Evolutionary Architecting of Embedded Automotive Product Lines: An Industrial Case Study

Jakob Axelsson

School of Innovation, Design & Engineering
Mälardalen University
SE-721 23 Västerås, Sweden
jakob.axelsson@mdh.se

Dept. 94100 HC1S
Volvo Car Corporation
SE-405 31 Göteborg, Sweden

Abstract

In the automotive industry, embedded systems and software play an increasingly important role in defining the characteristics of the vehicles. Both the vehicles and the embedded systems are designed as product lines, and two distinct architecture processes can be identified. The revolutionary process develops the architecture of a new product line, and focuses on abstract quality attributes and flexibility. The evolutionary process continuously modifies the architecture due to changes, such as additions of new functionality. In this paper, the evolutionary process is investigated through a case study. The study reviews a number of changes to an existing architecture, observing the cause of the change, what quality attributes were considered, and what technical aspects were included. It is also analyzed how the interplay between the two processes can be improved through systematic feedback about what evolution actually takes place.

1. Introduction

The automotive industry has in recent years witnessed a dramatic increase in functionality based on electrical and electronic (E/E) components. According to some sources, 80% of the innovation in a car in the premium segment comes from the electronics [12]. Many of the advances seen in the automotive industry, for instance in areas such as safety, emission control, comfort, and quality, would have been impossible without the use of advanced computer-based control systems. Also, electronics can be used to reduce cost, when expensive mechanical components are replaced by cheaper electronic controllers. This has led to a situation where a modern car contains a large number of Electronic Control Units (ECUs) connected via a number of communication networks and running complex distributed software applications.

Although the electronics has a great potential to improve vehicles, the systems are becoming increasingly complex and that makes the engineering more and more difficult. The functions are in many cases safety critical, requiring special care to handle any circumstances that may possibly occur during operation. At the same time, the system has a very long operational life time where only sporadic maintenance can be assumed. The products are mass-produced, so assembly must be very efficient. Many vehicles are consumer products where the price must be kept low.

Due to varying customer demands, but also due to different legal requirements in the countries where the product is being sold, many variants of the product must be designed and verified. To handle this, and to be able to have reasonable production volumes of each system, the Original Equipment Manufacturers (OEMs) usually employ a product line strategy in which many components are common across a range of products. This platform is refined over many years, and each vehicle has to cope with an extensive amount of legacy both in components and in the overall structure.

With this multiplicity of products and variants, the architecture is becoming very important and is a source of increasing interest from the OEMs. An *architecture* can be defined as the fundamental organization of a system embodied in its components, their relationships to each other, and to the environment, and the principles guiding its design and evolution [7].

This paper is based on the observation that the revolutionary architecting process for the platform is quite different in its nature from the evolutionary architecting of the different products based on the platform. It presents empirical data from a case study at an automotive OEM on how the evolutionary architecting is actually carried out, which is a topic where little evidence has been presented before. It further discusses the interplay between the two architecting processes, and how it can be improved.

978-1-4244-4985-9/09/$25.00

The paper is structured as follows. In the next section, the architecture processes are discussed further, and the theoretical framework is presented together with the research questions. Then, in Section 3, previous research related to the paper is reviewed. In Section 4, the case study organization is discussed in more detail, and in the following section the findings are presented. In Section 6, the results are discussed and suggestions for improvement are described, and in the final section, the conclusions are summarized together with some ideas for future work.

2. Problem definition

In this section, we will discuss the architecting processes in more detail, and what research questions they lead to. But before doing so, some more information is needed on how the automotive OEMs work with product lines to understand the challenges that automotive companies face in architecture development.

2.1. Context

An automotive OEM usually produces a number of different *car models* with distinct names and body styles. These products are organized into *product lines*, where the car models within a car line share a substantial number of components. Sometimes the term *platform* is used to describe this sharing. Within each car model, many *variants* are produced to reflect individual customer choice of features, demands on different markets, etc.

The product range is however not static. The typical production life of a car model is somewhere between 6-8 year, after which the production is discontinued. Often (but not always) a totally new car with the same name plate replaces the old one. However, not all the models are updated at the same time. For development capacity reasons and also to get maximum effects out of marketing, only a few models are replaced each year. This means that the process of moving all models in a car line to a new platform takes several years, and there is therefore usually a mixture of products based on old and new platforms in production. To handle the dynamics of the product range, the development work is organized into *projects* for platforms, new cars, and model year updates of existing cars.

