Experiences from the Architectural Change Process

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

Even-André Karlsson
Q-Labs
Ideon, SE-223 70 Lund, Sweden
even-andre.karlsson@q-labs.se

Martin Höst
Department of Communication Systems
Box 118, SE-221 00 Lund, Sweden
martin.host@telecom.lth.se

Abstract

A good software architecture is becoming recognized as a major factor for successful products. There has been much research on the technical aspects of software architecture and it is recognized that the driving requirements for architectures are "non-functional", but few have studied how organizations decide on architectural changes. In this paper we study the topic through several case studies. The changes to the architecture are in all cases changes to the "non-functional" requirements on the system. Issues that we want to evaluate are: when and how is the need for an architectural change discovered; what is the underlying non-functional requirement; who drives the change; how is it prepared and evaluated; and finally, who makes the decision and how is it implemented.

Through interviews with people that have experience from architectural changes we compare the decision process for architectural changes to the ordinary functional requirement change process and the organizational change process. We find that architectural changes have aspects of both functional and organizational changes. An architectural change does not only need to be technically sound, it also needs to be anchored firmly in the organization. This report gives both architects and managers guidelines to balance short-term project goals and long-term organizational goals with respect to architecture.

1. Introduction

Software architecture is becoming a well-established field in technical terms, i.e. the different types of architectures have been characterized [1]; different useful views of the architecture have been described [2, 3]; as well as books covering the whole area, e.g. [4, 5, 6]. However, little research has been done on how decisions on architectural changes are made in organizations.

Architectural changes are often different in nature from other functional changes. They can impact larger parts of the product, they can imply new ways of working, they are often not clearly connected to one customer requirement, and they are often expensive to implement. Functional changes often originate from a customer demand and are the responsibility of a defined role in a company, i.e. product management. Architectural changes, on the other hand, often emerge from various sources, and roles are seldom defined to drive such changes. All these factors imply that they differ from pure functional changes.

The process for taking decisions regarding functional changes and features has received attention in recent years [7, 8]. Software development processes generally support this rather well. When it comes to decisions regarding the software architecture, the architect is often not so well supported, neither for the analysis of the technical impacts nor the organizational aspects of the change.

Since architectural changes have impact on organizations they might be best compared to the organizational change process, as defined by Kotter [9]. Kotter's eight-stage process describes how to prepare an organization for major change, and how to anchor the change in the organization:

1. Establishing a sense of urgency
2. Creating the guiding coalition
3. Developing a vision and strategy
4. Communicating the change vision
5. Empowering employees for broad-based action
6. Generating short-term wins
7. Consolidating gains and producing more change
8. Anchoring new approaches in the culture

These steps will be referred to in the overview of the suggested process for architectural change in Section 3.

This paper examines how several changes to the software architecture have been handled at three software development organizations, and what internal or external forces that drive the need for changes and control which solutions are decided upon. Concretely we have looked at the following questions for each architectural change:

1. What is the architectural change?
2. Why was the architectural change needed?
3. Who initiated it?
4. How was the associated decision made?

Based on the analysis of these questions, the ordinary process for deciding on functional changes, and theories for organizational change, we propose a process for handling architectural changes, which provides guidelines to consider in each step.

The three companies involved in this study are industrial partners of the Center for Applied Software Engineering at Lund University (LUCAS). Part of LUCAS is the LUCAS Architecture Academy that is a one-year part time software architecture education program for the LUCAS partners. This research is done based on issues that came up in the context of the architecture academy.

2. Method

In this study we have studied seven architectural changes initiated at three Swedish software-developing companies. The project has included a number of sessions where the companies present their architectural work for each other, and issues in the area have been raised and elaborated.

The approach taken in this research can be described as *flexible* [7]. This type of research is characterized by less pre-specification than in, for example, controlled experiments. In a flexible design the major research questions can be specified in advance, although they must be allowed to evolve during the course of the research.

