Experiences in assessing product family software architecture for evolution

Alessandro Maccari
Nokia Research Center
P.O. Box 407
FIN – 00045 NOKIA GROUP (Finland)
+358718008000
alessandro.maccari@nokia.com

ABSTRACT

Software architecture assessments are a means to detect architectural problems before the bulk of development work is done. They facilitate planning of improvement activities early in the lifecycle and allow limiting the changes on any existing software. This is particularly beneficial when the architecture has been planned to (or already does) support a whole product family, or a set of products that share common requirements, architecture, components or code. As the family requirements evolve and new products are added, the need to assess the evolvability of the existing architecture is vital. I illustrate two assessment case studies I have recently worked on in the mobile telephone software domain: the Symbian operating system platform and the network resource access control software system. The former assessment has been carried out as a task within the European project ESAPS, while the latter has been performed solely by Nokia. By means of simple experimental data, I show evidence of the usefulness of architectural assessment as rated by the participating stakeholders. Both assessments have led to the identification of previously unknown architectural defects, and to the consequent planning of improvement initiatives. In both cases, stakeholders noted that a number of side benefits, including improvement of communication and architectural documentation, were also of considerable importance. I illustrate the lessons we have learned, and outline suggestions for future research and experimentation.

1. INTRODUCTION

Once the software infrastructure – operating system, programming language, middleware, communication mechanisms etc. – has been selected for a new product, the other parts of software architecture are often neglected. This is particularly evident in large organizations, where the software architect must make decisions that are often important, technically complex, and difficult to reverse when the development is under way.

The architect's job is particularly challenging, since it involves a continuous problem solving activity, which sometimes means re-architecting: over time, architecture "ages" and weakens the system's capacity to incorporate new features; re-architecting means changing the architecture to make systems easier to expand and maintain.

Stakeholders of a system (and hence of its architecture) are people or organizational entities who have an interest towards the fulfillment of the system requirements. The end user is the most visible stakeholder, and the one usually interested in functional requirements. However, there are usually other stakeholders, who are often interested in the fulfillment of non-functional, or quality, requirements. Quality requirements (the so-called "–ilities", plus performance) are usually the most difficult ones to satisfy (and those that become important when they are not satisfied!). Architectural assessment is an essential part of the architecting process, aimed to evaluate the degree of fulfillment of quality, or non-functional, requirements.

This paper focuses on the architecture of software product families. Product families are set of products that share architectural properties, features, code, components, middleware or requirements [1, 2]. The architecture of a product family must support all the envisioned products, and, optimally, some evolution paths that had not been initially forecast. Often, the product roadmaps have to be changed after the initial platform architecture is established. This justifies carrying out architectural assessment for product family evolution.

I report on the experience from two case studies that our research group has conducted together with Nokia Mobile Phones and other partners. Both case studies were performed on real systems, and had business relevance for the companies involved.

In the first one, we evaluated the suitability of the Symbian operating system platform to support a hypothetical, future mobile telephone product family. The second case study built on the previous experience, and assessed the architecture of the software system that controls access to network resources in our mobile handsets. The former experiment was performed in co-operation with some partners from the ITEA project ESAPS [3] and partially funded by the European Union, while the latter was performed and funded entirely by Nokia.

2. SOFTWARE ARCHITECTURE ASSESSMENTS

Several existing methods – such as ATAM [4], see section 7 for more related work – treat the assessment of software architecture quality. Various business goals can trigger architecture assessment activities.

- To evaluate and improve the architecture and its qualitative attributes.
- To evaluate its conformance to a certain standard.
- To check whether certain qualitative requirements are satisfied by the architecture.
- To identify the skills needed for implementing the system.
- To validate the partitioning for implementing the system within a certain organization.
- To identify the risks related to a particular architecture.

An important, indirect consequence of performing assessments is the improvement of communication between architects, developers and other stakeholders (see Section 6 for more details on this).

The input for assessment is the available documentation and knowledge about the architecture. Such knowledge can be extracted from different sources: architectural documents, interviews with experts or reverse architecting activities [5]. Ideally, every assessment should result in a series of improvement activities, targeted to both the architecture and its documentation.

