Applying architecture preservation core for product line stretching

Applying Architecture Preservation Core for Product Line Stretching

Yanja Dajsuren, Mark van den Brand

Eindhoven University of Technology, 5612 AZ Eindhoven, The Netherlands

Abstract

The product line engineering approach is receiving a broad interest to decrease the cost of development and time to market, and to increase product quality as software becomes more and more important for companies in all markets. Although a significant amount of research has been done to define a method for introducing product line engineering in an organization, these methods are limited when a product line stretches over time. When stretching a product line, the evolution of a product line and its products may require fundamental change of the software architecture and consequently result in discontinuous evolutions. In this paper, we discuss the issues of software architecture with respect to discontinuing evolution and present an economic model based on the architecture preservation core concept to influence the product line stretching decision.

Keywords: architecture preservation core, product line evolution, product line stretching

1. Introduction

The Software Product Line (SPL) [1] approach is getting more attention and adoption by companies in all markets such as Nokia, Hewlett-Packard, LSI Logic, Philips, and Cummins. These companies have improved time-to-market, engineering costs, portfolio size, and defect rates by factors of 3 to 50 using SPL techniques [2].

Several architecting methods for single-systems such as Attribute Driven Design [3], Bosch [4] and Bredemeyer [5] have been applied for creating product line architectures as highlighted in [6]. However, these methods do not take into account the case when a product line stretching occurs, which may introduce unforeseen requirements.

One of the challenging decisions for a product line strategy is determining the product line length - the number of products in the product line. The product line length is considered too short if profits can be increased by adding items; the line is too long if dropping items can increase profits [7]. Although company objectives and resources influence the product line length, over time new products tend to be added to the product line due to a number of reasons such as to satisfy customers needs for more complete product line or to increase sales and profits. One of the common approaches to increase the product line length is by stretching a product line. Thus, *product line stretching* occurs when a company increases the product line by lengthening it beyond its current range [7].

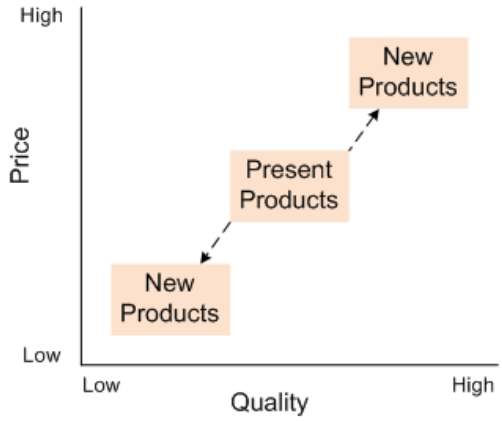

Figure 1: Product line two-way stretching

The company can stretch its product line downwards, upwards or both ways [7]. Many companies initially locate at the upper end of the market and later stretch their lines *downwards*. Many reasons can lead to this decision e.g. to fill a market hole or to avoid new competitors. For example, Compaq and IBM added less expensive personal computer lines to prevent competition from low-priced 'clones' and to take advantage of faster market growth in the lower

Email addresses: y.dajsuren@tue.nl (Yanja Dajsuren), M.G.J.v.d.Brand@tue.nl (Mark van den Brand)

end of the computer market [7]. Companies at the lower end of the market may want to stretch their product lines upwards. It can, for example, help the company to add prestige to their current products. One of the examples is that Lexus, a luxury vehicle offering both exceptional performance and quality, introduced by Toyota to address the higher end of the market. Companies in the middle range of the market may also decide to stretch their lines in both directions as shown in Figure 1. Texas Instruments introduced its first calculators with the medium price and medium quality. It gradually added calculators at the lower end taking the share from Bowmar and at the higher end to compete with Hewlett-Packard. This two-way stretch won Texas Instruments an early market leadership in the hand-calculator market [8].

When stretching a product line, the evolution of a product line and its products may require fundamental change of the software architecture and result in *discontinuous evolution dynamics*, which is discussed in Section 2. The main contributions of this paper are that it addresses a lack of connection between product line engineering and product line stretching strategy and proposes using an economic model to influence the decision of product line stretching. *Architecture preservation core* concept is used to define the product line architecture and further plays a significant role for evaluating a decision for introducing a product as stand-alone product or a part of a product line.