Qualitative data has been collected in two sets of interviews. The first set was held with architects and system designers at the three companies to collect information about the companies, their products, and their architecture. Recent architectural changes were identified. Key persons in those changes were interviewed in a second set of interviews. These interviews were guided by the four questions mentioned in the introduction. The data was then analyzed according to the following factors:

- Architectural change
- Phase of change process
- Topics that were considered important in changes

The collected data was categorized and tabulated according to these factors, and analysis was carried out through discussion and pattern searching.

3. Process Overview

This section describes our suggested process for making technical decisions. The process is illustrated in Figure 1 and has been derived from the case studies, Section 4. The relation between the process and the case studies is shown in Section 5. The purpose of this process is to enable organizations to make the right decisions by the right people at the right time. From an employee viewpoint the process shall give guidance in the decision process, both for change initiators and decision-makers. Note that one aspect that differentiates the architectural change from the functional change is that the functional change usually is initiated by a customer request, and there is usually someone in the organization dedicated to handling these, e.g. product management. Architectural changes can be initiated by many roles in the organization.

The general process for functional changes involves requirements elicitation, pre-studies, implementation, and related decision-points. It focuses on how an organization shall *make decisions*. Kotter's [9] process for large-scale organizational change instead focuses on how to *make changes happen*. In the following process the two features are combined. This has basically been done by mapping Kotter's change process onto the functional change framework, which is considered to be fairly established in software industry. In practice the process therefore has to be adapted to the present functional change framework.

1. **A need emerges**: The process is superceded by a chain of events where need for change emerges or is created, and someone, the change initiator, sees this need and considers it his or her responsibility. This can to some extent be compared to Kotter's *Establishing a sense of urgency*, and to requirements elicitation in a functional change process.
2. **Initial decision preparation**: In this phase the change initiator does preparations with the goal of getting resources to analyze and implement the change.
   - Document background: To increase the chance of having an impact on the resolution of the need, the change initiator should document the background of the need, i.e. what products, components or organizational entities are involved, the history behind the need, how it manifests itself, what effects it might have not to satisfy the need etc.
   - Identify stakeholders/decision makers: While documenting the background, stakeholders are sure to emerge. In order to have optimal impact, the change initiator should pay special attention to these and especially to the decision makers that will be involved in the following process. This is related to Kotter's *Creating the guiding coalition*.
3. **Decision: Go/no-go**: An initial decision must be made whether the issue at hand is adequate and feasible to treat. Probably, there has not been spent very much effort before this decision point, e.g. one person's work for hours or days. Work done in the rest of this process, but before a decision on any particular solution or implementation of change, probably requires resources that must be budgeted, e.g. a handful of persons or more, which work for days or weeks. Therefore a person responsible for resources must make a decision whether to go on with this process or not. The formality of this decision-point is controlled by the organization at hand. If the change

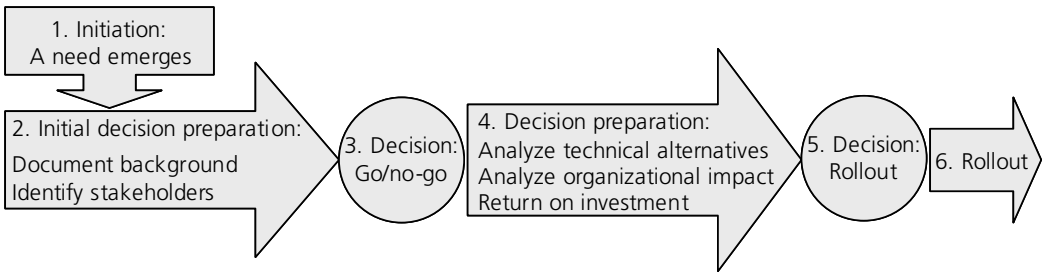

Figure 1. The process of architectural change

can be viewed as a normal product requirement or change proposal, it can be treated as such through the ordinary channels: implementation proposal and related decision points. If the change however is more of a change in the way people work, or a change in an internal quality attribute not leading up to completion of a specific project, the process steps that follow are of a different complexity. The risks of facing opposition are higher and the decision process and preparations must be more thorough.