The E/E system follows the structure of the products, and there is one architecture for each platform. However, since the E/E system is not so strongly connected to the car's size or body style, there is often an ambition to minimize differences between the E/E platforms. The dynamics of the product portfolio affects E/E system development greatly. A common scenario is that a new feature is developed as part of a new car model project. The question then immediately arises whether that feature can be carried over to other models on the same platform, or even carried back to the other car line or to car models of the same car line but that are still on an older platform.

For a good introduction to how the automotive industry works with software and electronics, see [4].

2.2. Theoretical framework

When a new platform is developed, there is an opportunity to do a major revision of the architecture. Changes that are typically introduced only at this time are a new communication concept, a different structure of the communication networks, or new basic software in the ECUs of the vehicle. Between these revolutionary steps, modifications such as the addition of a new ECU, a reallocation of some application software between two ECUs, or changing the connection of a sensor from one ECU to another, often occur.

Some of the differences between the revolutionary and evolutionary architecting processes (RAP and EAP) are:

- RAP is done rarely as a defined activity or project, perhaps once every 5-10 years when a new platform is introduced and each time with a duration of a few years. EAP on the other hand is an ongoing process all the time.
- RAP deals with the architecture as a whole, considering all the functions and systems together. EAP usually deals with changes to a singular, or a few, functions or systems within an existing framework.
- RAP tries to dimension an architecture that can support many (yet unknown) changes as smoothly as possible for a long time, whereas EAP tries to implement a specific and concrete change in a specific architecture as efficiently as possible (while trying to assure that the resulting architecture still remains as flexible to future changes as possible, although this aspect is often less explicit in practice).
- RAP tries to predict future requirements, which is a speculative activity dealing with abstract information. One of the most important parameters is the expected rate of change which dimensions the flexibility needed. EAP deals with concrete requirements, functions and systems. This means that RAP must deal with uncertainty to a much higher extend than EAP.

With these differences pointed out, it should also be said that there are situations where some aspects of revolutionary nature is also conducted within EAP,

simply because there is a need that was not foreseen at the time of the previous instantiation of RAP.

The automotive industry is currently heavily influenced by Japanese practices, many of them originating from Toyota and in the western world often presented under the label "Lean." One of the most cited aspects of Lean is *kaizen*, which stands for continuous improvement activities. The automotive industry is thus very used to the idea of evolutionary development. However, Lean also contains the idea of *kaikaku*, meaning revolutionary change, and this has not been widely recognized in the western automotive industry, nor has the interplay between the two been considered.

Within software development, the relation appears to be the opposite, with much focus on new development, and less on continuous improvement.

2.3. Research questions

This paper primarily addresses the following research questions, in the context of companies developing embedded systems as part of their products:

1. How is the evolutionary system architecture process carried out in practice?
2. What is the interplay between evolutionary and revolutionary architecting?
3. What are the potential areas of improvement in the architecting processes?

We hope to learn more about the first question by simply studying how it is done at a relevant company, and see what factors are considered in that process. If we could in this way improve the understanding of what factors are involved in the evolutionary process, this could be compared to how the revolutionary process is carried out at the same company, to give ideas about how they interact and can be ameliorated, and thereby answering the second and third questions.

3. Related Research

In this section, we provide a review of some related work on the EAP and its relation to the RAP. The section is divided into three parts, where the first contains theoretical descriptions of architecting methods, the second contains empirical evidence of the industrial application of architecting, and the third summarizes the contributions of this paper in relation to the existing publications.

3.1. Theoretical models

One of the most well-known methods for (software) architecture development is the Architecture Trade-off and Analysis Method (ATAM) [10]. In this method, quality attributes are introduced to assess the benefits of an architecture proposal, and scenarios are used to clarify architectural requirements. The approach is most suited for either new development or substantial revisions of legacy systems, and thus corresponds best to RAP. However, it gives little insight into how to perform the step-by-step refinement of EAP.

A better basis for describing EAP is given in [9] which views architecting as a set of architectural design decisions. This corresponds well to the sequence of design decisions made in the continuous evolution of the architecture. The paper presents a structure for describing the design decisions, consisting of a *problem* (the goal to solve), a *motivation* for the problem, a *cause* for the problem, alternative *solutions* to the problem, a *decision* capturing a number of *trade-offs* to select a solution, and an *architectural modification* which modifies a *context*.

In [3], an architecture design method based on evolution and transformation is presented. However, it focuses on the evolution that takes place while developing a new release of an architecture, rather than the evolution of an architecture over several releases.

