# A Case Study of the Architecture Business Cycle for an In-Vehicle Software Architecture

Ulrik Eklund
*Electric & Electronic Systems Engineering*
*Volvo Car Corporation*
*Göteborg, Sweden*
*E-mail: ulrik.eklund@ituniv.se*

Carl Magnus Olsson
*Software Engineering & Management*
*IT University of Göteborg*
*Göteborg, Sweden*
*E-mail: carl.olsson@ituniv.se*

## Abstract

*This paper presents the theoretical and practical benefits from a case study using a the Architecture Business Cycle to understand the management of software architecture at an automotive manufacturer. The study was done to prepare for architectural changes driven by new technology and in the automotive business environment.*

*Our results show that the architecture business cycle worked well in defining the theoretical context for the study after some modifications; the architecture had to be precisely defined in the interview situation to gain more useful data rather than broad generalisations. Further contributions of the study were a deeper understanding of role of the architecture and it's position among other artefacts in the organisation, and an increased focus on architectural issues in management meetings. The study also indirectly affected a subsequent re-organisation.*

## 1. Introduction

As all other car manufacturers, Volvo Car Corporation (VCC) is facing tough times and there is a strong demand to develop in-vehicle software with shorter lead times and improved quality. Several authors have identified that one key element to accomplish this, as well as handle the increasing complexity that follows the ever-increasing feature content, lays in the *establishment of a software architecture* [1]–[4]. For example, Broy states "The enormous complexity of software in cars asks for an appropriate structuring by architectures in layers and levels of abstraction" [3]. But not just any architecture will meet the challenges described, the architecture in question must address the relevant business forces and the architecture must actually be used as a guide for the software development in the organisation.

The first step towards improvement in working with architecture in an organisation is to capture the present situation, i.e. to identify the actual forces shaping the architecture, how well the architecture is used, and if the architecture actually helps in addressing the present challenges. In this particular case we wanted to capture how the Electronic and Electric Systems Engineering (EESE) unit at Volvo Cars viewed the present software architecture, originating from 1998, and how the architecture affected the work of the developers within the unit. Of particular interest to the architect running the study were the business forces and feedbacks involved in eventual implicit decisions concerning the architecture and how homogeneous the view on software architecture was among people working at the EESE unit.

In order to get that understanding, we performed the case study presented in this article. We choose to base the case study on the Architecture Business Cycle, originally presented by Bass et al [5], with the goal this would help identify potential areas for future study and improvement when developing software. As a side benefit we also draw some conclusions about the applicability of using the architecture business cycle to understand the role of 3 partial scenarios of a software architecture in the automotive industry in practice.

The contributions from this study are a procedure to capture an instance of the general architecture business cycle [5]–[7], with some modifications to the cycle to keep a manageable scope of our study, and the conclusion that the architecture business cycle works well as a theoretical framework for practical studies. We find the main contribution to be a rich empirical

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

insight in the role of software architecture in the automotive industry through the theoretical lens of the architecture business cycle.

Finally, the study contributes with a deeper understanding for the studied organisation of the role of the architecture and it's position among other artefacts, and an increased focus on architectural issues in management meetings.

## 2. Background

### 2.1. Software in Automotive Systems

A modern high-end car is an embedded software system consisting of 30-70 different Electronic Control Units (ECUs), each with a microprocessor[1] executing in the order of 1 MByte compiled code. All software in the vehicle is deployed to these ECUs and they control the behaviour of virtually all electrical functions, from power windows to valve timing of the engine.

The in-vehicle software executing on the ECUs share a number of characteristics common to the automotive domain (see e.g. [2] and [3] for further elaboration):

- A large number of variants and configurations (often several brands sharing common platforms)
- Highly distributed real-time system
- Distributed development at vehicle manufacturers and suppliers
- Low product cost margins
- Stringent dependability requirements

This combination of characteristics together with a steady growth of features realised by electronics and software, makes the electrical system in a vehicle a highly complex software system, even though the amount of compiled code is smaller than in many other business domains.