4. **Decision preparation**: This phase is akin to performing a pre-study or developing an implementation proposal in technical change management. In terms of Kotter's process, it resembles *Developing a vision and strategy*.
   - Analyze technical alternatives: When technical alternatives have been proposed, these can be analyzed from an architectural viewpoint in a number of ways [11], i.e. ATAM [12].
   - Analyze process and organization impact: When making a technical analysis, the organizational implications are often forgotten. This might lead to unexpected resistance to a change. An organizational analysis is therefore made, based on the initial analysis of stakeholders, in order to assess the impact of the change and prepare the organization for the change. The activity therefore contains parts of Kotter's *Communicating the change vision*.
   - Return on investment: The need that the change satisfies has to have a financial side. A return on investment analysis will simplify getting support for the change from top management and management of any project that might implement the change. This activity will support Kotter's *Generating short-term wins*.
5. **Decision: Rollout**: Software projects generally have a tollgate or decision point where it is decided which implementation proposals will be include in the resulting product. The same decision is made in this phase, regarding technical aspects of the architectural change. Organizational changes are however not suitable to implement in a product oriented project, and will therefore need another form of implementation and associated decision.
6. **Rollout**: This activity involves the implementation of the change. The objective of this process is that the rollout of the technical part of the change shall be carried out within an ordinary project, i.e. where generally most organizational resources are allocated. This has to be synchronized with the rollout of the organizational change, which must be managed by, and given resources from, the line organization. This activity is related to the late phases of Kotter's process: *Consolidating gains and producing more changes*, and *Anchoring new approaches in the culture*.

When comparing to Kotter's process it is important to keep the proper context in mind. Kotter presents a process for long-term organizational changes, which means some phases are of a different scale. Kotter's process also focuses on engaging employees and preparing an organization for a change, and not so much on how to perform the actual change. Since this paper focuses on changes to software architectures, we can use the decision framework common in software projects as a basis for a change process with features of both perspectives.

4. Case Descriptions

This section describes architectural changes at three companies, located in southern Sweden. All companies develop products to a mass-market, and their products have long lifetimes. This implies that their architectures need to support several simultaneous versions of their products, with several releases over an extended period of time.

4.1 Company A

Company A develops control system environments for industrial automation, e.g. chemical plants, dairies, oil platforms, etc. The control system environment consists of both a development view, called control builder, and a deployment view, i.e. the controller itself. Within the control builder, controllers can be designed by specifying hardware sensors and actuators, constructing control loops, and connecting variables in those control loops to the hardware devices. A fully specified system can then be compiled and deployed onto a controller in a control system.

Company A typically carries out one large project at a time, involving the entire organization. Each project evolves the same product further by adding features to the control builder, e.g. new editor facilities, and the controller, e.g. new hardware interfaces. Implementation proposals are developed during a feasibility study. Accepted implementation proposals pass a tollgate, where after implementation begins. Development is organized in teams, each working on a number of implementation proposals. Work is feature-focused and the organization has no module-responsible and no architects, but instead relies on senior developers to take responsibility for long-term architectural goals. Two changes were studied at the company:

Protocol Framework: Company A recently acquired companies within their domain in order to increase their market share. The controller developed by Company A was intended to replace those companies' products. To support the same customers, the controller therefore had to support a number of legacy protocols from those products. This was realized as a problem using the present architecture, as the protocols were intertwined with the rest of the code, and could only be developed at one site, the one studied here. This site only had capacity to develop 1-2 new protocols per project. To be able to develop

several protocols a year, Company A decided to develop a generic IO and communication protocol framework. The solution was developed through a pre-study and an implementation proposal, which resulted in a solution that enabled frequent releases of the product with many new or legacy protocols in each release. This would be accomplished by letting other departments of the company develop the protocols they were responsible for, using the protocol framework.

