

# Experiences from Architectural Evolution

Josef Nedstam

*NICTA-ESE*

*Locked Bag 9013, Alexandria, NSW*

*josef.nedstam@nicta.com.au*

*Department of Communication Systems*

*Box 118, SE-221 00 Lund, Sweden*

Even-André Karlsson

*Q-Labs*

*Ideon, SE-223 70 Lund, Sweden*

*even-andre.karlsson@q-labs.se*

## Abstract

*Frameworks for describing a company's architectural maturity have in recent years emerged in literature. These are often biased towards the particular flavor or architectural concept which is the point of interest of the specific author, be it component-based development, platforms, product lines or highly configurable code bases. They can therefore be valid for helping a company reach one particular level of architectural maturity. What they do not provide are guidelines for which such maturity levels are suitable for companies in particular situations.*

*The authors of this paper acknowledge that different companies have varying needs, and that the business situation and business goals of a company will determine what architectural maturity level is more suitable for that company. In this work the authors have studied architectural situations at twelve companies in order to determine why and how these companies have moved between maturity stages. We attempt to create a general framework for describing these different stages and giving guidelines for companies on how to traverse the maze of architectural evolution.*

## 1. Introduction

An architectural approach to reuse has emerged in literature. This has gradually meant that not only technical but also managerial aspects of architecture have received some focus. 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 into implementing a software product line, while Bosch extends this evolutionary view by providing several maturity levels with guidelines for which assets and organizational entities must be in place to successfully reach a higher state of software product line maturity.

However, companies will have different business goals and strategies; will have different resources to fulfill

goals and carry out strategies; and will have to do this under different external circumstances. The underlying assumption of this paper is therefore that each particular company will at any given time have a preferred maturity state which it should aim for. The research goal of this work is therefore to provide guidelines and recommendations for companies on how to determine their target maturity state, and how to go about reaching this target. This paper presents a first attempt at reaching the overall research goal by exploring how architectures evolve, and how this evolution relates to the business goals of companies.

Most of this work has been performed in a workgroup focusing on the concept of platforms, lead by SPIN-Syd [3], a Swedish node of the Software Process Improvement Network.

## 2. Method

The field of study, architectural evolution, involves extended chains of events, where each chain of events occurs rarely. It is difficult to identify independent and dependent variables, and it is not possible to separate the context from the instances of the phenomena under study. The research has therefore been carried out in a qualitative fashion [4], rather than in quantitative terms.

Information for this study has been gathered from architectural initiatives conducted, or being conducted, at twelve companies. The participants of the SPIN workgroup in platforms represented eight of these, and the focus for initial interviews was jointly discussed during three sessions in the workgroup. These interviews were to focus on current or recent architectural initiatives such as introduction of platforms to the participating companies. The interviews were held to provide a description of the architectural initiative and its relation to the business model of the company, in relation to the evolution of the company and its products, issues itemized in an interview guide. The interviewees from the eight SPIN companies were in most cases initiators of the initiatives, but their roles in the companies varied from developers to top

management. The interviews were open ended, based on the interview guide. The guide was also updated between interviews. Information from another four companies was extracted from a previously conducted study [6].

In order to draw qualitative conclusions from the information, it was analyzed to find dimensions, which were classified and given values [4]. The significant dimensions that emerged resulted in a table where the information from each company was decomposed into comparable fields. The classification of each field was not done in a strict, multiple-choice fashion, so that the participants could give feedback on both the facts of their companies, and on the discovered dimensions. This feedback also made it possible to fill in blanks in the table.

### 3. An Architectural Evolution Framework

In this work we have had practical use of the framework of maturity levels for software product lines proposed by Bosch [2]. The results of the work have been an adaptation of this framework. By classifying the material according to the identified dimensions, we have found new states and possible transitions in the framework. This section presents these results.