### 2.2. The Architecture Business Cycle

According to Bass et al. the model of the architecture business cycle (ABC) is based on the assumption that "software architecture is the result of *technical*, *business* and *social* influences". The resulting architecture "in turn affects the technical, business and social environments" [5]. The key elements of the cycle are the forces influencing the architecture, the requirements that result from these forces, the architect and his experience, the architecture and the system (or systems in a product line architecture). The architecture business cycle also shows how these key elements influence each other, seen in Figure 1. In a later report the originators clarified the purpose; "...the architecture business cycle was envisioned as a means to depict the influences on a software architect and to show how architectures can eventually influence the very things that originally shaped them" [6].

1. A few safety-critical ECUs have two microprocessors for redundancy or internal monitoring.

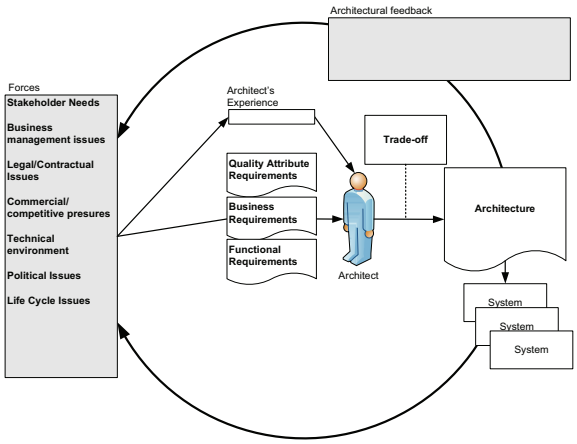


Figure 1. The architecture business cycle as defined by Kazman et al. [7]

The influences of the original cycle have been updated by the original authors in [6] and are subsequently called forces in [7]. This study is based on the latest of these updated architecture business cycles, since the seven categories of forces, seen in Figure 1, shaping the architecture was easier to relate to the interview responses.

The main idea of the cycle, that the architecture provides feedback in turn affecting one or more of the original influences or forces, have remained the same through all evolutions of the the architecture business cycle. The cycle is often used as a theoretical framework, e.g. in textbooks [5], [8], but it is hard to find empirical studies involving the actual stakeholders and not only as an observation of an architecture business cycle from a distance.

### 2.3. Automotive Software Development at Volvo Car Corporation

The Electronic and Electric Systems Engineering (EESE) unit is responsible for defining the software requirements for the in-vehicle software for all Volvo cars. The EESE unit is one of five units within Product Development at VCC. In most cases the coding of

the software is outsourced and written by suppliers, a practice very common in the automotive industry, but some parts of the code is also written in-house or auto-coded from models. The software development process at the EESE unit includes everything from collecting or defining vehicle level use cases down to allocating specific requirements to software deployed to certain Electronic Control Units (ECUs). The ECU software requirements details for example interfaces, state machines and/or control algorithms.

The EESE unit is also responsible for defining the architecture of the electrical system, both hardware and software. This includes enabling the quality attributes in the architecture necessary to achieve the business goals of the product development organisation.

## 2.4. Studied Subset of the Software Architecture

The architecture studied dates back to the first Volvo S80, launched in 1998. Obviously the software in present vehicles is not identical to that car, but many of the fundamental architectural strategies and views are still the same as when they were defined in 1995-98. We limited the scope of our study to a subset of the entire software architecture, described as three scenarios resulting from architecture decisions affecting virtually all in-vehicle software developed at the EESE unit, regardless if the code was written in-house or by suppliers. These scenarios are mostly unchanged in the ten years between when the first architectural decisions were made and when the study was conducted, even if updates have been necessitated by added functional content, legal requirements, etc.