Real-Time Operating System: Company A had for a number of years had discussions about cutting licensing costs on Real-Time Operating Systems (RTOS). A suggestion from local product management at the studied site to develop their own RTOS was rejected by local development management. In parallel, high-level management decided to reduce the number of RTOSs to only one. This would not only lower licensing costs but also provide focus on a common competency regarding RTOSs and tool support, which would standardize and simplify distributed development. Top-level development management initiated a pre-study across all departments of the company. Participants were interviewed regarding their use of, and competencies in RTOSs. The site studied here used one RTOS, but the pre-study led to a recommendation for all departments to switch to another. Eventually the recommendation became a requirement for a project at the studied site. This requirement was postponed by the local organization, while an OS expert prepared a solution with a Virtual Operating System (VOS) layer, which was introduced in a later project.

4.2 Company B

Company B develops platforms for consumer electronic devices. These platforms are sold to external customers who configure the services within the platform to create complete products. The software platform consists of a number of modules, and a middleware layer hides the internal architecture from the customers.

Projects are organized in: a project management group, with product management responsibility; a system-engineering group, with expert groups and function groups responsible for major features within market requirements; and a system realization group, which receives specifications from the system engineering group and develops the platform. The system realization group is divided into a hardware- and a software branch, which are subdivided into development teams responsible for a set of modules. The organization has module responsible that work with function groups during specification and development teams during implementation. The company also has a dedicated architecture group that performs most of its work within projects, especially supporting and influencing the system-engineering group. Three changes were studied at the company:

Data Router: During routine reviews the system-engineering group discovered several modules handling data streams in similar ways. These modules could instead use a common data router and thereby save memory. The architecture group developed a design proposal that was approved, but no resources were provided from the project. Project management did not consider the memory savings to be large enough. Therefore the solution was implemented by the software architecture group, and integrated with a small-scale system on an isolated branch of the code. After inspection this branch was merged with the main track, and the software architecture group initiated documentation and education on the new architectural mechanism. The solution was still not widely accepted, as most modules already had their own implementations of the same functionality.

Hardware Abstraction Layer (HAL) Split: The bottom layer of the architecture had existed in previous versions of the product, but had not been formally defined, and therefore there had been no clear rules as to how to access the hardware. The hardware was also not encapsulated well enough from the majority of the software, leading to unnecessary impacts in the software when the hardware changed. The developers working in the lower layers of the product realized the need for a clearer definition of these layers. They proposed a solution that meant clearing the HAL interface from hardware dependencies, i.e. creating a logical layer on top of the previous HAL. One driving force for introducing this logical layer is that the cost for a product developed from the platform is very dependent on the hardware components used, and therefore these are often changed to provide cheaper solutions. The purpose of the logical layer is to allow such changes without expending effort in the higher layers of the software.

The solution was presented for the system-engineering group and brought to the software architecture group. When the proposed solution was established within the system-engineering group and the software architecture group, project management decided to assign resources to the change. The software architecture group introduced new coding rules according to the suggestion and made changes to the architecture descriptions. At the same time, the developers in the HAL prepared by planning the change, before doing the actual implementation when resources were assigned and the architecture was updated.

Include-file Reorganization: The software architecture group had created a flexible structure for the source- and include-files. The design rules that enforced this structure required several files for each component, and when the number of modules grew to around 100, unexpected effects on the development tools emerged. Compilation times increased, the configuration management system behaved sluggishly and the globally distributed CM servers started to crash more frequently. The persons responsible for tool support within Company B were in contact with support personnel from the tool supplier, who identified the problem as having too many files in the system. The software architecture group was assigned to create a new structure.

The flexibility provided by the original structure was only needed by a few of the about 100 modules, and these could continue to use the previous structure. The rest of the modules were given a new structure, which basically involved merging three or four source files into one file. This resulted in a three-to-one reduction of source files.

4.3 Company C

Company C develops software engineering tools. One of their main products is a design tool that consists of a front-end with editors for various types of diagrams and source code, and a back-end for compiling the diagrams into code. Other utilities such as a simulation tool are also part of the design tool.