2. Software architecture with discontinuity evolution dynamics

A software product line architecture is used as a reference architecture to effectively facilitate the commonality and differences of products in the product line. The reference architecture is created during the domain engineering phases. It is used as a common asset in the application engineering phases, where it is instantiated and extended to create product architectures as illustrated in Figure 2 [6]. A number of methods for creating product line architectures have been defined such as COPA [9], FAST [10], FORM [11], KobrA [12], and QADA [13].

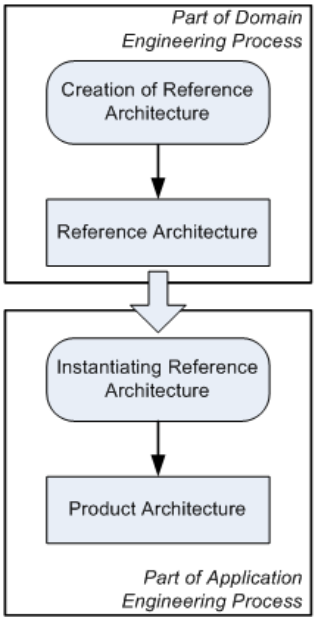

Figure 2: Creating Reference and Product Architecture

Designing a reference architecture is a difficult task because conflicting requirements might emerge when capturing requirements for all the products in the product line [6]. To support all the products in the product line, the reference architecture needs to be designed with the variation points of the product line in mind. Furthermore, a product line and its products evolve over time and may influence one another.

The notion of evolution dynamics (change or growth) was extended with continuous and discontinuous evolutions in [14]. Software evolution is considered *continuous* if a software system preserves most of the aspects when it evolves. When essential aspects of a software system are changed, it is called *discontinuous* software evolution [14]. It was claimed that the discontinuity of evolution includes two types of changes - *architectural* and *feature* change [14]. In the case of architectural change, a software architecture is extended substantially resulting in a significant change of the source code. In the case of feature change, when a large set of new requirements are introduced in the systems, it also results in significant addition and change of the code.

To identify whether the evolution is discontinuous, a set of concepts and metrics for software architecture evolution based on the similarity of tree structures were introduced in [15]. One of the main concepts is an *Architecture Preservation Core* (APC), which is a part of the software architecture that is preserved when evolution occurs. If a change impacts the APC, it is considered discontinuous evolution as it requires a change of the entire software architecture. Cost of changing the APC is higher than changing the other parts [15].

3. Economic model for software product line stretching

Product line engineering plays an important role for the business performance of companies as it provides a common platform for a set of products mostly addressing a whole market. The product line platform needs to be well defined for supporting the business strategy by enabling development of products that fit the platform. Therefore,

having a strong link between product line engineering and market strategy can enable successful business performance of a company [6]. Scoping approaches is used for integrating technical and marketing-driven product line planning e.g. a product portfolio scoping approach determines the range of products for the product line mainly driven from market inputs [6]. Furthermore, an economic model for product line development can be used for the decision making process when introducing or evolving a product line.

Bockle et.al. [16] have introduced a first-order software cost model in the context of product line engineering. It is based on a set of product line scenarios addressing development cost aspects of starting, running, adding to, merging, splitting, or retiring a product line. The cost of developing a software product line is expressed as following and used in the formulation for each scenario in terms of the cost model [16]:

$$C_{org} + C_{cab} + \sum_{i=1}^{n}(C_{unique}(p_i) + C_{reuse}(p_i)) \tag{1}$$

where, C_{org} is the cost to an organization of adopting the product line approach such as reorganization, process improvement, and training. C_{cab} is the cost to develop a core asset base of the product line. C_{unique} is the cost to develop unique software that is not based on a product line architecture. C_{reuse} is the cost to reuse core assets in a core asset base.