Evolution of software product lines is also the topic of [16], which points out that much of the research on product lines concerns the RAP rather than EAP. It also observes that it is difficult in practice to predict what future evolution will occur. The paper discusses techniques for improving variability, but these are software focused and less relevant for automotive systems engineering.

One of the few research contributions that deal with the architecture of embedded systems (with examples from the automotive domain) rather than just software is [6]. It introduces a hierarchy of decision levels, where top-level decisions would correspond to those made in RAP and low-level decisions to those of EAP. An elaborate analysis procedure for architecture alternatives is also described, but again it is focused on the initial development of an architecture rather than the evolution.

The relation between RAP and EAP is discussed theoretically in [1], which identifies the need for both processes and that an exploration of the evolution history is important to understand when revolution is needed.

The application of kaizen to software product lines is discussed in [8], which provides many interesting ideas how to systematically improve both the product line and the work standard. However, the focus is more on maintaining the software core assets of the product line than on developing the architecture, which makes the concrete results less applicable for our purposes.

3.2. Empirical investigations

A number of case studies on software evolution have been presented. In [2], a case study based primarily on interviews at two companies using software product lines is reported. Although the focus is on the evolution of the software assets rather than the products, the authors note that this evolution actually often takes place within product projects rather than as dedicated activities. Another issue was the difficulty to decide when to split off a product from the product line, i.e. when to perform a revolutionary step.

One of the two companies in that study reappears together with a third company in [15]. Again the focus is on evolution of the software assets and not the products. A difference is that they have actually studied the evolution of the software over several releases. One observation is that at some point in time, revolutionary revisions are made in an otherwise evolutionary process, but the authors find it hard to pinpoint the cause for this. The paper also describes a set of useful categories for describing evolution.

Two industrial case studies on teams assessing software architecture for evolution are presented in [13]. However, what actual evolution takes place in these systems is not described.

Yet another case study is presented in [5], which describes continuously evolving software and the issue of detecting when there is a need for re-architecting activities. The focus of the case study is however on a comparison of three approaches for assessing the evolvability of the software architecture, rather than a study of the actual evolution process as is.

3.3. Contribution

Much of the literature thus mainly describes the RAP, where a new system is designed. There is also a focus on software architecture, which is less relevant for automotive OEMs who focus on system architecture and leaving much of the software details to suppliers.

The contribution of this paper is three-fold. Firstly, it contains an empirical study of the interplay between RAP and EAP. Secondly, it is not restricted to software, but studies embedded systems considering both hardware and software aspects and the relation to the overall product. Thirdly, it is an empirical study of actual architecting for automotive E/E systems. None of these have to our knowledge been reported before.

4. Case study description

Since this research enters a new area, we did not have enough a priori information to form any clear hypotheses or theories around the research questions. Instead, we choose to conduct an exploratory single-case study [18] at an automotive OEM to gather more information. The methodology used is both quantitative based on a classification of events, to get an idea of their magnitude and frequency, and qualitative to be able to study underlying causes.

4.1. The company

The case study was carried out at Volvo Car Corporation (VCC). The company has its headquarters, including product development and many other functions, in Gothenburg, Sweden. The company is a producer of premium cars, with special focus on safety, environment, and quality. At the time period focused in the study, it had approximately 25,000 employees and manufactured and sold close to 500,000 vehicles each year worldwide. It is a subsidiary of the Ford Motor Company (FMC) since 1999, and had at the time close co-operation within FMC primarily with Ford of Europe in Germany and Jaguar-Land Rover in the UK. For these brands, VCC had a leading responsibility for the E/E architecture.

As described in Section 2, the automotive industry works with car models, platforms, model years, and variants. In the case of VCC, there are approximately 10 different car models (i.e. cars with different name plates, such as Volvo S80 or Volvo XC90). These products are organized into two car lines for small and large cars. Within FMC there is also a cross-brand sharing of platforms, so that a Ford car and a Volvo car can share some components or technologies.

Usually at VCC, there is one project for each new car. In addition, there is one specific project when a new platform is developed, and this project usually runs in parallel with the project for the first car to use that platform. The model year changes also run as projects but usually there is only one model year project per platform that takes care of all the name plates on that platform.

At VCC, it is typical that revolutionary changes to the E/E architecture occur when a new platform is developed. Most often, it is the large car platform that carries the largest changes, since these cars are usually richer in features. The small car platform development is usually a revolutionary step compared to the previous small platform, but at the same time often includes architectural solutions from the current large platform. The new car model projects can sometimes carry revolutionary changes to a limited segment of the system but evolutionary changes or pure carry-over to other parts. In the model year projects, the changes are almost entirely evolutionary.