Company C releases a new version of their product every six months, and successive release cycles overlap. Features are implemented by development teams in an assembly-line fashion, described in [13, 14]. The organization has architects per project but no establisehd line organization for architecture, and module responsibility is assigned to senior experts. Two changes were studied at the company

Communication Mechanism: New requirements, especially related to new language standards, have meant that the old architecture could not support further development. Therefore top-level management decided to create a new product generation. Company C had recently acquired other companies, which developed software engineering tools that were to be integrated into the new product. One of the problems with the new requirements was an increase in the number of diagram editors. The old communication mechanism did not support this increase, but one of the acquired companies had recently solved that problem, using a common object model. A technical discussion led to a consensus of using the new solution, although it meant major architectural changes.

Editor Framework: The editor framework used to develop graphical editors was also changed using a more generic solution, a decision also taken by consensus in the development project. The drivers for this change were increased reuse of common editor elements, and outsourcing of development throughout the organization. Several other decisions in this change process had to be enforced by the responsible architect, as consensus could not be reached. Both these changes were introduced in the same project.

5. Analysis of Process versus Cases

This section compares the process suggested in Section 3 to the architectural changes described in Section 4.

5.1 A Need Emerges

Before the suggested process is initiated a need for a change somehow appears. The reasons for changes in this report has included business decisions to increase market share, lower costs and lead time, but also more technical reasons where the architecture has not been able to support increased complexity and new features.

In Company A the need for the protocol framework was initiated when top management decided to increase the market-share by acquiring other actors in the same domain. Mid-level managers and experts then saw the need for support of legacy protocols found in the newly acquired companies' products. The need for a change of RTOS on local level came from a higher-level need to save licensing costs and focus competencies by reducing the number of RTOSs. The process was initiated by higher-level management and supported by developers at other sites of the organization.

In Company B the introduction of a data router was driven by memory size being an important quality attribute. The opportunity to save memory was discovered by system engineers during routine code-reviews. The need for a HAL split emerged as the company wanted to be able to change hardware components frequently in order to save costs. The hardware-related developers themselves initiated the change in order to simplify the frequent changes. The need for an include-file restructuring became apparent, as the configuration management tool did not support the existing structure. The architecture group initiated this change since they were responsible for the include-file structure.

The product generation shift performed in Company C contained two major changes. A new mechanism, which allowed different editors to work against the same system representation, was introduced in order to increase the number of possible editors. A framework for editor development was introduced in order to increase reuse of common editor components and enable outsourcing of editor development. Local experts initiated these changes and the technology came from the newly acquired companies.

Change initiators have been identified from all levels of the companies, i.e. managers, experts appointed when the issue came up or as part of their ordinary role, where a special case is the architects themselves, and down to the developers. This can be compared to functional changes where needs often emerge from customers and are taken care of by marketing or product management.

5.2 Initial Decision Preparation

A decision process that can be initiated by non-decision makers will eventually have to be brought before a decision maker. In this phase the change initiator documents the background of the issue, and identifies stakeholders and decision makers.

When the need for legacy protocol support had emerged in Company A, local experts and managers analyzed the protocol framework solution in a pre-study. Limited attention was however paid to other departments that were supposed to implement protocols on this framework. Regarding the change of RTOS, the pre-study had been carried out by higher-level management. This re-

sulted in recommendations to change to a single and specified RTOS. The pre-study involved interviews on all company sites.

The introduction of a data router in Company B was initially prepared by the system-engineering group by marking the places were similar functionality had been found. Stakeholders such as current users of such functionality and future clients to the data router were loosely identified but not further analyzed. The only stakeholder that was approached was the architecture group, who would be responsible for developing an implementation proposal. Regarding the HAL split, the developers in that layer prepared a solution themselves, and set up a meeting with the appropriate decision-makers, in this case the system-engineering group. In the case of the include-file restructuring, the initial preparation was made by the tool-vendor's support organization. They concluded that the projects contained too many files. The architecture group was identified as a stakeholder, since they had developed the previous structure. Apart from that, stakeholder identification was not done actively, since the frequent tool failures meant that stakeholders presented themselves.