In our study the selection of what scenarios to include was made according to four basic criteria:

- The scenario should be well-known by all developers and not need significant explanations in order to be studied.
- The scenario should not be excessively complicated to grasp.
- The scenario should be non-trivial, i.e. the captured cycles should be representative for the theory behind the architecture business cycle.
- The scenario should affect the development and design of *software* at the EESE unit (some architectural decisions mostly affects hardware, such as physical routing of the cable harness) .

The selection of scenarios was then made by an architect at Volvo Cars based on his "inside" perception of what subset of the architecture would be interesting to understand more in-depth. Of particular interest to the architect were scenarios resulting from eventual implicit decisions concerning the architecture. A second goal for the architect was to suggest a subset that were the result of architecture decisions that are likely to be affected by the future introduction of the AUTOSAR standardised software architecture [9]. Based on this the study focused on three architectural scenarios:

S1) Network topology of the in-vehicle multiplexed communication networks.
S2) Handling of software variants in production.
S3) Split of development responsibility among teams at the EESE unit.

**2.4.1. Network Topology.** The ECUs in a Volvo vehicle, and the software that runs on them, exchange information via a number of multiplexed network buses to enable functions that would not have been possible otherwise or overly costly if not being distributed. Almost all ECUs have a number of sensors and actuators connected to them depending on purpose and location, and these can be shared among distributed functions. The communication buses that exchange information between these ECUs are typically 2-4 CAN buses, 1 optical MOST bus and a number of LIN sub-buses.

Controller Area network (CAN), Local Interconnect Network (LIN) and Media Oriented Systems Transport (MOST) are all de-facto standards in tthe automotive industry, each having a different balance between cost versus bandwidth and dependability. CAN [10] and LIN [11] are twisted-pair and single copper wires respectively, while MOST [12] is an optical fibre for interconnecting multimedia components. All Volvo CAN and MOST connected ECUs are re-programmable, i.e. has flash memory and not ROM, which allows programming both in the manufacturing plant as well as at dealers and workshops after delivery to the end-user.

A multiplexed network topology was a major change for Volvo Cars when the decisions were made leading to the topology seen in Figure 2. At that time, 1998, only a few high-end cars had distributed system based on multiplex networks and the most common solution among vehicle manufacturers was to have point-to-point communication on dedicated wires between ECUs that needed to exchange information.

The layout of which ECUs are connected to which bus and what ECUs are acting as communication gateways between the buses is the network topology of a vehicle, of which the Volvo XC90 shown in Figure 2 is a representative example. Compared to other similar competitor vehicles designed at the same

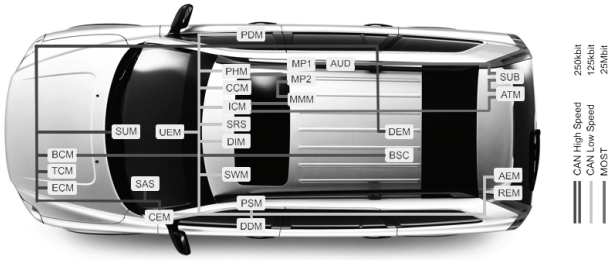


Figure 2. The network topology of a Volvo XC90. The ECUs connected to CAN and MOST and the main multiplexed networks are seen in their approximate physical location.

time the number CAN buses are usually lower in a Volvo vehicle, integrating all powertrain and chassis ECUs on one CAN bus and all comfort and body functions on another.

**2.4.2. Software Variant Handling.** The architecture prescribes two main strategies on how to support vehicles built to order with all the thousands of variants this means. The first solution is to handle software articles the same way as nuts and bolts in manufacturing, i.e. with separate article numbers for ECU software and hardware in the manufacturing system. Before this was introduced in 1998 all ECUs with both hardware and software were considered a single article, and the software were mostly stored in ROM, i.e. there was no possibility to update or change the software without changing the hardware.