4.2. Unit of analysis

The unit of analysis was the E/E systems engineering department at VCC. At the department, a weekly semi-formal meeting plays the role of an Architecture Change Control Board (ACCB). At the meeting, issues are discussed that affect the E/E architecture of one or several cars, and the architects can get advice on what solution to choose. The architects working on different projects can also co-ordinate their actions to avoid that the architectures of different cars or platforms drift apart unnecessarily, and avoid conflicts over resources in the architecture that are needed by changes processed in parallel.

A fundamental role of the ACCB is to ensure that the best trade-offs are made in the evolution of the architecture. Therefore, not all architectural design decisions are brought to the attention of the meeting. If the changes appear uncontroversial, meaning that no trade-offs are needed, the individual architects can make the decision.

4.3. Data collection

The case study was carried out by reviewing archival records from the ACCB meetings. The records include meeting minutes, but also investigation reports and presentation material that were associated with the item, and complemented with discussions with people involved and personal experiences of the author.

We choose to study all the meeting items treated at the ACCB during the calendar year 2006. During this year, VCC launched a new large car platform, and the architecture work for that product line was already completed. Therefore, the ACCB was expected to be focusing on evolutionary changes to the existing platforms during this time period. At the same time, investigations were already starting, in the form of an advanced engineering project, on what the next revolutionary step would be. Although that work was in an early phase, it could be possible to get some ideas about the interactions between the two processes.

Often, a certain issue was not resolved directly at an ACCB meeting, but an investigation was started, and reported back at a later time. Sometimes, several iterations at the meeting were needed. We therefore followed all the items until they were concluded, even if this in some cases was after the end of 2006.

After an initial screening and removal of some irrelevant items, the total material included 31 items. Since most of the items were discussed on an average 4.1 times at the meeting, there were a total of 128 meeting minutes to analyze, together with supplementary material. Disregarding vacation brakes and a few extreme items, the average duration of an item was 11 week. Thus an item was typically brought back for review or status report every 2-3 weeks.

4.4. Case study protocol

The raw data was in a free, unstructured form and therefore a way of structuring the data as part of the analysis was necessary. This was captured in a case study protocol. Based on the research questions, we decided to focus on four areas:

1. *Problem*. What was the reason for bringing up the item? This relates to the process inputs in that it captures what triggered the change.
2. *Trade-off*. What quality attributes of the architecture were evaluated to conclude how to handle the item? This relates to what trade-offs were made within the process.
3. *Solutions*. What technical areas of the architecture were affected by the item? This relates to the process outputs, which are decisions for change of the current architecture.
4. *Decision*. What was the actual decision made? Was the change proposal approved or not?

These areas correspond to a subset of the information used to describe architectural decisions in [9].

The data was first recorded as free text, where only the information necessary for the study was included. The information in the text was further analyzed to find common traits between different items and finding ways to structure the material further. Based on this structuring, a tabular summary was constructed that could be used to calculate some statistics. These statistics were used as a way of discovering patterns, and should not be seen as research results themselves.

5. Findings

In this section, the findings of the case study are classified. First, the reasons for change are discussed, followed by the affected attributes, the technical aspects involved, and what decision was made.

5.1. Reasons for change

The first question we asked was why the change was initiated. After studying the data, five main categories emerged (similar to those used in [15]):

1. *Integrate new electrical function or system*. In this case, a totally new feature was being developed, that did not exist in any Volvo cars before. The items concerned how to integrate this feature into the existing or planned

architecture in the best way. 12 items (39%) fitted best into this category.

2. *Integrate modified electrical function or system*. In this case, a similar feature already existed in some Volvo cars, but either the functionality was changed or the system solution was modified. An example would be that a part of the functionality was suggested to be re-allocated from one ECU to another. 7 items (23%) fitted best into this category.
3. *Integrate carry-over system*. Here, an existing system from another car model was to be integrated. Typically, it would be a system from an external supplier, that should be modified as little as possible. 3 items (10%) fitted best into this category.
4. *Reduce cost*. Product cost is very important to high volume automotive companies, due to its large influence on profit. Therefore, automotive companies continuously strive to improve their products by finding cheaper ways of implementing functionality. Still, only one item (3%) fitted best into this category.
5. *Provide strategy or future protection*. Whereas the first three categories were usually expressed in terms of visible customer functions, the last category was described in terms of internal concepts in the architecture, such as networks, electrical load management, or configuration data parameters. These are often cross-cutting concerns that affect many customer functions, and the items were brought up because responsible persons were starting to see bottle-necks in the implementation. 8 items (26%) fitted best into this category.