Currently, there is no explicit connection established between product line engineering and product line stretching. Therefore, we use the economic model of Bockle to assess whether it is worth to stretch an existing product line or to introduce in the market as a stand-alone product. If we stretch product lines downwards after locating at the upper end of the market, it implies that the price and quality of present products need to be lowered as illustrated in Figure 3. Functional and qualitative features can be used to describe a product [1]. We assume the development cost plays a crucial role for defining a product's price, thus implying here the development cost as price.

Stand-alone case for stretching would cost:

$$\sum_{i=1}^{m} C(p_i) \tag{2}$$

where, p is a product.

Since new products are based on existing product line, the following equation gives the cost of stretching the existing product line by m number of products:

$$C_{cab} + \sum_{i=1}^{m} C_{unique}(p_i) + \sum_{i=1}^{m} C_{reuse}(p_i) \tag{3}$$

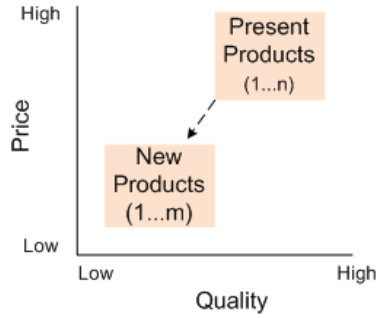

Figure 3: Product line downwards stretching

As an organization has already adopted the product line approach, the C_{org} is not necessary and the cost of reuse is required for every new product. Comparing these equations supports the decision for introducing new products either as part of product line stretching or as stand-alone products.

4. Architecture preservation core of product lines

As stated in [17], the cost estimation accurateness of developing the product line depends on the measures for implementing each of the cost functions. We focus on elaborating the cost function of a core asset base C_{cab}.

$$C_{cab} = C_{cva} + C_{refarch} + C_{swdev} \tag{4}$$

where C_{cva} is the cost of executing a commonality/variability analysis, $C_{refarch}$ is the cost of designing and evaluating a reference architecture (generic architecture), C_{swdev} is the cost of developing the designed software. We investigate further the $C_{refarch}$.

The software architecture is modeled as a tree (preordered), where the nodes and edges represent components and connectors respectively [15]. Figure 4, adapted from [15], illustrates the software architecture evolution based on the tree model. The evolution is defined as a set of collective changes made on the tree by adding, modifying, and removing nodes and edges, while preserving existing nodes and edges as well [15].

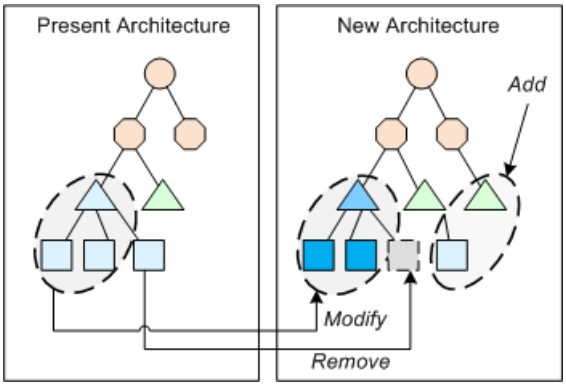

Figure 4: Software architecture evolution

Aoyama proposed a set of distance metrics for software architecture evolution based on the rewriting operations (adding, modifying, and removing operations) on the trees.

The equation for the cost of a set of changes is defined as [15]:

$$C_0 = W_A \times \sum_i A_i + W_M \times \sum_j M_j + W_R \times \sum_k R_k \quad (5)$$

where, A_i, M_j, and R_k denote the cost of Add, Remove, and Modify operation respectively, and W_A, W_M, and W_R denote the weight of the costs.

As mentioned earlier, the cost of changing architecture core is expected to be higher than changing other parts. So, the cost equation considering the architecture weights (by means of architecture preservation cost) is defined by equation:

$$C_1 = (1/2^d)[W_A \times \sum_i A_i^d + W_M \times \sum_j M_j^d + W_R \times \sum_k R_k^d] \quad (6)$$

Note that A, M, R respectively denotes the cost of Add, Modify, and Remove operation at depth d of an architecture tree model.

