Finalizing a PhD Thesis in Architectural Evolution

Josef Nedstam
Department of Communication Systems
Box 118, SE-221 00 Lund, Sweden
josef.nedstam@telecom.lth.se

Abstract

This Student Paper describes work in progress within the field of architectural evolution. The research done for this PhD Thesis has involved study of individual architectural changes, a view that is now integrated to study how companies have evolved with their software architectures. Participation in EDSER will provide opportunity to discuss the relation between an organization's business model and software architecture; and to discuss how architectural initiatives are best funded and organized.

1. Introduction

An architectural approach to reuse has emerged in literature, with technical and also managerial aspects of architecture receiving attention. Two examples are the Product Line Practice initiative by SEI [1], and the framework of maturity levels for software product lines proposed by Bosch [2]. SEI's initiative can guide companies on how to implement a software product line, while Bosch extends this evolutionary view by providing guidelines for which assets and organizational entities must be in place to successfully reach any state in his framework.

Companies will have different business goals, will have different resources to fulfill these, and will have to do so under different external circumstances. The view of the author of this paper is therefore that companies may require different architectural strategies. For example, Kruger [3] makes the case for a product line explicit, when saying that the objective of a software product line is to optimize software engineering efficiency by exploiting commonalities among products. However, a company with other objectives may have to employ different architectural approaches. The underlying assumption of this paper is hence that any particular company will at any given time be favored by a particular architectural approach. The research goal of this work is to provide guidelines and recommendations for companies on how to determine their most suitable architectural approach, and how to implement this approach.

Current work on this PhD Thesis has yielded preliminary results from a study of how 13 companies have evolved along with their software architectures. These results are in the form of a framework for architectural evolution [4], which extends Bosch's framework for software product line maturity. This framework now has to be finalized by analyzing the business motives for architectural change and evolution, and on a more concrete level how architectural initiatives should be funded and managed.

A qualitative approach has thus far been taken to this research [5], as it is exploratory in nature. Most of the work has been performed in the form of open ended interviews in a workgroup focusing on the concept of platforms, hosted by SPIN-Syd [6], a Swedish node of the Software Process Improvement Network.

2. A Framework for Architectural Evolution

Results from the interview material analysis were mapped to Bosch's framework. As a result new states and transitions had to be included in the framework. This section gives an overview of these results, and also presents findings regarding linkage between business goals and architectural initiatives, and how the participating companies funded their initiatives.

2.1 States of Architectural Evolution

The maturity levels for software product lines that Bosch presents are a useful framework for companies to employ in order to increase their product line maturity. All companies in this study did not, however, see this as their business goal. When their evolution was compared to the framework, additional states were identified in the framework, in order to properly describe the options a company can have. The reworked framework is shown in Figure 1, consisting of the following states:

- **Independent products**. Products are stand-alone, without any reusable assets.
- **Standardized infrastructure**. Products are built on a standardized, externally developed generic infrastructure, such as an operating system, middleware infrastructure or database manager.
- **Internal platform**. Products are based on an internally developed and managed domain specific platform.
- **Platform as product**. The company markets this domain specific platform as a product.
- **Platform customer**. This state is similar to *Standardized Infrastructure*, but implies domain specific platforms, rather than generic infrastructures.
- **Software product line**. The platform also includes functionality that is not used by all products, and several products within the product line are marketed simultaneously.
- **Configurable product base** and **Configurable product base (unmanaged)**. Products are developed by configuring a product base, which can either be adhering to an enforced architecture, or not.
- **Consultants**. This was the initial state of some of the studied companies, and not a proper architecture strategy.
- **Consecutive releases from stable architecture**. One company in particular was able to produce new versions of their product on a regular basis, based on the previous version, but without clearly visible architectural assets.

One company also decided to develop a *platform as product* from startup, a state which is not depicted in Figure 1, although it is very similar to *Consultants* in this framework.

2.2 Transitions between Architectural Approaches

Figure 1 also shows the transitions between states encountered in this work. These are presented here:

- **New product generation**: *Consecutive releases from stable architecture → Independent products*. One company was able to develop new versions of their product at regular intervals for an extended period of time. Eventually the architecture of the product did not support new market requirements, and the company had to develop a new product generation, i.e. go back to *Independent Products*.
- **Company split along platform interface**: *Internal platform → Platform as product & Platform customer*. A company struggling with the balance of distributing resources between the internal platform and the product oriented projects decided to split in two, where the company that supplied the platform was free to sell it to other customers.