The second solution for handling variants of the software on ECU level is using adaptation during start-up [13], i.e. when the ECU is powered it receives configuration parameters from a centralised stored parameter file over the vehicle networks. In some cases an ECU can in addition to the central parameters also have a local configuration file, which in this case is also having a separate article number in manufacturing.

**2.4.3. Development Responsibility.** The development responsibility at the EESE unit is typically assigned to teams according to end user functions, or grouping of functions. A example of development responsibility would be for locking functions, including central locking, double lock and child-blocked door. The teams are part of the line organisation at the EESE unit, i.e. the development responsibility does not vary a lot between different vehicle projects. These end user functions can span over several ECUs, and the development responsibility for the physical components (e.g. ECUs, sensors and actuators) are usually assigned to the same line organisation that are most strongly involved in defining the functional requirements on that component. Some support functions, not visible to the end-user, are handled in the same way, examples of these could be vehicle diagnostics, electrical energy management, etc.

Table 1. Interviewees at Volvo Car Corporation.

| | Role | Years at VCC | |
|---|---|---|---|
| 1 | Project manager | 5 | |
| 2 | Function designer | 2 | Consultant |
| 3 | System designer | 3 | Consultant |
| 4 | System designer | 6.5 | Consultant |
| 5 | Programmer | 10 | |
| 6 | Tester | 5 | Consultant |
| 7 | Tester | 2 | |
| 8 | Domain expert | 18 | |
| 9 | Function designer | 1 | Consultant |
| 10 | Domain expert | 7 | |
| 11 | Component Designer | 25 | |
| 12 | Line manager | 24 | |
| 13 | Quality Assurance Staff | 15 | |
| 14 | Project manager | 6 | |
| 15 | Architect | 1.5 | Consultant |
| 16 | Tester | 10 | |
| 17 | Architect | 7 | |
| 18 | Component designer | 2 | |
| 19 | Line manager | 7.5 | |
| 20 | Line manager | 11 | |

## 3. The Case Study

In the course of the study 20 persons were interviewed, which were selected as a purposive sample [14] in order to cover a comprehensive variety of roles and teams at EESE unit. We selected the interviewees to cover all roles of those that develop, deliver and maintain the system that is, all developer stakeholders according to IEEE Standard 1471. More specifically we aimed to have at least one interviewee each of "architects, designers, programmers, maintainers, testers, domain engineers, quality assurance staff, configuration management staff, suppliers and project managers or *developers*" [15]. The 20 interviewees who participated in the study are seen in Table 1.

The interviews were semi-structured with open-ended questions and started with some introductory questions to get some background about the respondent, like present role in the organisation, time employed at Volvo Cars, and a general idea of how the respondent viewed software at the EESE unit. Then the three architectural scenarios were briefly described to achieve a mutual agreement of what subset of the architecture was included in the scope

of study. The scenarios discussed in the interviews were chosen so they would allow the respondents to present their view of forces and feedback based on their understanding. The majority of each interview was based on a set of questions (Table 2) directed at exploring the respondents view of the architecture business cycle without necessitating an explanation of the theory behind the cycle. The questions were originally published in [16].

The interviews were performed by three students from the IT University of Göteborg, minimising bias in the interview situation by having an interviewer with a preconceived understanding of the current situation at Volvo Cars. This also eliminated any personal bias during the interview situation if the interviewer would have been previously known to the interviewees or would have worked together with them. The students performed and transcribed the interviews as part of their final project towards their bachelor's degree in Software Engineering and Management [16].

We used the theory of the architecture business cycle in our case study "as an initial guide to design and data collection" [17]. Our main goal with the study was neither to validate nor to redefine the existing theory behind the architecture business cycle. Our intent was to do an exploratory case study [18] in order to better understand how the EESE unit viewed the software architecture and how the architecture affected the work of the developers at this organisation.