Now the product stretching can be planned considering the total cost of architecture core for all products of the product line:

$$C_{refarch} = \sum_{ii=1}^{n+m}((1/2^d)[W_A \times \sum_i A_i^d(p_{ii}) + W_M \times \sum_j M_j^d(p_{ii}) + W_R \times \sum_k R_k^d(p_{ii})]) \quad (7)$$

When the product line stretching occurs, not only the present products are changed to new products, it may also require a change of reference architecture. Having $C_{refarch}$ estimated is a valuable input for a product line stretching decision. However, the situation with respect to precise estimation is rather limited as we adopted a simplified tree model for a software architecture with a number of assumptions at hand.

The cost of building assets for product line reuse is higher than building assets for single system development [18]. However, it enables much more efficient production of the individual systems [6]. Therefore, other aspects (e.g. quality requirements) further need to be addressed in the economic model to influence the product line stretching strategy.

5. Concluding remarks

Companies in all markets are seeking more efficient ways to develop and maintain software systems. One of the approaches receiving broad attention is software product line engineering. If a company succeeds in adopting a software product line approach, it benefits from developing low cost, but high quality software effectively. A scoping approach is used for integrating technical and marketing-driven product line planning, which determines the product line range and defines precise functionality of reusable components.

However, when a company plans to stretch its product line, there is no strong link between the product line engineering and stretching strategy. Therefore, we propose to use the economic model of Bockle to assess whether it is valuable to stretch an existing product line or introducing a stand-alone product in the market. For making the estimation more concrete from the software architecture point of view, the architecture preservation core concept is used to define the total cost of changing the architecture core for all products of the product line. The proposed method can be extended further with a cost model that considers quality related attributes for the architecture core.

References

[1] G. B. Klaus Pohl, F. van der Linden, Software Product Line Engineering: Foundations, Principles, and Techniques, Springer-Verlag, Berlin Heidelberg, 1st edition, 2005.

[2] Software production lines, http://www.softwareproductlines.com/, 2007.

[3] L. Bass, M. Klein, F. Bachmann, Quality attribute design primitives and the attribute driven design method, the 4th International Workshop on Product Family Engineering (2004).

[4] J. Bosch, Design and use of software architectures (2000).

[5] Software architecture workshop, course handouts, http://www.bredemeyer.com/, 2002.

[6] F. J. van der Linden, K. Schmid, E. Rommes, Software Product Lines in Action, Springer, New York, NY, 1st edition, 2007.

[7] P. Kotler, G. Armstrong, J. Saunders, V. Wong, Principles of Marketing, Pearson Education Limited, Italy, 3rd edition, 2001.

[8] Line stretching in marketing, http://www.citeman.com/1290-line-stretching-in-marketing/, 2006.

[9] P. America, H. Obbink, J. Muller, R. van Ommering, Copa: A component-oriented platform architecting method for families of software intensive electronic products, The First Conference on Software Product Line Engineering (2000).

[10] D. Weiss, C. Lai, R. Tau, Software product-line engineering: a family-based software development process (1999).

[11] K. C. Kang, S. Kim, J. Lee, K. Kim, E. Shin, M. Huh, Form: A feature-oriented reuse method with domain-specific reference architectures, Annuals of Software Engineering (1998) 143 – 168.

[12] C. Atkinson, Component-based product line engineering with uml (2002).

[13] M. Matinlassi, E. Niemel, L. Dobrica, Quality-driven architecture design and quality analysis method, a revolutionary initiation approach to a product line architecture, VTT Technical Research Centre of Finland (2002).

[14] M. Aoyama, Continuous and Discontinuous Software Evolution: Aspects of Software Evolution across Multiple Product Lines (2001).

[15] M. Aoyama, Metrics and Analysis of Software Architecture Evolution with Discontinuity (2002).

[16] J. D. M. D. M. Günter Böckle, Paul Clements, K. Schmid, Calculating ROI for Software Product Lines (2004).

[17] J. D. M. D. M. Gnter Bckle, Paul Clements, K. Schmid, A Cost Model for Software Product Lines (2003).

[18] K. Schmid, Planning Software Reuse A Disciplined Scoping Approach for Software Product Lines (2002).