gradually by exploiting commonalities between products that had been developed in customer specific projects. Thus an exhaustive scoping was not performed or even no systematic approach to scoping was used. Moreover scoping decisions may not be documented, or lost during further development.

2.2. High coordination efforts

During maintenance and evolution of an SPL, new or changed requirements have to be integrated which

Software Product Management, 2006. IWSPM '06. International Workshop on , vol., no., pp.13-18, Sept. 2006
doi: 10.1109/IWSPM.2006.1

do not fit to the initial scope and therefore require the adoption of existing core assets as well as introducing additional variation points. Since these changes may affect other existing or planned products, each change must be coordinated with other development projects within the SPL. Moreover, different customers may impose conflicting constraints on delivery dates. This requires considerable efforts for coordination.

2.3. Hindrances to systematic reuse

Since software development takes place under continual deadline pressure it may often be faster to produce a customer specific solution instead of spending additional effort on investigating chances for reuse and coordinating necessary changes induced by an urgent customer request with other products.

Within an SPL there may typically be products which have a bigger impact on business success, need larger resources and undergo a more dynamic development. These products have stronger impact on core asset development whereby requests related to other products may not be considered sufficiently. Hence developers of less important products may again be coerced to construct product specific solutions.

2.4. Insufficient product communication

The many temptations to product managers to introduce new customer specific features or adaptations of existing features, lead to a large amount of variability accompanied by an increasing technical and cognitive complexity.

The offering of features within an SPL can therefore not completely be communicated to the customer. Thus aligning customer requests with existing features of the SPL as well as identifying chances for reuse becomes difficult. Either it is not known to the analyst that there exist other potentially reusable features, or the similarities to features requested by the customer are not recognized. Similar features may be developed independently several times. If this is perceived later, it will require considerable efforts to merge these features into shared assets. But if no counteraction is performed, the scarce resources of the development organization will increasingly be loaded with coordination efforts.

2.5. Lifecycle costs of variability

Added initial development costs due to missed chances for reuse are not the only problem caused by large variability. Each customer specific feature must be maintained, integrated in subsequent releases, tested separately and possibly considered during deployment, user training and customer support. Requirements engineering for new and existing products becomes more complex and therefore costlier.

Summarized, variability is a cost driver during the whole SPL lifecycle.

2.6. Misunderstood customer orientation

The concept of customer orientation may guide product managers to fulfill many new customer requests with new customer specific features. But with regard to the follow-up costs, it must be examined whether a new feature offers a corresponding business value to the customer, or if the request can also be fulfilled with a similar existing feature. In this case it must be pointed out to the customer that an adaptation of the requirements can result in a better fit to the SPL. The customer can profit by better quality, better support and reduced maintenance costs, if his product relies more on the shared assets.

2.7. Lack of economic incentives

The mentioned difficulties indicate that managing variability is the core task of product management for SPLs.

One problem is that there is no sound basis of decisionmaking. Development costs of a customer specific feature can possibly be estimated. But neither the resulting follow-up costs will be transparent; nor the business value of a specific feature will usually be assessed. Therefore there is no clarity on the effects of variability on costs and accordingly a lack of economic incentives to guide managerial decisions.

We believe that this is a major barrier for successful further development of an SPL in the long term, which not had been sufficiently addressed by SPL research yet. Existing cost models for SPLs rather follow a top down approach (see [2] for an overview) and support SPL investment decisions on a high level. Developing an approach to gather the costs of variability more accurately will provide complemental input to enable better estimations of future costs and sound variability management decisions.

3. Variability management in production engineering

The depicted challenges resemble a situation perceived in manufacturing industry during the eighties. Due to competitive pressure manufacturers increased the variety of their product portfolio with the objective of product differentiation. Only later it was

Software Product Management, 2006. IWSPM '06. International Workshop on , vol., no., pp.13-18, Sept. 2006
doi: 10.1109/IWSPM.2006.1

recognized that the expected economies of scale were overestimated, and the costs of additional variants (diseconomies of scope) were not considered sufficiently [3]. The added complexity leads to increased costs in all phases of the product lifecycle, including development, production, sales and customer support.