In Company C the first steps of the product generation shift were taken on many levels, both within the original organization and by developers and managers in newly acquired organizations. Technical discussions were held which lead to the realization that the whole architecture had to be changed. Solutions were gathered from all parts of the organization, and the new architecture was adapted to enable distributed development. Stakeholders and decision-makers were therefore covered.

In the case studies we have seen examples of less successful changes, where too little has been known about the impact of the change. Effects of functional changes are often more limited and customer-oriented. As opposed to architectural changes, functional changes often have resources allocated to this phase, such as product management performing requirements elicitation.

5.3 Decision Point: Go/No-Go

In this activity the first decision to commit resources is made. The right decision maker shall have been defined previously, and process and organizational issues must not be forgotten in this decision.

In Company A, local level management decided that an implementation proposal of the protocol framework should be developed, since the solution would allow for more protocols and more frequent releases of the product. The organizational impact was not given much focus in this decision. Regarding the RTOS switch, top management decided to turn the recommendation into a requirement for the following projects. This requirement was later postponed by the local organization.

In Company B, the system-engineering group decided that the architecture group should develop an implementation proposal of a data router. Regarding the HAL split, the solution was so well prepared by developers that neither the system-engineering group, nor the architecture group had to invest large resources in preparation, and therefore the related decision was of little significance. Regarding the include-file structure the architecture group themselves decided that they should develop a solution. Resources spent by the architecture group were considered insignificant in comparison to the resources wasted during tool problems. Organizational impact related to difficulties in rolling out the new structure was considered at this stage.

In Company C, the decision to apply resources to the change process was at a higher level, since it involved starting a whole new line of product-oriented projects. A decision was therefore made by top management to prepare and plan for a first project, which should result in a prototype for the product.

A forum for architecture issues could be helpful when making this decision. Considering functional changes, organizations sometimes have product management fora, making similar decisions. The problem for the architect is that the decision is one of resources, for which the architect seldom has responsibility. Getting project resources has a benefit since the change can be more easily embraced by that project. It is however not trivial to receive resources from a project manager.

5.4 Decision Preparation

In the decision preparation phase a small group of people will analyze technical alternatives, process and organizational impact, and return on investment. From a company viewpoint this is done to make the right decision, and from an architect or change initiator viewpoint this will help convincing people of the need for change. This phase is similar to developing an implementation proposal when making a functional change, and should therefore be adapted to how implementation proposals are handled within the organization. The analysis of technical alternatives can be done in parallel with the analysis of process and organizational impact.

At Company A, the protocol framework was prepared by developing an implementation proposal, in the same way as a normal requirement. The technical solution was based on expert opinions. The process and organizational impact was considered, and a pilot study was made which involved developing a protocol at another site in the same company. However, there are many developers in the acquired companies that are impacted by this change but have not been involved in the first phase. The change of RTOS was postponed to a later project, and in the meantime an OS expert prepared a solution involving a VOS layer to allow for several operating systems. One organizational impact was overlooked, as the change meant that new RTOS support contacts had to be established. Regarding return on investment, the change of RTOS lead to no short-term wins for the local organization.

In Company B, the architecture group developed the data router solution in a pre-study. It was based on already

implemented solutions, but the group failed to realize opposition from project management and developers. A return on investment was calculated late in the process. The developers had already prepared the HAL split so the architecture group only had to prepare changes to architecture documentation and design rules. No quantitative return on investment was made but the ability to change components was considered an obvious benefit. Regarding the include-file restructuring, the architecture group found that the flexibility provided by the original structure was only needed in a few modules, and a simpler structure was created for other modules. Return on investment calculations were made regarding the rollout, since rollout was expensive and did not contribute directly to any product.

In Company C the first project of the new product generation was planned. When making technical decisions many parts of the organization were involved, and consensus in joint forums was the goal. When this could not be reached, the architect responsible for that type of functionality had to make the decision. Organizational impact was not only considered when selecting solutions, but also when distributing development of various modules. This distribution could at least in one case have been better planned, as they ended up with developing a module at one site, which was highly dependent on two other modules at another site, leading to unnecessary problems.