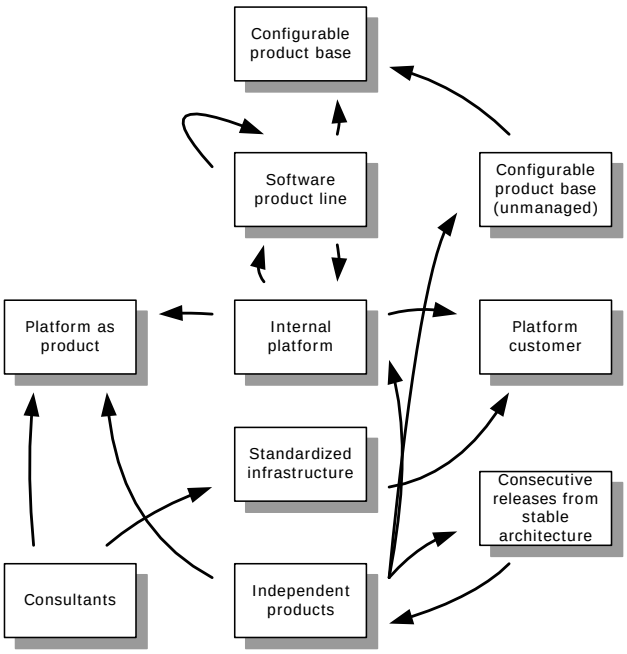

Figure 1. States of architectural evolution

- **Packaging consultancy knowledge as product**: *Consultants → Independent products*. One consultancy company saw the opportunity of packaging their domain knowledge into a product, and could do so by an injection of venture capital.
- **Generalizing product into Platform as product**: *Independent products → Platform as product*. The company mentioned in the previous bullet realized that their domain knowledge was more suitable for a platform than a proper application, and generalized their product into an application server.
- **Generalizing product into Internal platform**: *Independent products → Internal platform*. Since the workgroup was focused on platforms, this was the most common transition. The company that split did this as their first step; another company implemented a GUI framework to support multiple operating systems; and in other examples the platform was the initial step to implement a product line.
- **Packaging consultancy knowledge as Platform as product**: *Consultants → Platform as product*. In one case a consultancy company decided to take the step directly to a platform packaged as a product.
- **Startup platform as product**. A similar transition was performed by a startup company, whose first product was a platform packaged in a development tool. Their business idea is that it is not feasible for companies to fund internal platforms, but should package them as products or acquire them from external sources.
- **Increased scope leading to product line**: *Internal platform → Software product line*. Companies in

the study in some cases found opportunities to differentiate their product portfolio based on the platform they had, leading them into a software product line. This transition also appears in Krueger's taxonomy of software product lines [3].

- **Decreased scope for existing product line**: *Software product line → Internal platform*. One of the product line companies later reduced their scope, and might now have too extensive architectural assets.
- **Outsource existing IT resources**: *Standardized infrastructure → Platform customer*. One organization decided to reduce risk by outsourcing their IT resources, and becoming a customer rather than a developer for internal use.
- **Synchronization between applications and platform in product line setting**. This is a constant evolution in the *Software Product Line* state, where development of architectural assets has to be synchronized with development of products from these assets, and a cause of constant frustration for managers of these assets. Krueger [3] refers to this transition as *enhancement*, and allows for variation in architectural maturity within the *Software product line* state.
- **From contractual development to off-the-shelf products**: *Independent products → Configurable product base (unmanaged)*. Products developed under contract are often independent from each other since there is no incentive for managed reuse. The company in this situation has a product that can be made into an off-the-shelf product, if the proper generalizations are made.
- **Packaging project-internal platform without organizational support structure**: *Independent products → Internal platform*. The developers in the aforementioned case have themselves implemented a framework capturing domain knowledge, which could be the base for such generalizations, but no organization exists to manage such assets.

2.3 Relating Architectural Approaches to Business Goals

The study has included architectural initiatives implemented in response to changing business strategies, implemented as integral parts of business strategies, and initiatives that have been more or less unrelated to any business decisions. None of these scenarios can be said to be better or worse. Developers have to make technical decisions, and should pursue opportunities of cost reductions and reuse. Managers on the other hand should pursue opportunities of mergers, acquisitions and entering new markets, ventures that often will have technical impact. However, increased awareness and communication between the managerial and technical side of the companies in this study could in many cases have led to both management decisions that were more aligned to the capabilities of the development department, and developer decisions that were more targeted at the current business strategies. These issues have been affirmed by Hohmann [7] and Faulk et al [8]. Further work from these initial results should therefore focus on finding the links between business goals and strategies, over quality attributes, to not only architectural strategies, but also to ways for how to organize development around selected strategies.

2.4 Funding and Resources

The major development management difficulty in most of the cases under study has been to find the balance between investments in reusable assets and investments in products sold to customers, i.e. the balance between long term and short term investments. The problem is one of funding, but also has other important aspects such as resources, organization, and how to synchronize work.