The data did not reveal any significant differences in how many weeks were needed for the meeting to conclude an item depending on the initial cause.

5.2. Quality attribute impact

Once an item has been brought up, it is interesting to study how it was evaluated. In architecture methods, it is often advocated to use quality attributes to guide development [10]. This approach has to some extent been adopted by VCC when it comes to the revolutionary architecture development, and a structure of important quality attributes has been defined. We therefore decided to use this structure to investigate which quality attributes were considered in the different items in the study as an indication of what trade-offs were made. (It should be mentioned that the attribute structure is evolving, and exists in several versions within the company, but the one we refer to here was the most recent at the time of the study.)

The attributes are divided into three main categories:

1. *Cost attributes*. This is focused due to its importance to the business. The cost category includes: (a) product cost; (b) development cost; (c) production cost; (d) investment costs; (e) operating costs; and (f) maintenance cost. (The latter two refer to the operation and maintenance of the individual products by the customer, and not the maintenance of the engineering artifacts within the company.)
2. *Product attributes*. This category consists of attributes of the product as shipped to the customer. It contains four subcategories:
   i. *Energy usage*. This category includes: (a) energy efficiency; (b) power consumption during normal operation; (c) operational time during parking (which essentially relates to battery capacity); and (d) physical weight of the system, since that has an effect on the overall energy usage of the vehicle.
   ii. *Communication performance*. Includes (a) throughput and (b) responsiveness of the communication networks, but also (c) interoperability, i.e. being able to exchange data with outside entities such as factory or service equipment.
   iii. *Dependability*, which is decomposed into: (a) availability; (b) integrity; (c) reliability; (d) safety; and (e) robustness.
   iv. *Integrability*. This has to do with how well the embedded system is integrated into its environment. It contains: (a) physical fitness (which relates to packaging the components into the available space); (b) styling compatibility (for parts visible to the customer); and (c) EMC.
3. *Delivery process attributes*. The last category includes attributes that are important to the company when it comes to efficiently developing and refining the products. It has two subcategories:
   i. *Development feasibility*. These are the attributes that capture how well the architecture can evolve efficiently, and includes: (a) configurability; (b) scalability; (c) flexibility; (d) complexity; (e) extendability; (f) time to market; (g) commercial efficiency (i.e., how well the solution matches what suppliers can offer); and (h) testability.
   ii. *Manufacturing and service feasibility*. This consists of the two attributes (a) produceability and (b) serviceability.

In total, the quality attribute structure thus contained 31 attributes at the lowest level.

Although this structure (and variants of it) is and has been in use at VCC, it was not always evident how to map the items discussed at the ACCB to it, and sometimes interpretations had to be made. In most cases each item is related to several of the attributes, and in the 31 items included in the study, the number of attributes considered varied between 0 and 12 (out of the total 31 attributes), with an average of 5.6 and a standard deviation of 3.2. The most frequently considered attributes were:

- Product cost (18 issues, or 58%).
- Responsiveness (12 issues, 39%).
- Configurability (12 issues, 39%).
- Flexibility (11 issues, 34%).
- Power consumption (10 issues, 32%).

Testability was never considered. Operating cost, service cost, energy efficiency, weight, interoperability, and styling compatibility were each considered once only.

There were no significant differences in the number of attributes considered depending on the initial cause for the issue.

5.3. Technical area affected

The last area investigated was what parts of the architecture were affected by the change. Again, an existing structure of categories at VCC was used to classify the different items:

1. *System structure*. This includes: (a) what ECUs the system consists of; (b) what connections exist; (c) the logical dependencies of the customer functionality; (d) how functions are allocated to ECUs; and (e) the mechanical structure of the system (including packaging).
2. *ECU platform*. The basic technologies that are similar in all ECUs regardless of their functionality, including: (a) communication protocols; (b) operating system; (c) diagnostic software; (d) software download support; (e) network management; (f) vehicle mode management; and (g) hardware components.
3. *Energy handling*, including: (a) energy storage; (b) energy generation; and (c) energy management.
4. *Electrical distribution,* consisting of: (a) overcurrent protection; (b) junction boxes; (c) wiring; and (d) ground distribution.
5. *External interfaces* for: (a) communication and (b) electrical power.
6. *Vehicle information management* including: (a) vehicle identification number; (b) vehicle configuration; and (c) tampering notification.