3.1. Deficiencies in costing systems

A large part of these costs can not directly led back to corresponding product variants. In traditional cost accounting systems these overhead or indirect costs are allocated to products on a per-unit basis. This leads to an unconsciously cross-subsidization. More exotic product variants will be sold below their actual costs, thus leading to further competitive disadvantages for standard products whose prices are charged with increasing overhead costs. A collateral effect of complexity in the product portfolio is the deterioration of market power due to a declined effectiveness of the distribution system, longer reaction rates on the market and cannibalism in the product portfolio.

3.2. Variability management

Due to these problems variability management went into the focus of product management. It was realized, that the maximum benefit neither lies in arbitrary widening the product portfolio nor in radical avoidance of new variants. The costs of additional variants must be balanced with the value offered to the customer. The achievable additional customer satisfaction does only increase on a degressive scale with the number of possible variants. Therefore product management has to find the economic optimum in the number of variants.

3.3. Activity based costing

The depicted deficiencies of traditional costing methods led to the development of new approaches like activity based costing [4]. This approach seeks to identify the activities in the product lifecycle and to determine their costs. Cause and effect relationships must be analyzed to find out how to allocate activity costs to products, services or customers. So most of the former overhead costs can be allocated more accurately. This enables the identification of unprofitable products, services, or customer relations. Product managers can react in many different ways to establish profitable customer relationships, e.g. by changing the prices, reconfiguring or replacing products, improving production processes, changing the business strategy or eventually abandon a product completely.

3.4. Analogies to software product line management

Can these approaches be transferred to the management of SPLs? Analogously the costs of developing and maintaining an SPL are for the greater part indirect or overhead costs, which can not easily be allocated to a certain product or customer.

But since there are no or only marginal production costs in software development, cost structures are totally different to production engineering. Costs are mostly independent from the number of sold units of a product variant. Therefore one can not easily divide between standard products and exotic products in the product portfolio and assume the latter generate more overhead costs. With an adequate product line architecture those exotic products might be easily configured based on the core assets.

But as described before, the number of customer specific features has a large impact on lifecycle costs. Therefore one can analyze which features are standard features that are relevant for most customers and which features can be seen as more exotic features, only included in products for one or a few customers. Gathering the costs caused by these features more accurately, would help to attain more clarity on the influence of variability on costs.

The prevailing approaches to costing in software development have the limitation that they are unable to gather the costs caused by variability. Costs are usually allocated either to customer projects, maintenance projects or internal development projects. This may suffice for accounting and budget control. But with multiple reuse the relationship between direct labor hours that went into development and the costs of software breaks down [5]. Fichman and Kemerer therefore propose the adoption of activity based costing to component based software development. Since systematic reuse is the core of SPL development, an activity based costing approach will presumably be suited for the information needs of managing SPLs.

4. Variability management for software product lines

In this section an approach to variability management is proposed, based on activity based costing and the assessment of the customer value of the variability. Subsections 4.2 to 4.4 depict how this information can be utilized to achieve the following goals:

Software Product Management, 2006. IWSPM '06. International Workshop on , vol., no., pp.13-18, Sept. 2006
doi: 10.1109/IWSPM.2006.1

- Estimation of future costs of variability
- Guidance of scoping decisions and strategic steering of variability
- Process improvements in SPL development and customer support

4.1. Activity based costing for software product lines

The application of activity based costing to software development can be based on several existing techniques. Defined development processes aid in identifying the relevant activities. Change request management systems, task management systems or time registration systems basically enable a detailed gathering of labor hours for most activities.

But the allocation of activity costs to customers and software components respectively raises many difficulties. It is not clear how development and maintenance costs of core assets can be distributed to customers. The use of software components as cost objects is problematic when there is no direct relation of an activity to a certain component, e.g. activities like requirements engineering or system testing. Therefore we propose the use of features as the primary cost object for activity based costing in SPL development.