In their framework for software product line practice, the SEI presents 9 strategies for funding product line activities, along with guidelines for their appropriateness. Some of these were identifiable and applicable to the studied cases. The collected material includes some interesting patterns that could extend the SEI guidelines:

- Architects often found it very easy to get funding for initiatives related to growth strategies. Cost-saving initiatives were on the other hand harder to fund – such funding was especially sensitive if the initiative would delay release of products, and therefore delay short term revenues.
- Venture capitalist funding gives more liberties for long term investments, compared to being funded by sales and customer projects.
- How does a customer specific project determine the value provided by using the architectural assets? Three approaches not conforming to the traditional product line approach were found among the studied cases: the company which split in two, enabling the market to set the price of the platform; the company that from startup decided to build a platform packaged as a product, since their business proposal assumes that it is too difficult for companies to manage development of internal platforms; and the organization that instead decided to outsource most development. An important lesson they learnt was that requirements engineering capacity cannot be outsourced.

Independent of funding strategy, the companies still had organizational and resource difficulties. The con-

tract driven company had no organization that could package the developed framework in order for it to be reused internally. The company that had the most mature software product line in this study had a constant problem of balancing resources between reusable assets and product-specific assets: product projects would not wait for new releases of reusable assets, but rather do their own implementations of common but not yet developed functionality. One solution to this problem was to not have a separate development unit for reusable assets. Two of the companies only used what Bosch calls *development departments* that were mainly responsible for product development, but would evolve architectural assets when necessary or between projects; they were therefore not using the suggested *domain engineering unit*, to avoid synchronization problems. One company that did have a *domain engineering unit* constantly had some of its members as apprentices on the product-developing projects, to simplify requirements elicitation for the reusable assets and set the correct expectations on these assets among other developers.

2.5 The Case for an Architectural Investment

A problem for initiators of architectural initiatives is to make the business case for it [9]. ROI is often said to be the metric to focus on when making such decisions, and this might be true if the initiator or decision maker is a product manager [10]. The priorities of a project manager would on the other hand be to stay on time and on budget, while the role of the architect seldom is tied to any such particular responsibility. Other metrics, such as time to break-even, as discussed by SEI, or market share, might be more suitable cornerstones of a business case for any architecture initiative. As previously stated, the goal of the initiative will determine how to make the business case; a business case for growth will be very different from a business case for cost reduction. None of the development level initiators in this study made a formal business case, or even produced any figures supporting the initiatives.

3. Discussion

A PhD Thesis based on these and previous results would focus on describing the process of architectural evolution, in order to guide companies and individual architects in how to apply the benefits of an architectural approach to reuse. In relation to the EDSER workshop, this would involve more work on analyzing the relation between business goals and strategies, and architectural initiatives. This would also on a more detailed level involve analysis of the most preferable ways to make business cases for different types of architectural initiatives.

The work is currently geared towards finding a link between overall business goals, and the architectural approaches that suit these goals best. A suggested position statement is under development: A company needs or wants a *sustainable competitive advantage* [11], which is based on the way the company competes (strategies, tactics), its basis for competition (resources, competencies), the market it competes in, and the competitors in that market. A strategy for sustainable competitive advantage can be derived from analysis of these factors, and part of that strategy is the *positioning strategy*. If this positioning strategy is known to all parts of the organization, and aligned to available resources, competencies and architectural assets, an architect should be able to determine the driving quality attributes for the products being developed. These quality attributes then determine which architectural approaches are most favorable.

In order to conclude this PhD Thesis, some additional special cases of software evolution might have to be studied, but the bulk of the work is in analyzing available information, and subjecting the conclusions to the scrutiny of peer researchers. EDSER participation is hoped to provide just that type of discussion.

References

[1] Clements, P., Northrop, L., Software Product Lines: Practices and Patterns, Addison-Wesley, 2002

[2] Bosch, J., “Maturity and Evolution in Software Product Lines: Approaches, Artefacts and Organizations”, SPLC2, San Diego, Cal., 2002

[3] Krueger, C. W., “Towards a Taxonomy for Software Product Lines”, 5th Intl. Workshop on Product Family Engineering, Siena, Italy, November 2003

[4] Nedstam, J., Karlsson, E.-A., ”Experiences from Architectural Evolution”, AWSA’04, Melbourne, Australia, 2004

[5] Patton, M. Q., *Qualitative Evaluation and Research Methods,* 2nd Ed., Sage Publications 1990

[6] SPIN-Syd, http://www.spin-syd.org

[7] Hohmann, L., *Beyond Software Architecture: Creating and Sustaining Winning Solutions*, Addison-Wesley 2003

[8] Faulk, S. R., Harmon, R. R., Raffo, D. M., “Value-Based Software Engineering (VBSE): A Value-Driven Approach to Product-Line Engineering”, SPLC1, Denver, Colorado, 2000

[9] Nedstam, J., Karlsson, E.-A., Höst, M., ”Experiences from the Architectural Change Process”, STRAW’03, Portland, Oregon, 2003

[10] Nejmeh, B. A., Thomson, I., “Business-Driven Product Planning Using Feature Vectors and Increments”, IEEE Software, November/December 2002

[11] Aaker, D. A, *Strategic Market Management,* 6th Ed., John Wiley & Sons, 2001