3. APPLICATION TO PRODUCT FAMILIES

A product family can be defined simply as a set of products that share common requirements, architectural properties, components, middleware, code or any other software artefact.

Ideally, product families should be planned and scoped in the very early lifecycle phases, and the common architecture should be designed to support all the planned products and as many evolution paths as can be foreseen. However, such a thorough and infallible planning practically never happens, and the case where the architecture has to be ported to support new products (that it was not originally designed for) happens rather frequently.

Building new products on top of an existing architecture has clear advantages: most importantly, it allows saving architecture development time, reusing knowledge, expertise and documentation, and employing known development methodologies.

However, it presents some risks as well. New requirements coming from future products may imply modifications in the architecture, which in some cases may be cumbersome. If the required architectural modifications turn out to be too expensive, it may be worth to change the architecture, or re-scope the whole product family. The impact of such requirements should be estimated as early as possible, preferably when architectural and system-level design decisions are made. If this does not happen, the unsuitability of the architecture for a certain product may reveal only during development or even as late as testing phase, thus increasing the cost of change.

This risk justifies the need to assess a product family architecture before it evolves. The flow of assessment activities that we followed is illustrated in Figure 1. As soon as the requirements of the future products are known to some degree of certainty, the stakeholders of the product (including developers, architects, customers, domain experts, etc.) evaluate the architecture for evolution. Together, they elaborate the evolution scenarios, which represent the consequence of implementing requirements introduced by new products. After the scenarios are agreed upon, the actual assessment takes place during a meeting (which usually lasts one or two days). There, after the chief architect has given an overview of the architecture, the evolution scenarios are walked through sequentially. The meeting participants brainstorm and discuss about the capability of the architecture to support every depicted evolution scenario. Finally, the assessment co-coordinator writes a report that lists and elaborates all issues, identified architectural defects or shortcomings as well as improvement ideas. This should constitute input for re-architecting and re-documentation.

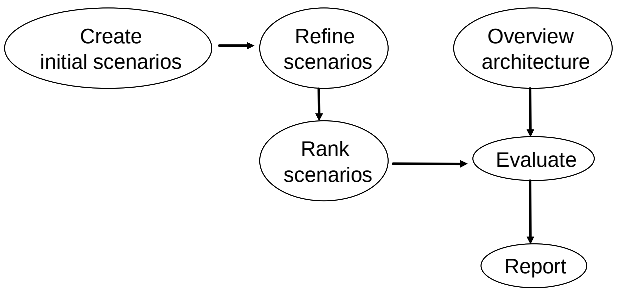

Figure 1. Architectural assessment activities.

I hereby focus on the experience gained from our two case studies, with attention to social and organizational issues. I produce some empirical evidence on the usefulness of architectural assessments, based on simple error detection statistics, and illustrate the experience as evinced from interviews with the participants. Finally, I outline some issues for future research in this field.

4. CASE STUDY 1: SYMBIAN OS PLATFORM

The first assessment experiment I describe was carried on the Symbian operating system platform. This work was carried out as part of the EU-project ESAPS [3] and hence in a slightly different way than a company-specific, business-critical assessment. The case study was performed partly as a validation experiment for the product family architecture assessment method that had been previously devised in the project, and partly as an exercise for the project team to become familiar with the new method.

4.1 Background

The Symbian operating system [6] was originally developed by Psion as a lightweight, window-based operating system for portable devices equipped with full, PC-like keyboard and a large, graphical display. The operating system was a natural candidate to be employed in mobile telephones that include PDA functions. At the moment, several commercially available products (such as the Ericsson R380S or the Nokia 9210 Communicator) run the Symbian platform.

At the time of the experiment, Symbian had only been used as an operating system for personal digital assistant (PDA) products, and on mobile telephones derived from the same concept. The companies that participated the assessment needed to assess whether the Symbian platform could be used to support a full mobile handset product family. This would include at least the following product categories.