5.5 Decision Point: Rollout

When a feature-oriented implementation proposal is completed, it is generally passed through a tollgate in the project. In this tollgate the project decides which features or implementation proposals shall be included in the upcoming release. The activity described here is similar, but the changes we have studied have had organizational impact. Such changes, and their related decisions, are hard to make in a product-oriented project, i.e. a project that will result in a product aimed at the market.

In Company A, the implementation proposal for the protocol framework involved two different types of protocols, and a set of services for these protocols. The decision to implement was made according to the standard project model. Both protocol types were to be implemented in the upcoming project, but a part of the services were postponed to later projects. Regarding the change of RTOS, the expert's VOS solution was chosen, and local development management decided to roll it out onto a current project. This project had to start implementation before the VOS was ready, and therefore local development management decided that the VOS team would make relevant modifications of the project's code when the VOS was ready.

In Company B the system-engineering group approved the implementation proposal for the data router, but the architecture group did not receive project resources to implement the proposal. They then decided to implement the data router with their own resources. The HAL split was however granted resources by project management, because it had backing from developers, system engineering, and the architecture group. The include-file restructuring was urgent, but difficult to roll out. First a script was developed that would automate rollout. This script depended on that the design rules had been followed, which was not the case. A second strategy was to halt development over a number of days, and perform the changes manually. This solution was too costly and eventually appropriate line management decided to roll the new structure out onto newly started projects, letting old projects use the old structure.

Company C decided to launch the series of projects for the new generation of products. Top management took this decision, and the content of each project has slowly been decided throughout the first projects by top management, product management and project management.

One conclusion from this activity is that it might be beneficial to restrict functional content of a new product when introducing major architectural changes. This was adequately done when introducing the IO and communication framework in Company A, as the number of services available to the protocols was restricted in the first release. Company C has however had problems deciding on the final content of the first product to be released on the market. Restriction of functional content is a tradeoff since customers will not accept lower functional content, and the new architecture must be able to support future functional content. Another tradeoff regarding how many future features an architecture should enable concerns the debate of programming for the future or, as XP [15] advocates, programming only for the present.

5.6 Rollout

Implementation of technical aspects of changes is made successfully within product-oriented projects. Implementing technical aspects elsewhere is more problematic, since such implementations are not so easily embraced by developers in projects. The problem is that the process and organizational aspects are often forgotten in product-oriented projects, and there seldom exists a standard routine for carrying out such changes, as opposed to carrying out a product-oriented project.

In Company A the protocol framework was implemented as part of a product-oriented project, but many departments that were intended to develop protocols have not yet had opportunity to give feedback on the framework. There is therefore still a risk that some departments will object to the framework. The VOS was developed in parallel with a product-oriented project. When the VOS was ready the two projects were merged, and the VOS team had to make remaining modifications.

In Company B the architecture group developed the data router on an isolated branch, which was later merged with the main branch. The problem was that most of the clients to the new data router already had implemented their own solutions, and usage of the router was only rec-

ommended, not required. It has therefore not provided the anticipated memory savings. The HAL split had been well prepared by both developers and architects before decisions were made, and it was rolled out as part of a project. The new include-file structure was rolled out onto one project at a time across the whole organization. The roll-out coincided with an architectural change, which lead to little overhead when the key module responsible checked in the new file structure into the tool at project startup.

Company C has implemented their architectural changes in a prototype project, and a product for the market is under development. The main problems have been to settle on feature content, and as previously mentioned, the distribution of work.

In the case studies we have seen several examples of changes where the technical part has been assigned to a certain project as a requirement, but postponed to later projects. We have also seen examples where the changes have been performed outside of product-oriented projects, further decreasing the chance of embracing the change. One of the cases made a satisfactory tradeoff, where the change of operating system was postponed to a later project, but prepared by an expert ahead of the project start.