In total, there were thus 24 technical aspects included in the analysis. The number of aspects included for each item varied between 1 and 10, with an average of 3.5 (standard deviation 2.5). The most frequently considered aspects were:

- Which connections there should be between ECUs (18 issues, or 58%).
- Which communication protocols should be used on specific links (13 issues, 42%).
- Which ECUs should be in the system (11 items, 35%).
- How the wiring was affected (9 items, 29%).
- How the mechanical structure was affected (7 items, 23%).

Energy storage, external power interface, and vehicle identification number management were not considered in any item. The logical structure, hardware platform, over-current protection, junction boxes, ground distribution, and tampering notification were only discussed in one item each.

The data did not show any significant difference on the number of aspects considered depending on what the initial cause was for the item to be brought up. There was however a weak positive correlation between the number of quality attributes considered and the number of technical areas affected (Pearson correlation coefficient $r = 0.50$, with $p < 0.005$). A possible interpretation is that the more areas that are involved in the solution, the more quality attributes need to be considered in the trade-offs.

5.4. Decision

Since the ACCB functions as a change control board, the fundamental decisions were expected to be either approval or rejection of the change request. However, for rejections some variants exist, and in some cases the decision was not clear. We ended up with the following categories:

1. *Approval*. In 16 cases (52%) some kind of change was decided. Many items contained several alternatives, and it was not always the initially favored alternative that was selected, but at least some change was approved.
2. *Rejection*. In 7 cases (23%) no change was approved.
3. *Request withdrawal*. In one case (3%) the change request was withdrawn before the ACCB had concluded its processing, due to factors outside the E/E system.
4. *Unclear decision*. In 3 cases (10%) the archival records were incomplete and it was not clearly stated what the outcome was.
5. *Transfer of item*. In 4 cases (13%) the issue was handed over to another team, since other aspects than the E/E architecture needed to be resolved first.

Interestingly, none of the items where the initial cause was to integrate a new electrical function or system

was rejected (although in a few cases, the decision was unclear or the item was transferred). This could be an indication that the organization assigns a high value to functional growth, and the architects strive to implement all the new functions requested.

6. Discussion

Based on the findings presented in the previous section, we will now discuss what conclusions can be drawn from these, and what other observations of more qualitative nature were made during the case study. The section contains one part related to each of the three research questions described in Section 2.3.

6.1. Evolutionary architecting in practice

The items that are brought up at the ACCB depend very much on the current state of the architecture, and what bottlenecks exist. If, for instance, communication capacity is a bottleneck, this will appear often in the trade-offs and be a major issue. If that bottleneck is resolved, e.g. in the next revolutionary step by adding a faster communication bus, there will be an overcapacity. Then, the architects do not need to bring communication issues to the ACCB for trade-offs, and hence other issues will dominate the meeting agenda. Therefore, it can be expected that a repetition of this study at VCC at another time would yield a different result regarding the frequency of quality attributes and technical solutions that are discussed.

When looking into the items in more detail, it is striking that much of the discussion relates to technical details at a much lower level than what one normally expects when dealing with (revolutionary) architecture. In a way, it is natural because the work in the EAP is based on an already completed architecture where all the details are available. But it probably also reflects the way that the OEMs interact with their suppliers. It is sometimes the case that a request for new functionality is accompanied with an existing solution from a supplier which has developed a similar system for another OEM, and for that solution many details are also available. The architecting in that situation amounts to finding the best way of integrating an existing solution into an existing architecture, while doing as few changes as possible to either one.

For some technical areas, it is evident from the meeting records that the company has clear routines for what aspects should be analyzed. Network communication is one such topic, where VCC has for a long time had an internal competence and fairly well-defined processes. In other areas, it is less evident how to analyze the situation, and the meeting minutes reflect a process which is trying to define itself. The ACCB did not work with pre-defined checklists for what aspects should be considered, so there is a risk that some aspects were missed in certain items due to ignorance or neglect.

An interesting observation was that there were so few changes driven by product cost optimization. If one should speculate, this might be because the engineers refrain from cost cutting actions that have a large effect on the architecture. If there is an effect on the architecture, it is likely that the change will cause further alterations in other functions or components due to interface modifications, and this will lead to more verification efforts being needed. The risk of doing such changes in terms of potential quality problems might simply be too high to justify the cost saving. On the other hand, when it comes to the trade-offs, the product cost is clearly present in the analysis, but there is not a systematic process or well-defined model to weigh cost against different benefits, such as maintaining flexibility for future evolution. (This supports the observation in [17] that there is a lack of model to evaluate business value when choosing the architecture.)