- Basic phones. These products should include only the essential features and be sold at a cheap price (which, among other things, implies a low-frequency processor and little memory); basic phones provide the most important communication-related applications, but only reduced input (usually, a telephone keypad plus a number of function keys) and output devices (a small-sized, low-resolution display).
- Communicators. They combine a mobile phone with a PDA. The most common communicator concept features a dual input/output mechanism: an ordinary mobile phone keypad and display that handles communications functionality, and an extended, laptop-PC-like keyboard with a large, high-resolution graphical display for PDA functionality.
- Smart phones. This rather new concept is an evolution of the basic mobile phone. The typical smart phone has a medium-size, high-resolution screen, a reduced keypad aided by different, co-existing input mechanisms (e.g. touch screen, pen, voice, keypad, mouse).
- Web pads, or pocket web browsers. Web pads are similar to smart phones, e.g. they also support different input devices (such as wireless mouse, pen, voice recognition, eyeball movement). There is a difference in flavor between smartphones and web pads: the former are mainly conceived to be used as telephones, with additional functionality such as multimedia or wireless internet connection; the latter, on the other hand, are specially made for web navigation, although they do support basic telephony functionality and may run selected office applications.

Given the relative novelty of the problem, it was decided to perform the assessment as a joint effort between two mobile phone manufacturers (Nokia and Ericsson), an academic institution (University of Karlskrona in Ronneby, nowadays called Blekinge Institute of Technology) and the platform developer (Symbian).

This was possible since the first three partners were taking part in the same EU-ITEA project, which partially funded the activity. The organization responsible for the development of the operating system platform (Symbian) had an obvious interest in participating, justified by the possibility of gathering improvement ideas.

It was thought that joining forces might help all participants understand the domain and correct the inevitable mistakes due to general inexperience in assessments. Cross-inspections of the assessment material (architectural models, assessment scenarios, etc.) were held before the assessment meeting.

4.2 Activities

All partners except for Symbian independently collected the relevant evolution scenarios. After internal reviews, they were submitted to the other partners for inspection. This happened a few weeks before the assessment meeting, allowing time for corrections. The inspection helped clarify ambiguous scenarios and remove duplicate or irrelevant ones.

Subsequently, the scenarios were merged into a single list, categorized according to the focus area and prioritized in decreasing importance order. This was the chosen walkthrough order for the assessment meeting, and ensured that a possible lack of time would not leave out the most critical scenarios.

The assessment meeting took place in April 2000 and lasted one day. It was performed according to the process depicted in Figure 1.

The assessment report was circulated among the experiment participants and other partners in the ESAPS project for inspection. Once approved, it was published as a consortium-wide deliverable for the ESAPS project.

4.3 Results

The assessment was regarded unanimously as a positive experience. The report highlighted three major architectural shortcomings, along with a high number of positive evaluations. The defects were reported to the developing organization, which started corresponding improvement activities. From this I infer that the assessment was successful also business-wise, in that it led to the isolation of significant problems in the architecture, and to allocation of effort for their solution.

All experiment participants valued the knowledge they had acquired during the experience as useful. The participants from the industry partners were also positive about the possibility of performing a second internal assessment, which would build on this experience, adding details on the implementation that could not be revealed in the presence of competitors and externals. In Nokia’s case, a follow-up assessment did take place on the wake of the first one. This constitutes more evidence that the experience provided useful insight and a basis for similar initiatives.

5. CASE STUDY 2: NETWORK RESOURCE ACCESS CONTROL

The Symbian operating system case study was our first experience of architectural assessment for product family evolution. We exploited the knowledge acquired during that experience and started an architectural assessment initiative that is still ongoing. One of the first case studies has been the network resource access control software system.

5.1 Background

Applications in a cellular phone need to access different network resources, which correspond to different ways of carrying data over the air (bearers). Examples of network resources are: voice, SMS (Short Messaging Service), CSD (Circuit Switched Data, used e.g. for modem dial-up connections), etc. Recently, a new protocol (called GPRS – General Packet Radio Service, []) was introduced in wireless networks and (obviously) also in mobile handsets. GPRS allows data transmission over the air on a packet-switched connection. This is advantageous for the end user, who is billed proportionally to the amount of transferred data (as opposed to connection time, which is the case for CSD), but also for the operator, who can share data channels between different users, optimizing the usage of air space in congested networks.