Feature modeling was established as a technique for modeling commonalities and variabilities in requirements engineering for SPLs [6][7]. Basically feature models present a hierarchical structuring of features and contain domain relations like alternative or optional features and furthermore dependencies between features.

Since features are visible in all phases of the development lifecycle, most development assets, like requirements, software components, test cases, documentation, change requests and even support calls can be linked to one or more features. Thus the cost for most activities can be distributed to features as cost objects.

The cost allocation from features to customers must be based on an assessment of the importance of the provided features to the different customers (Figure 1). In case of features that are shared between different customers, the costs have to be divided. In order to do this, the value of the feature for each customer must be assessed independently. The distribution of the costs can then be based on proportions of these assessment values.

Hence an integral part of this approach to variability management must be the analysis of the value a feature provides to customers. It might not be possible to express this value in monetary units, but it suffices to assess the contribution of a feature to user satisfaction on a simple scale of values. Techniques like the Kano method [8] or Quality Function Deployment [9][10] may be applied to this purpose.

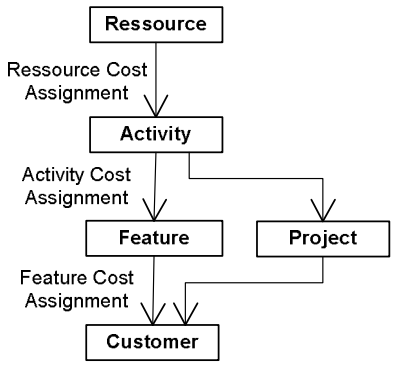

Figure 1: Features as cost objects

4.2. Estimation of variability costs

Gathering experience with accurate cost information of features will enable an analysis of the relation between initial development costs of a feature and its follow-up costs.

A better understanding of the cost implications of variability will allow better estimations of future maintenance costs. This can guide decisions on the architecture of the product line, e.g. on the use of certain variability mechanisms or the merging of existing product variants or features. Furthermore cost estimations can be employed for quotation costing and negotiations during early requirements engineering phases.

4.3. Guidance of scoping decisions

Bringing together cost information based on features as cost objects and the assessment of the customer value of features provide a sound guidance for scoping decisions (figure 2).

Features which are not that important for the customers but generate high additional costs can be identified. They can either be replaced by existing similar features, modified in a manner that allows a better fit to the SPL architecture, or can possibly be abandoned completely. If a feature is important for certain customers, it must be examined up to which extent the customers can be charged with the real costs.

In all cases the importance of a feature or customer for the market strategy has to be considered. It can be a reasonable decision to take a loss in order to open or develop a market. But to decide on the strategy, there

Software Product Management, 2006. IWSPM '06. International Workshop on , vol., no., pp.13-18, Sept. 2006
doi: 10.1109/IWSPM.2006.1

must be some estimation where and how much money is lost.

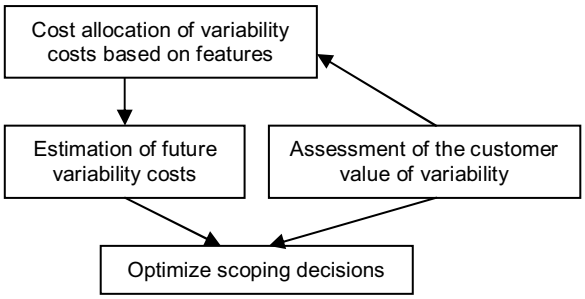

Figure 2: Approach to variability management

4.4. Process improvement

Cost information will not only help to decide on the range of offered features and their pricing. It can also be utilized to identify weaknesses in the internal processes and guide respective improvements. As an example it might be identified that there is a high effort for testing certain features in each subsequent product release. This can justify investments in flexible testing technologies. Other examples might be a high effort for customer support or for bug fixes related to certain features.

5. Further work

In our further work we aim at developing a detailed model for a costing approach based on features as cost objects. Within the context of an ongoing industry cooperation project we will conduct a case study on gathering cost information in an SPL development process, in order to validate if this approach can help to evaluate product management decisions and support cost estimations. To provide appropriate tool support, the collection of costing information should be integrated with our existing tool set for feature modeling [11].