6. Conclusions

In the case studies we have seen that need for architectural changes can emerge from various sources, and that various roles, such as managers, architects and developers, may take responsibility for initiating the change. The decisions regarding architectural changes are often carried out in the same way as companies make decisions regarding functional changes, while the implementation of architectural changes may take many forms, such as part of ordinary projects, parallel but separate projects, independent smaller projects or as new full-scale projects.

We have discovered three major differences between functional changes and architectural changes. First of all, architectural changes are often more complex than functional changes and affect large parts of the product without showing a clear connection to a customer need. Secondly, architectural changes do not only have impact across large parts of the product, but often across the whole organization, and changes of processes and organization are often overlooked and hard to implement in product-oriented projects. Finally, while companies often have mechanisms and resources in place to treat functional changes, such mechanisms are seldom established for architectural changes, and it is also hard to commit resources to activities without clear customer value.

We believe the process presented here helps putting focus on organizational issues in an architectural change, while taking advantage of the decision support found in the ordinary functional change process. This will lead to that the technical part of the architectural change is implemented according to company standard, hopefully within a product-oriented project.

In further studies, the process presented here could be optimized by running it in pilot studies. A goal of such studies could be to find a framework for implementing the organizational part of the change.

Acknowledgement

We would like to express our gratitude to the companies and individuals participating in the LUCAS architecture academy. Without their help this work could not have been completed.

This work was partly funded by The Swedish Agency for Innovation Systems (VINNOVA), under a grant for the Center for Applied Software Research at Lund University (LUCAS).

References

[1] Garlan, D. and Shaw, M., *Software Architecture: Perspectives on an Emerging Discipline*, Prentice Hall, 1996

[2] Soni, D., Nord, R., and Hofmeister, C., "Software Architecture in Industrial Applications", Proc ICSE, pp. 196-210, Seattle, 1995

[3] Kruchten, P., "The 4+1 View of Architecture", IEEE Software 12 (6), pp 42-50, IEEE, 1995

[4] Bass, L., Clements, P. and Kazman, R., *Software Architecture in Practice*, Addison-Wesley, 1998

[5] Hofmeister, C., Nord, R. and Soni, D., *Applied Software Architecture*, Addison-Wesley, 1999

[6] Bosch, J., *Design and Use of Software Architectures*, Addison Wesley, 2000

[7] Evans, R., Park, S. and Alberts, H., "Decisions not Requirements: Decision-Centered Engineering of Computer-Based Systems", Proc IEEE Int. Conference and Workshop on Engineering of Computer-Based Systems, pp. 435-442, 1997

[8] Regnell, B., Paech, B., Aurum, A., Wohlin, C., Dutoit, A. and Natt och Dag, J., "Requirements Mean Decisions! - Research issues for understanding and supporting decision-making in Requirements Engineering", Proc 1:st Swedish Conference on Software Engineering Research and Practice, pp 49-52, Blekinge Institute of Technology, 2001

[9] Kotter, J. P., *Leading Change*, Harvard Business School Press, 1996

[10] Robson, C., *Real World Research, 2nd Ed.*, Blackwell Publishers Inc., 2002

[11] Dobrica, L. and Niemelä, E., "A Survey on Software Architecture Analysis Methods", IEEE Transactions on Software Engineering 28, pp 638-653, IEEE, 2002

[12] Clements, P., Kazman, R. and Klein, M., *Evaluating Software Architectures*, Addison-Wesley, 2002

[13] Regnell, B., Berenmark, P. and Eklund, O., "A Market-Driven Requirements Enigneering Process", Journal of Requirements Engineering 3, pp 121-129, Springer-Verlag, 1998

[14] Höst, M., Regnell, B., Natt och Dag, J., Nedstam, J., and Nyberg, C., "Exploring bottlenecks in market-driven requirements management processes with discrete event simulation", Journal of Systems and Software 59, pp 323-332, Elsevier Science Inc., 2001

[15] Beck, K., "Embracing Change with Extreme Programming", IEEE Computer, October 1999, pp 70-77