The introduction of the GPRS bearer increased the number of constraints on the simultaneous usage of different bearers by applications. Such constraints are largely dictated by network standards (which vary according to the network type), but may also originate from architectural choices done by the manufacturer, usually for the sake of simpler design, higher usability or better maintainability.

An example of such constraints may be the mutual exclusion of two applications (e.g. WAP browsing over CSD and voice call). In some cases, however, constraints are not as simple as mutual exclusion: there are examples of applications that may or may not run simultaneously, depending on the status of a third application. Moreover, there could be constraints on the simultaneous running of certain applications, regardless of the bearers they use. Also, interruption rules and priorities may change dynamically, according to factors such as network connectivity (GPRS has three network modes), quality of service, user preferences.

Such interaction complexity brought up the (previously hidden) problem of resource access control. The term “resource” may be interpreted as a synonym for “network resource”, or, more generally, as “connectivity resource”, which includes e.g. local infrared connectivity. The system must ensure that the applications use the (network or connectivity) resources fairly, avoiding conflicts or deadlocks and handling errors and timeouts in a safe and reliable way.

In order to solve this problem we devised a control system that monitors the requests coming from applications to access the resources, handling all possible conflicts. When we started the assessment, most of the system software had not yet been written. In this context, the first goal of the assessment was to improve the quality of the existing architecture. We trusted this would allow us to identify potential problems and weaknesses of the current design, requirements that had been overlooked and general architectural issues.

Secondly, we wanted to reuse the solution that had been designed for the lead GPRS product in other products (belonging to the same family) that would support different combinations of bearers and applications. Hence the second goal of this assessment: to evaluate the scalability of the existing architecture to support new requirements (and the different constraints they involve). For instance, new bearers such as Bluetooth were in the roadmaps when the assessment was performed (Bluetooth-compatible products have already been announced at the time of writing), and possible architectural shortcomings needed to be pinpointed rapidly.

5.2 Activities

Due to the dual nature of the requirements, the collected scenarios were divided into two main categories: those referring to the lead product implementation (scenarios for architectural improvement) and those referring to the future products (scenarios for architectural evolution). Obviously, the scenarios for improvement got higher priority than those for evolution, since the bulk of the implementation work was to start very shortly after the assessment.

The process for the assessment preparation and meeting was similar to the one we followed during the Symbian case study. Naturally, the scenario elicitation did not take place in a distributed fashion (as was the case in the first experiment), and therefore took less time and fewer inspections and interactions. Also, the facilitators were more familiar with both the assessment practice and the architecture than in the previous case. However, none of the other stakeholders had any previous assessment experience, therefore some of the activities were not performed with the ideal effectiveness.

The assessment participants were the network access control software architect, the chief architect for the mobile phone software, two software developers, two network protocol requirements experts (who proved to be very useful when discussing details about the connections), two researchers and one requirement manager for future products. A few other people gave contributions to the scenario collection phase, but were not physically present in the assessment.

After the meeting, the facilitators produced a report that was inspected by the assessment participants, as well as by other stakeholders who could not make the meeting. After some minor modifications it was approved, and became part of the architectural documentation. In particular, the main architectural defects that had been identified in the assessment were transcribed in the “open items” section of the architectural document, and led to the start of architectural improvement actions. This constitutes evidence for the effectiveness of the assessment.

5.3 Results

Three architectural shortcomings were identified during the network resource access control assessment. One of them related to the current implementation, while two others related to the evolution requirements. The product family architecture has already been improved according to the recommendations that sprung from the assessment.

As in the Symbian case, all participants judged the experience positively, especially from the learning perspective. Feelings were mixed about whether architectural assessment would be the most efficient way to find errors: several people remarked that the preparatory activities took a considerable amount of time. I feel this may be due to the general lack of assessment experience.

The chief architect also noted that the assessment could have produced comparable (albeit not as detailed or motivated) results without involving such a large and diversified community of stakeholders. I believe that this remark should be interpreted in the context of a large organization that is constantly subject to extreme schedule pressure.

In fact, a few months after the experiment, a researcher in my team conducted a series of informal interviews with the stakeholders. In that occasion, the chief architect initially restated doubts about the efficiency of the assessment activity, but later conceded that having such a large group allowed very detailed analysis of the architectural shortcomings and hotspots that had been previously identified. The architect claimed that such detailed analysis had proved useful during the following architectural improvement phases.