### 3.1. Adaptation of the Interview Strategy

One dry-run interview was performed initially which was not included in the data. The sole purpose of this interview was to test the feasibility of the interview strategy and the questions and the data from this respondent was not included in the results. The interview result of this dry-run did not capture much useful knowledge for Volvo Cars. The forces identified were very general and basically only mimicked the general architecture business cycle as described by the original authors (Figure 1). These findings lead us to re-evaluate our interview design from general questions about software architecture to specific questions about the three significant automotive architecture scenarios in Section 2.4. The change in interview design was necessary since we judged this the most efficient way to keep a manageable scope of the study with the resources and time available, rather than elaborating with an interview design to cover the entire architecture. This change meant we gained a detailed insight in some parts of the architecture at the expense of coverage.

Table 2. In-depth questions to identify the architecture business cycle for the architecture scenarios [S1/S2/S3]. The questions were repeated for each of the three scenarios.

| # | *Question* | *Purpose* |
|---|---|---|
| 1 | Could you, briefly, tell us about the architecture for [SCENARIO] from your own perspective? | Understand how the respondent sees the realisation of the architecture. Build a foundation to base the next questions on. |
| 2 | What do you think influenced the architect to structure the architecture in that particular way? Persons, Roles? Techniques, Documents, Standards, Laws, Business Goals, Competition, Lifecycle Issues? | Gain knowledge of what the respondent thinks influences the architecture. Categorise in the following 7+2 force categories: Stakeholder Needs, Business Management Issues, Legal/Contractual Issues, Commercial/Competitive Pressures, Technical Environment, Political Issues and Lifecycle Issues, Developing Organisation and Legacy. |
| 3 | Do you normally consider "Non-functional requirements" or "Quality attributes"(QA) in your work? If so, which ones are the most important for you for [SCENARIO]? What kind of trade-offs among QA's does the architecture exhibit? | First of all, realise if the respondent uses/thinks of quality attributes at all when working, see if they are related to software engineering and not only hardware specific. See which QA's are the most important and how they conflict/relate. |
| 4 | Do you know who or which group of architects created the architecture for [SCENARIO]? How did their previous knowledge and experience affect the outcome? | Map the force category: "Architect's experience" |
| 5 | In which ways do you think the influences have been realised in the actual architecture for [SCENARIO]? | Find out about concepts, strategies, patterns, or the respondents lack of knowledge of them. |
| 6 | Have you ever reflected over how the architecture of [SCENARIO] or the system has influenced your role at VCC? Your group's role? The entire organisation in any way? | Map the force-feedback of the cycle, see how the influences that influenced the architecture are affected. |

### 3.2. Data Collection

The interviews were recorded, transcribed and coded by looking for statements related to the forces, architecture and feedbacks according to the general architecture business cycle. The data after transcribing and coding the 20 interviews are 60 diagrams each

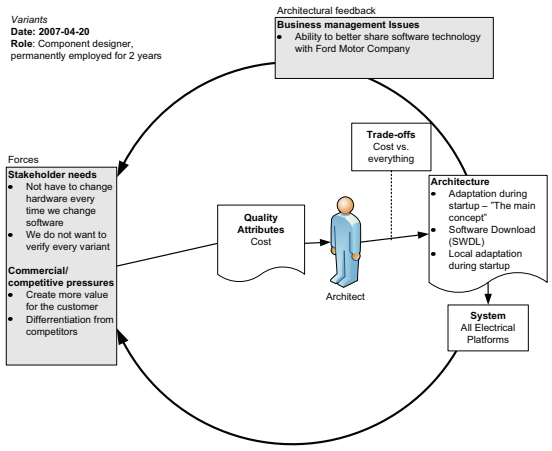


Figure 3. The architecture business cycle for software variant handling according to one of the interviewed component designers. Figure adapted from [16].