#### 3.1 Dimensions from the Material

The information gained from interviews was roughly decomposed into the following dimensions:

- **Product, service:** A description of what the company supplies to its customers, which in some cases is the platform or architectural initiative itself. The products studied have ranged from software development tools and information systems to control systems and consumer electronics.
- **Business strategy:** This includes value proposition, sales strategy and value generation, in order to see if the architectural initiative was aligned to the business strategy.
- **Platform:** A description of the architectural initiative.
- **Goal and role of platform:** This can be an integral part of the business strategy, or a technical implementation to fulfill specific quality attributes. The material includes initiatives that have been integral parts of companies' businesses and even startup companies where the architecture initiative was at the core of the business idea, but we have also seen platforms and frameworks that have emerged from technical decisions by individual developers.
- **Quality attributes:** Describes the quality attributes the platform aims at. Most initiatives aim at lowering cost through reuse, simplified change management, increased maintainability or increased understandability, but others have aimed at increasing variability and feature content.
- **Evolution:** The evolution the business and the architectural initiative have gone through, and how the com-

pany has balanced its long-term and short-term focus. Analysis of these issues is at the core of the contribution this paper makes to the framework presented by Bosch, and is further discussed in Section 3.2 and 3.3.

- **Organizational structure and funding:** How the organization has been structured around the architectural initiative and how funding and resources have been distributed between the platform and the products developed from it. The material includes companies that have tried to manage organizational units for the platform, or architectural asset, separate from product development, and have had quite different approaches to this. Bosch calls this organizational structure a *Domain Engineering Unit*. To solve the problem of synchronizing resources and scheduling between architectural assets and product developing projects, one company split in two along the interface of the platform, so one half now provides the platform as their product. Another company was started with the intention of selling a platform as a product, as they consider it too hard for companies to develop internal platforms. Other companies just have one *Development Department*, or a *Business Unit*, for each product.
- **Scope and variation:** The technical parts of the platform describe how much of the products the platform covers, what type of domain it covers, and how the platform accommodates variation in the products. The scope has varied from specific domain knowledge packaged in a blood analysis platform, to general GUI frameworks. The companies have managed variability in different ways; by using configurable product bases managed to various degrees; by performing customer projects that implement the required variability; or by focusing on one customer and after such a project refactoring the generic parts of the resulting product.

#### 3.2 Maturity States in Architecture 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 order to properly describe the options a company can have. In this discussion, and in Figure 1, we have omitted *Product Population*, a state with high component development maturity but with less focus on an enforced architecture, and *Program of Product Lines*, neither of which we encountered. States that were identified in the material were:

- **Independent products.** Here each project developing a product will stand on its own, without relying on any architectural assets.
- **Standardized infrastructure.** Bosch makes the distinction between a platform and a standardized infrastructure, where the infrastructure is the selected oper-

ating system, database manager, or any other generic external software.

- **Internal platform.** In Bosch’s original framework this state is just labeled *Platform*, a description that was not adequate to describe our material. We have therefore relabeled it to signify companies developing a platform intended for internal use.
- **Software product line.** Bosch makes the distinction between *Platform* and *Software Product Line* when the platform also includes functionality that is not used by all products. This state was identifiable in our material, with the additional requirement that the company develops several products that are marketed simultaneously, along the lines of the SEI definition of a software product line.
- **Configurable product base.** In Bosch’s model this is considered a more mature state than *Software Product Line*. The material therefore led us to include a similar stage, *Configurable product base (unmanaged)*.
- **Consultants.** The material includes consultancy companies, some of which have packaged their consultancy knowledge in products of various kinds, and we have therefore included this state in the framework.
- **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. The architecture of the initial version of the product would allow such development for several releases, which made us include this state.
- **Platform customer.** This state can in Bosch’s terminology be interpreted as *Standardized Infrastructure*, but we acknowledge a difference between standardizing an operating system or communication middleware, and acquiring a domain specific platform which is turned into a product by configuring variation points, rather than developing code. In one case we studied just such a company; in others we studied companies that had customers of this category.
- **Platform as product.** The material includes companies that have packaged their platform as a product while still using it internally, companies that have divested their product development in order to focus on selling the platform as a product, and consultancy companies and startups that have packaged their knowledge in a platform to be sold as a product.
- **Configurable product base (unmanaged).** This state is similar to *Configurable product base*, but companies in this state must run full-scale projects to develop a product from that product base. It would also be possible to merge this state with *Consecutive releases from stable architecture*; our main reason for not doing so is that companies in this category will have to support several releases over an extended period of time.