The chief architect also claimed that the assessment did not require a large investment in terms of either time or money. Therefore, he maintained that the comparison between results obtained and the effort spent makes assessments a worthwhile activity.

This opinion can be substantiated by a few rough metrics. The whole experiment required approximately three months (elapsed time). The total manpower we invested in it consisted in 15 working days for each of the facilitators, and 2 to 5 working days for all other stakeholders (save for those who did not turn up at the assessment meeting, whose contribution should be computed in the order of a few hours). This adds up to approximately 50-60 person days, or no more than 3 working months. Considering that the facilitators did not work for the business unit where the assessment was performed, the effort spent by developers and architect is minimal.

Nevertheless, the assessment allowed us to identify 3 architectural defects, and over ten minor improvement items for the architectural documentation. These rough figures more than substantiate the opinion of the chief architect, demonstrating the effectiveness of assessment. To the picture should be added uncountable benefits, such as the increase in experience, communication and knowledge.

Moreover, the numbers are likely to improve in the average case, since all stakeholders (save for one of the facilitators) that took part in this assessment had no previous experience of this kind of activity. Further research is needed on this issue.

6. EXPERIENCE AND RESEARCH ISSUES

A number of interesting facts and issues have emerged from both experiences. Unless otherwise specified, the considerations that follow relate to (or originate from) both case studies.

6.1 Result

I collected some statistics and metrics for both case studies. The first case study examined an operating system platform in a fairly general fashion. Due to the presence of competing industries in the same room, no technical details could be revealed. Some of the stakeholder noticed that the assessment of certain scenarios could not be performed in a detailed way because the industrial partners were reluctant to share information about the implementation of certain features.

During the first assessment case study we debated 12 scenarios, out of which 3 highlighted potential problems in the operating system platform. One of the problems was unanimously recognized to be a potentially major weakness. During the follow-up phase, it was deemed necessary to start some research on the issue, a clear sign that the problem may have had a serious impact on the software product family quality.

In the second assessment experiment we walked through 9 main scenarios (most of which included more than one sub-scenario). Out of those, 3 concerned requirements for the lead product implementation and 6 were derived from forthcoming products.

Three scenarios revealed some kind of flaw in the architecture. In all cases, the chief architect and system expert believed they could be overcome with a relatively small effort. The assessment showed that the design of the architecture mostly supports the considered evolution requirements. In fact, it really turned out that none of the changes that aimed to solve the discovered flaws involved major effort allocation.

All the flaws that were discovered regarded dynamic system behavior when in particular combinations of events were considered. This shows how difficult requirements for mobile phones are, and how effective assessments can be in highlighting possible problematic areas.

From a critical analysis of the responses to the interviews we performed with the stakeholders, I could extrapolate a number of lessons learned, which I list below, distinguishing between process and organizational issues and lessons learned.

6.2 Process and organizational issues

6.2.1 Assessments are easy to do

Assessments are reasonably easy to perform, even by inexperienced people. In both cases, most participants were new to the practice. Yet, both activities run on or near schedule, and no one reported experiencing particular difficulties with any of the steps. When asked for the reason, a number of people seemed to indicate that the brainstorming-centered approach allows ideas to be generated fast, while ensuring constant checking of their validity by other participants. This aspect, however, calls for more qualitative research before it can be validated.

6.2.2 Collaboration between competitors is possible

In the Symbian assessment case, we were faced with a somewhat weird situation that featured competitors working together on a platform that they used for different (competing) products, and thus were on the border of confidentiality levels. The feeling was that the separate scenario collection followed by cross-inspections was the best approach. The case when competitors join efforts to evaluate an architecture in which they all have a stake is particularly interesting. I feel that, with all the different standardization activities that are currently taking place, this kind of situation will present itself more frequently. Therefore, it will be essential to have a method that works even in these delicate cases.

6.2.3 Disagreement must be technical

Perhaps surprisingly, all arguments during the assessments have been settled on technical grounds. Disagreements have been about how important a certain feature is, or how difficult it would be to accomplish certain improvements. Developers felt they need not defend their work during an assessment, as the goal is to document the strengths and weaknesses of architecture.