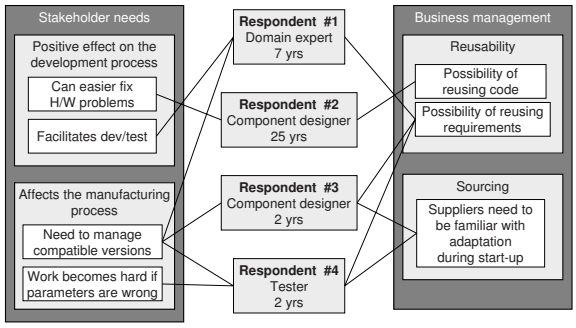


Figure 4. Example of how statements from some of the respondents are categorised and then sorted under the seven forces in the architecture business cycle.

showing an instance of the general architecture business cycle for a scenario. An example of such a diagram for the software variant handling scenario from one of the developers can be seen in seen in Figure 3. Each of the 20 respondents' opinions were captured with one diagram for each of the three scenarios studied.

The subsequent coding of the data to a comprehensive cycle for each of the three scenarios was done by grouping similar recorded quotes together into categories. The categories emerged when examining and comparing the data from the individual interviews. The categories found could then be assigned to a single force or feedback according to the general architecture business cycle. An excerpt of this categorisation is seen in Figure 4. This coding made it possible to build up a comprehensive cycle for the three architecture scenarios based on the merged information from all 20 respondents. The coding was done by three different persons, depending on the scenario being studied due to practical reasons, but the same person coded all responses regarding the same scenario to ensure consistency.

## 4. Results

Figure 3 shows an example of the data resulting from one of the interviews, in this case a component designer's view of the cycle for software variant handling. This cycle could be seen as a little "thin", he only mentioned one single explicit quality attribute requirement, cost, which in his experience influenced the architecture. This respondent's view of the cycle only had forces regarding stakeholder needs, and commercial/competitive pressures while the feedback in the cycle was thought to affect business management. While other respondents were not as extreme in their reflections, cost was clearly on of the main forces mentioned in all interviews. A broader view of the architecture business cycle from the viewpoint of a section manager with 24 years of experience at Volvo Cars can be seen in Figure 5. We note also in this cycle only quality attributes were listed as explicit requirements to the architecture and our conclusion is that the subset of the architecture studied is mostly driven by quality attributes rather than functional or business requirements.

The original architects responsible were not included in the study due to practical reasons, the architectural decisions were made ten years prior to the study, but a summary from one of the original architects can be found in [19]. Considering our focus was not to timeline what decisions had led to the current architecture, we do not perceive this omission to be problematic. Instead we were looking to capture the current view of the architecture business cycle for our three scenarios.

### 4.1. Disagreement on Development Responsibility

The more visible the results of architecture decisions were in the actual system and component design, the more agreement between the interviewed stakeholders there was in how the cycle looked like for that the scenario. For the abstract decisions leading to design

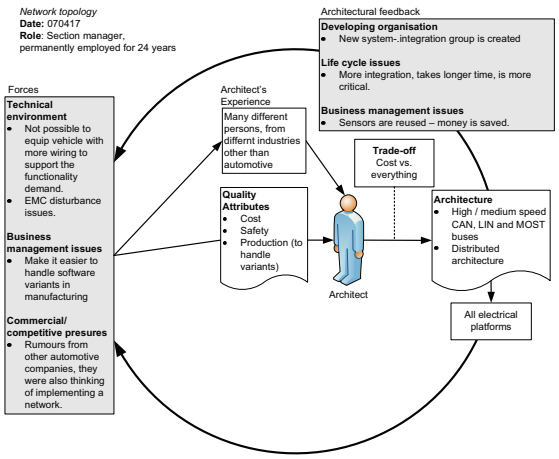


Figure 5. The architecture business cycle for network topology according to one of the interviewed line managers. Figure adapted from [16].