It is also striking that organizational issues often are brought up as part of the technical discussions. When a new solution is introduced, it is not clear who in the organization should take responsibility for that, and release all the appropriate specifications. This is a consequence of the fact that the organization of product development at VCC (and many other OEMs) reflects the current design of the vehicle, but architectural issues are often cross-cutting, and hence affect many parts of the organization. Our investigation thus reaffirms the observation in [11] that architecture, processes, and organization are strongly interrelated.

6.2. Evolution versus revolution

As mentioned in Section 4.2, an advanced engineering activity was starting up at VCC at the time of this study, to look at the next generation E/E architecture and thus reflecting the RAP. It is therefore interesting to discuss similarities and differences between the issues considered in that project versus those treated at the ACCB as part of the EAP.

The next generation architecture (NGA) project did an exercise to try to prioritize among the quality attributes presented above, to see what attributes would drive changes in the architecture. After lengthy discussions, five attributes were prioritized, among them product cost and flexibility which were also among the top 5 at ACCB. The other three attributes were related to energy usage, communication performance, and dependability, which are areas that

also rank high at ACCB. Apparently, there is thus not a clear distinction between the two processes regarding what quality attributes they consider. On the other hand, many of the persons involved in NGA also participate at ACCB, so this could be an indication that the feedback between the two processes actually works, but in an informal way.

As described above, the EAP has access to all the technical details of the architecture, and the discussion tends to be on the corresponding level. In the RAP, the discussion starts at a much higher level, and a lot is unknown or undecided. When looking at the two processes side-by-side in real life, it becomes evident that very different models are needed to describe and analyze the architecture in these two situations.

A curious discovery was that the logical (functional) description of the system plays very different roles in the two processes. In EAP, it is almost not discussed at all (only one item at the ACCB deals with this), whereas the logical view of the architecture has been one of the hottest topics in the NGA project as a means for mastering complexity. We do not have a definite explanation for this. One possibility is that the logical view is not needed when all the details of the solution are available, but plays an important role in the steps towards defining the architecture. Other possible reasons for the differences are that the importance of the logical architecture is exaggerated in the NGA project, or not treated enough at the ACCB. (For a further discussion on how logical architecture is used at VCC, see [14].)

The fact that one fourth of the items at ACCB actually concerned strategic issues reflects the interrelation between the two processes. These items were dealt with in similar ways as would have been done in the RAP, and in two cases the items were actually deferred to the NGA project. However, the conclusion is that the two processes, although distinct in nature, do not represent the only two possibilities, but rather the end points of a continuum. Some issues that are revolutionary in nature still need to be handled within the EAP, simply because they cannot wait for the next revolution to happen. Urgent bottlenecks need to be removed to allow continuous functional growth.

6.3. Process improvement opportunities

As described in the previous section, there is reason to believe that the RAP selected quality attributes to prioritize based on the current trade-offs done within the EAP. This is in a sense a good thing, because current bottlenecks need to be removed in the next platform. However, by looking at the instantaneous situation, there is a risk that important trends are missed. In particular, it would be beneficial to monitor the development of important attributes over time, to be able to extrapolate the rate of change and thereby identify future bottlenecks. This is not done systematically today. In fact, the RAP analyzes the capacity needs based on ideas of future functionality created by the product planning department. These needs are not always correlated with what the company actually can afford to develop, and hence tend to be exaggerated. The true rate of change is a trade-off between the functionality needs and the capacity of the development organization, and both must be considered when dimensioning the architecture for future change. Since a platform's prime objective is to support the evolution over its lifetime, understanding what kind of evolution actually occurs is essential and this can be done by systematically monitoring the EAP. If a scenario-based approach such as ATAM [10] is used in the RAP, data on the actual evolution can be an excellent source during scenario elicitation.

It can be suspected that the EAP is heavily affected by random factors caused by the individuals involved, regarding how issues are analyzed and what issues end up at the ACCB at all. A more systematic approach, with clear rules for what should be brought to ACCB and checklists for how the issues are analyzed would give a basis for a more efficient process and allow a kaizen approach to be applied to improving the EAP itself. It would also provide a basis for a systematic collection of data needed by the RAP, which could itself give the organization earlier warnings when bottlenecks start to arise.