Also, architects are generally open to criticism against the architecture they have produced, provided that they are made duly aware that the outcome of the assessment will help them do their job better in the future. This proved to be particularly important in the second case, when the direct superior of the chief architect was present at the meeting. Management must ensure that the right “climate” is set up before the assessment. Intentionally avoiding some issues may mean overlooking important defects that may become expensive to fix.

6.2.4 Collaboration is essential

The active collaboration of all stakeholders (especially the chief software architects, but also developers, testers, etc.) in scenario elicitation, categorization and filtering has been extremely helpful. The participant felt that it has been an essential success (or lack thereof) factor for the experiments. Especially for the second case study, a few of the stakeholder complained that the scenario classification and engineering phase had been ineffective. This may be due to the fact that most stakeholders were familiar with the requirements. A number of them expressed

6.2.5 Management must sponsor

Especially in large, schedule-oppressed organizations, activities (such as assessments) that do not produce any immediately visible benefit for the software may be regarded as low-priority. Therefore, high-management awareness and sponsorship is mandatory in order to get the necessary commitment and resources from the development departments.

6.2.6 Follow-up must be ensured

A convenient way to avoid misunderstandings *a posteriori* has been to make sure that all participants agree with the findings. An assessment report that documents the common understanding was produced in both case studies. We have found it convenient to prioritize the issues contained in the report according to their importance, as evaluated by both the chief architect and (when needed) other assessment participants. In both case studies, the report was circulated among the participants for inspection, and went through a formal approval process, just like any other architectural document.

Ideally, an assessment report should contain a sort of diary of the assessment, with a focus on the major qualitative attributes of the architecture that have emerged during the assessment. Any shortcomings should result in some architectural improvement activities, as has been in both our case studies [16].

We had little control over architectural improvement activities in the first case study. However, as already mentioned, in the resource access control case, we ensured that the assessment would be followed up duly by adding the identified problems and issues to the “Open Items” section of the architectural description document.

6.2.7 Preparation is essential

The participants also felt that doing the big bulk of preparatory work offline before the meeting helped avoid overlooking a number of facts and scenarios that were afterwards deemed relevant. It also gave a common understanding of the system under assessment, and made sure that the (different) business goals of all involved partners could be equally fulfilled.

The thorough preparation made it possible to allocated a time range of 15 to 40 minutes to each scenario, depending mainly on the complexity. A certain degree of flexibility was allowed during discussion, which the participants felt helped discuss certain problematic items with the necessary depth.

6.3 Lessons learned

The participants to both case studies were asked to list the main benefits of architectural assessment. The majority of them listed one or more of the following items.

6.3.1 Communication is made easier.

Participating in an assessment allowed people who normally work in different departments to gather together and have focused technical discussions; it helped have a common view about the architecture in line with that of the chief architect. This is particularly important in a community of developers located in more sites. It also provided a technical justification to for people to speak out loud, and point out problems clearly without fears of compromising a career. Even though in northern Europe’s open work environment this is not usually regarded as a major problem, it could turn out to be a key factor when applied to different cultures and organizations.

6.3.2 Documentation is improved.

Especially in the resource access control case, the system was not well documented before the assessment, and there was still a certain degree of uncertainty about the requirements and behavior of the system. Incidentally, the assessment helped originated a few action points for architectural documentation improvement.

6.3.3 Implementation work can be scheduled more carefully.

Assessments give the best possible estimation (at that time) of the amount of work needed on the architecture before implementation can start. Imaginably, the system manager with the help of the chief architect must continuously revise such estimation.

6.3.4 Architectural shortcomings can be identified and corrected soon.

Naturally, this is the main benefit of assessments. The central role of the chief architect in this task was evident in both cases, although there were instances in which major defects were traced thanks to other stakeholders, thus proving the effectiveness of the brainstorming technique.

6.4 Needed improvements

A useful remark some participants made is that architectural assessment would be easier and more efficient if good tools that support management and categorization of scenarios, visualize the architecture or calculate architecturally significant metrics existed. Performing such activities manually is error-prone, and, especially for complex architectures, sometimes impossible or just too expensive. However, at the moment no commercially available tool (other than word processors or spreadsheets with macros) seems to fulfill the description.