6. Summary

Managing a software product line imposes novel challenges to software product management due to multiple stakeholders with diverging needs, and complex dependencies between the products. The cognitive complexity of the variability of features leads to high coordination efforts, insufficient scoping decisions and a suboptimal exploration of the reuse potential. Difficulties in steering the increase of feature variants within the product line are evidenced by increasing maintenance costs.

This resembles the significant increase of overhead costs caused by the growth of product variants in manufacturing industry. Variability management was therefore recognized as a core task of product management. A key to gain control on product variability was the application of new costing systems to obtain more accurate cost information.

We believe that intransparent cost structures are a major barrier for successful management of software product lines. The presented approach to variability management for software product lines aims at closing this gap by utilizing an activity based costing approach to software development with features as primary cost objects. Features provide a natural basis for allocating costs and are anchored in existing variability modeling techniques. The gathering of more accurate cost information is combined with an assessment of the customer value of features within the product line. This approach can support better estimations of future costs of variability as a basis for customer negotiations and enable a strategic steering of the variability within the product line.

References

[1] Schmid, K. (2003): Planning Software Reuse – A Disciplined Scoping Approach for Software Product Lines. PhD Theses in Experimental Software Engineering, vol 12, Fraunhofer IRB Verlag, Stuttgart.

[2] Clements, P.C., J.D. McGregor, S.G. Cohen (2005): The Structured Intuitive Model for Product Line Economics (SIMPLE). CMU/SEI-2005-TR-003, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania.

[3] Schuh, G. (2005): Produktkomplexität managen – Methoden, Strategien, Tools. (in German) Hanser Verlag, München.

[4] Kaplan, R.S., R. Cooper (1997): Cost and Effect. Harvard Business School Press, Boston, Massachusetts.

[5] Fichman, R., C. Kemerer (2002): Activity Based Costing for Component-based Software Development. Information Technology and Management, vol 3 (1/2), 137-160. Springer, Netherlands.

[6] Kang K., S. Cohen, J. Hess, W. Novak, A. Peterson (1990): Feature-Oriented Domain Analysis (FODA) Feasibility Study. CMU/SEI-90-TR-021, ADA235785, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania.

[7] Lee, K., K.C. Kang, J. Lee (2002): Concepts and Guidelines of Feature Modeling for Product Line Software Engineering. In Gacek C. (Ed.) Proceedings of the 7th International Conference on Software Reuse: Methods,

Software Product Management, 2006. IWSPM '06. International Workshop on , vol., no., pp.13-18, Sept. 2006
doi: 10.1109/IWSPM.2006.1

Techniques, and Tools, ICSR-7, Austin, Texas. 62-77. LNCS 2319, Springer, Berlin.

[8] Kano, N., N. Seraku, F. Takahashi, S. Tsuji (1996). Attractive quality and must be quality. In J. D. Hromi (Ed.) The best on quality (Vol. 7). 165-186. ASQ Quality Press, Milwaukee, Wisconsin.

[9] Akao, Y. (1988): Quality Function Deployment QFD: Integrating Customer Requirements into Product Design. Productivity Press, Portland, Oregon.

[10] Helferich, A., G. Herzwurm, S. Schockert (2005): QFD-PPP: Product Line Portfolio Planning Using Quality Function Deployment. In Obbink, H., Pohl, K. (Eds.) Proceedings of the 9th International Software Product Line Conference, Rennes. 162-173. LNCS 3714, Springer, Berlin.

[11] von der Maßen, T., H. Lichter (2004): RequiLine: A Requirements Engineering Tool for Software Product Lines. In van der Linden, F. (Ed.) Software Product-Family Engineering, 5th International Workshop, PFE 2003, Siena. 168-180. LNCS 3014, Springer, Berlin.

Software Product Management, 2006. IWSPM '06. International Workshop on , vol., no., pp.13-18, Sept. 2006
doi: 10.1109/IWSPM.2006.1