7. Conclusions

After having studied how the EAP is carried out in practice at an automotive OEM, and how it relates to the RAP, our initial assumption that these two processes are in fact quite different has been confirmed. In a previous case study at the same company [17], an issue that came up was the lack of a defined architecting process. With the results of this study, the immediate follow up question is: Which process? The EAP and RAP are so different in nature that it is unlikely to find one process description that would fit them both. If we could improve our understanding of both these processes, the interplay between them could also be ameliorated. It would be a basis for defining what tools, models, and analyses are needed in each of them, and how they can support each other with information.

7.1. Future work

Since this case study was conducted at a single company, it would be interesting to replicate it to other automotive OEMs and to companies in other business sectors, to see what similarities and differences exist and generalize the conclusions. It would also be worthwhile to investigate some of the discoveries of this study in more depth, for instance:

- What kind of feedback would be most useful from the EAP to the RAP, and what metrics can be used to capture that information?
- What are the primary causes for moving from EAP to RAP? What is it that implies the need for revolution?
- Is it possible to discover earlier when the architecture is evolving into a bottleneck situation, in order to remove that in a more limited action than a revolutionary change? Or taking the same question to the extreme: can we remove the RAP and handle all changes within EAP?
- How can we improve the analysis methods used in the EAP to give a higher value to flexibility, and in that way better account for the remaining evolvability potential in the architecture after the change?

References

[1] M. Aoyama. Metrics and analysis of software architecture evolution with discontinuity. In *Proc. of the International Workshop on Principles of Software Evolution*, pp. 103-107, Orlando, Florida, 2002.

[2] J. Bosch. Product-line architectures in industry: a case study. In *Proc. 21st Intl. Conf. on Software Eng.*, pp. 544-554, Los Angeles, 1999.

[3] J. Bosch and P. Molin. Software architecture design: evaluation and transformation. In Proc. *IEEE Conf. on Eng. of Computer-Based Systems*, pp. 4-10, March 1999.

[4] M. Broy, I. Krüger, A. Pretschner, and C. Salzmann. Engineering automotive software. In *Proc. IEEE,* vol. 95, no. 2, pp. 356-373, Feb. 2007.

[5] C. Del Rosso. Continuous evolution through software architecture evaluation: a case study. *J. of Software Maintenance: Research and Practice*, vol. 18, pp. 351-383, 2006.

[6] B. Florentz and M. Huhn. Architecture potential analysis: a closer look inside architecture evaluation. *J. of Software*, vol. 2, no. 4, pp. 43-56, Oct.. 2007.

[7] IEEE 1471-2000, *Recommended Practice for Architectural Description of Software-Intensive Systems*. 2000.

[8] M. Inoki and Y. Fukazawa. Software product line evolution method based on kaizen approach. In *Proc. of the ACM symposium on Applied computing*, pp. 1207-1214, Seoul, Korea, 2007.

[9] A. Jansen and J. Bosch. Software architecture as a set of architectural design decisions. In *Proc. 5th IEEE/IFIP Working Conference on Software Architecture,* pp. 109-119, 2005.

[10] R. Kazman, M. Klein, and P. Clements. *ATAM: Method for architecture evaluation*. Technical Report CMU/SEI-2000-TR-004, Software Engineering Institute, 2000.

[11] S. Larsson, A. Wall, and P. Wallin. Assessing the influence on processes when evolving the software architecture. In *Proc. 9th Intl. Workshop on Software Evolution*, pp. 59-66, Dubrovnik, Croatia, 2007.

[12] G. Leen and D. Heffernan. Expanding automotive electronic system. *IEEE Computer*, vol. 35, pp. 88–93, 2002.

[13] A. Maccari. Experiences in assessing product family software architecture for evolution. In *Proc. 24th International Conference on Software Engineering*, pp. 585-592, Orlando, Florida, May 2002.

[14] D. Selin, H. Svensson, P. Sundsten, A. Wikström, and U. Eklund. A reference architecture for infotainment systems. *SAE Paper No. 2006-21-0013*, 2006.

[15] M. Svahnberg and J. Bosch. Evolution in software product lines: two cases. *J. of Software Maintenance: Research and Practice*, vol. 11, pp. 391–422, 1999.

[16] M. Svahnberg and J. Bosch. Issues concerning variability in software product lines. In *Lecture Notes in Computer Science*, vol. 1951, pp. 146-157, Springer Verlag 2000.

[17] P. Wallin and J. Axelsson. A case study of issues related to automotive E/E system architecture development. In *Proc. 15th IEEE Intl. Conf. on Eng. of Computer Based Systems*, pp. 87-95, Belfast, 2008.

[18] R. K. Yin. *Case Study Research,* 3rd ed. Sage Publications, Thousand Oaks, 2003.