Another aspect to be improved is the scenario collection phase, which, the participants remarked, could have been more systematical and better guided. I was the only person who took part in both case studies. However, the experience and insight acquired in the first case study helped me perform and better guide the scenario collection phase in the second one. Hence, the problem may only be an assertion of inexperience.

Another problem that the stakeholders noted (especially in the second case study) concerned the roles in the assessment. When arranging the assessment, we assigned roles to people that were as close as possible to their real jobs. We thought this would make it easier for them to express their opinion. However, some of the participants pointed out that they would have been able to express less biased opinions had they been assigned to a different role. They also expressed concerns that, since not all stakeholders were present, some of the responsibilities were not taken by anyone in the meeting, and therefore some of the requirements may have been overlooked completely.

Overall, a weird problem seemed to emerge from the fact that the stakeholder tended to value more the so-called side benefits of assessment (improved communication and awareness of the architecture, better documentation, etc.) than the main one (identification of architectural shortcomings). The problem emerged especially in the second case study. It may partly depend on the fact that the assessment team members had strong control over the architecture, and therefore were aware of most shortcomings. Instead, having the chance to gather together to discuss technical issues for a full day does not happen often in tight-schedule environment. The beneficial consequences of such discussion may have been appreciated more than those of the assessment itself.

6.5 Assessments are not all

Trivial as it may sound, I claim that, especially in an industrial environment, stakeholders should always be made aware that architecture description alone can guarantee neither quality nor functionality of the final software product. Wrong design choices, bad management, poor implementation, insufficient testing and difficulty of communication between team members could drive to failure, even in presence of a good architecture.

Therefore, assessments should not be limited to the early phases of the software lifecycle. They could be devised as a regular initiative that helps keeping the software development work up to the customers' and users' expectations. Our experience in this approach is still limited, and more experiments are needed before its effectiveness can be proved.

7. RELATED WORK

Architectural assessments have been the subject of several research papers.

Kazman et al. [7] describe their method for assessment, as well as experiences from a number of case studies. Interestingly, the list of experience issues I gathered resembles the one they published, containing "proper (software) description level", "enhanced communication" and "improvement of traditional metrics" among the main benefits of the assessment experiments they performed.

Bengtsson et al. [8] propose a method for assessing modifiability of software architecture. While modifiability is a similar quality attribute to evolvability, it does not explicitly concern issues such as derivation of new products from a family architecture. Their method is similar to the one I applied, in that it is based on scenario elicitation and walkthrough.

Bengtsson and Bosch [9] also investigated the subject of assessing software architecture for maintainability. The problem of maintainability mirrors that of evolvability, and mainly concerns large software systems that need to be evolved in small steps for a long period of time.

In a newly published book, Clements, Kazman and Klein [10] illustrate several methods for conducting architectural assessment. The book has been published only a few days before writing, and constitutes new material for me.

Kazman, Carriere and Woods [11] discuss the subject of scenario-based software architecture analysis, with particular focus on qualitative attributes. Their paper is the first one that proposes scenario-based approaches for all phases of software architecture development.

Kazman, Klein and Clements [12] have also presented an application of their ATAM method to real-time system.

Concerning industrial applications, Ferber et al. [13] discuss the experience they gathered from applying the ATAM method to the automotive domain. Interestingly, this paper seems to suggest that the most prominent results of the application of architectural reviews is not the identification of defects, but rather the improvement of documentation and a better understanding of the architecture. The results of this experience seem to be in line with mine, thus providing more evidence to the fact that "side benefits" of assessment seem to be at least as important as the tracking of defects.

Finally, Galal [14] proposes the application of scenario-based techniques to software architecture development, with particular focus on the evolutionary aspects of architecturally significant requirements (architectonics). The theoretical approach that this paper establishes poses the foundation for a completely new approach to the discipline of software architecture evolution.

8. CONCLUSIONS

I described the experience gathered from two industrial case studies of product family architecture assessment for evolution. The two case studies differed slightly: in one case, the platform under assessment had already been employed in a few commercial products, while in the other the majority of the software had not been written at the time of assessment. The first case study was a cross-organizational, international experiment on a system of common interest, while the second one was performed entirely inside our organization.