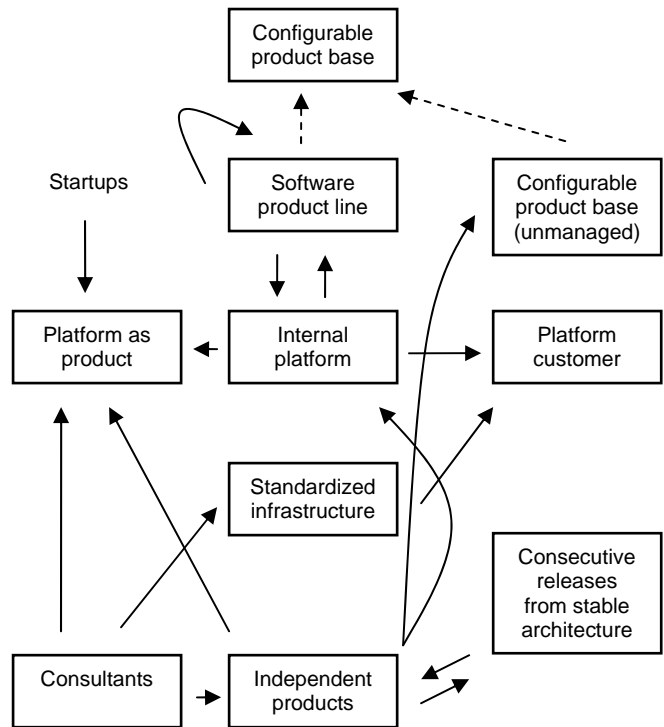


Figure 1. Maturity states in architecture evolution

### 3.3 Transitions in the Architecture Lifecycle

Figure 1 also shows the transitions between states that we have encountered in this work. Although we have removed the direct links from *Independent Products* to *Standardized Infrastructure* and on to *Internal Platform*, as found in Bosch’s original framework, we do believe these to be valid; they just did not show up in our material. The fact that a company can move from one state to another does not provide much information, but the reasons for such transitions do, and here we present some of them:

- **New product generation:** *Consecutive releases from stable architecture – Independent products*. One company was able to develop new versions of their product in a line oriented fashion 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*. This was a costly move as they also had to switch over to project oriented development. This transition is not wanted but sometimes necessary, and needs to be prepared for.
- **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; a novel approach to the balance between architectural assets and products.

- **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 a change in ownership enabled them to generalize their product into an application server. While currently living off funding from customers, they are struggling with the balance of long term and short term focus. Their current approach is to refactor their product after every customer specific project, and to not have dedicated organizational units with only a long term or short term perspective.
- **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. The difficulty was in all cases to get a proper level of funding for the architectural initiative. Introducing a platform in order to save costs seems to be difficult to get funding for, especially if it might mean delaying profits from products.
- **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. The initiative was to be done in parallel with two customer projects, funded by venture capital. It was possible to get funding as the initiative was part of a growth strategy, and not for cost reduction.
- **Startup platform as product.** A similar transition was performed by the startup company in the workgroup, whose first product is 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. As this is a growth strategy, funding for implementation of the software product line is often feasible to get.
- **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. The strategy aims at cost reduction, but it has proven very difficult to estimate such savings. The organization has also realized that the requirements engineering capability cannot be outsourced; they still need to know what they want from their suppliers.
- **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.
- **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.

## 4. Discussion

Establishing a set of architectural evolution state transitions, such as the one presented here, can show industry the existing options, the pros and cons of these, and in which situations each option is feasible. In order for the framework to be usable by industry it needs to be packaged in an explanative theory, rather than in the current descriptive form. Further work also needs to be done to identify what different strategies for funding, organization and management of architectural initiatives exist.

## 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] SPIN-Syd, <http://www.spin-syd.org>
- [4] Patton, M. Q., *Qualitative Evaluation and Research Methods*, 2<sup>nd</sup> Ed., Sage Publications 1990
- [5] Nedstam, J., Karlsson, E.-A., Höst, M., "Experiences from the Architectural Change Process", STRAW'03, Portland, Oregon, 2003