Despite the differences in the setup, both experiments shared the goal of assessing the capability of a software product family architecture to adapt to evolution.

The fact that previously unknown architectural issues were identified in both case studies provides an indication of the effectiveness of the assessment practice. Comments given by stakeholders constitute evidence of other important side benefits of architectural assessment and point to a number of improvements in the methodology, which is still incomplete and not validated.

Following the experiences hereby described, we have started an architectural assessment initiative inside Nokia that we intend to continue in the future. We plan to gather more experimental data and perform additional research on the topic, with the aid of various research approaches. This way, we aim to contribute to the validation of the assessment practice in the context of a multi-site, industrial organization that develops large, complex systems.

9. ACKNOWLEDGMENTS

Thanks to the ESAPS project team (in particular to Tapio Tallgren) that took part in the first case study, and to Stefan Baggström, Andy Turner and Christian Del Rosso for the invaluable help, funding and support with the second one. Thanks to Anne Volmari and especially to Jan Bosch for the hard review work and good advice for this paper. The work behind the first case study has been part of the Eureka Σ!2023 Programme, ITEA (project 99005, ESAPS), while part of the research work connected to the second case study has been part of the Eureka Σ!2023 Programme, ITEA (ip00004, CAFÉ [15]).

10. REFERENCES

[1] Bosch, J. Design and Use of Software Architectures. Addison-Wesley, 2000.

[2] Jazayeri, M., van der Linden, F., and Ran, A. (eds.), Software Architecture for Product Families. Addison-Wesley, 2000.

[3] ESAPS project official web site. http://www.esi.es/esaps/

[4] Kazman, R., Klein, M., Clements, P. ATAM: A Method for Architecture Evaluation. Technical Report CMU/SEI-2000-TR-004, Software Engineering Institute, Pittsburgh, PA, 2000.

[5] Riva C. Reverse Architecting: an Industrial Experience Report, in Proceedings of the 7th Working Conference on Reverse Engineering WCRE2000 (Brisbane, AUS, November, 2000).

[6] Symbian technology web page. http://www.symbian.com/technology/technology.html.

[7] Kazman, R., Abowd, G., and Bass, L. Scenario-Based Analysis of Software Architecture, in IEEE Software (November 1996), pp. 47-55.

[8] Bengtsson, P.O., Lassing, N., Bosch, J., van Vliet, H., Analyzing Software Architectures for Modifiability, Blekinge Institute of Technology Research Report 2000:11, ISSN: 1103-1581.

[9] Bengtsson, P.O., and Bosch, J., Assessing Optimal Software Architecture Maintainability, Proceedings of the fifth European Conference on Software Maintainability and Reengineering, September 2000.

[10] Clements, P., Kazman, R., Klein, M., Evaluating Software Architectures: Methods and Case Studies, Addison-Wesley, 2001.

[11] Kazman, R., Carriere, S. J., Woods, S. G., Toward a Discipline of Scenario-Based Architectural Engineering, Annals of Software Engineering, Vol. 9, 5-33, 2000.

[12] Kazman, R., Klein, M., Clements, P., Evaluating Software Architectures for Real-Time Systems, Annals of Software Engineering, Vol. 7, 1999, 71-93.

[13] Ferber, S., Heidl, P., Lutz, P., Reviewing Product Line Architectures: Experience Report of ATAM in an Automotive Context, the Third International Workshop on Product Family Engineering, Bilbao, Spain, October 2001.

[14] Galal, G. H., Scenario-Based Software Architecting. In: I. Borne, S. Demeyer, & G. Galal (Eds.), 13th European Conference on Object-Oriented Programming: ECOOP'99, 13-18 June 1999. Workshop W4: Object-Oriented Architectural Evolution. Lisbon, Portugal. LNCS 1743: 68.

[15] CAFÉ, http://www.itea-office.org/projects/cafe_fact.html

[16] Booch, G., Conducting a software architecture assessment, Rational white paper, see http://www.rational.com/products/whitepapers/391.jsp