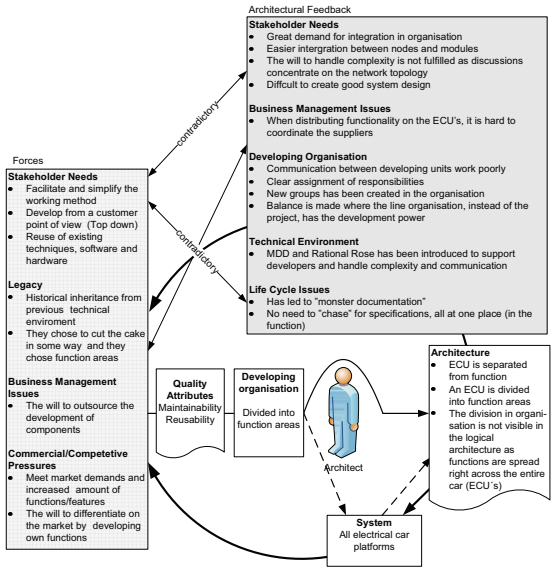


Figure 6. The architecture business cycle for the split of development responsibility based on the set of respondents which thought the architect was bypassed in the cycle. The figure highlights where some of the feedback in the cycle counteracts the original forces. Figure adapted from [16].

responsibility (where we could find no underlying conceptual principle) there was no overall agreement on the cycle. On one end there was a set of respondents saying the architect was in full control of the architectural decision, while on the other end respondents said the architect was largely bypassed and the feedback in the cycle counteracted the original forces. Figure 6 shows some details of the business cycle according to the latter respondents[2]. An interpretation of the latter respondents is that the development responsibility was perceived as an meta-level constraint to the architecture and not a decision possible for the architects to make, or at best an implicit architectural decision. The organisation of domain experts on brakes, climate, locking and other customer functions into separate groups together with respective hardware and software developers could be seen as a natural from an management viewpoint, while an architect could conceive it as a constraint.

## 4.2. Counteracting Feedbacks in the Captured Business Cycles

Our study revealed examples where an architecture decision actually counteracted the initial forces leading to the decision. For example, it was considered time consuming to request a new or changed global parameter used for the adaptation during start-up. Another example, from Figure 6, is where one of the needs was to "facilitate and simplify the working method" while the split of development responsibility has lead to "monster documentation". There was a clear "will to outsource the development of components" (i.e. ECUs) but in practice it was "hard to coordinate the suppliers" with the present deployment of functionality onto ECUs.

Since our study was a snapshot in time of the cycle we have not evaluated if these counteracting feedbacks have affected a change in the originating forces for the next generation architectures developed at EESE, as is explained by the theory in [5], but we are looking to explore this aspect in future work.

## 4.3. Role of the Architect at the EESE Unit

To capture the benefits for the organisation participating in the study we interviewed the Senior Manager for Electrical Architecture, a section within the EESE unit, who sponsored the original case study. This interview was made approximately 1 1/2 year after the original study took place and looked to capture the lasting effects of the initial study. The manager reflects on how the respondents at EESE

2. Some forces and feedbacks have been left out due to intellectual property rights of Volvo Cars, but those listed still illustrate the implicit effects on the business cycle from this scenario.

unit showed a discrepancy in perceptions of the role architecture and architect plays:

> "Where primarily management viewed architects to be the main contributors to early prerequisites of design work and limiting the number of design alternatives down-streams [later in the development work], the architects themselves found their role to be more reactive and victimized of decisions taken elsewhere."

This difference in views was only captured since from looking at such a variety of roles among the respondents, and was a major finding for the organisation. The manager continues and clearly finds major benefits for the EESE unit as a whole from the participating in the case study:

> "The study was an eye opener for the organisation that the architecture team may be under-utilized, which has since led to greater focus on strategic architectural issues in management team meetings. The study itself is no longer referred to, but acted as the initiator of the discussion."

When asked specifically about possible impacts the study have has on processes, artefacts or organisation he answers:

> "There has been a major reorganisation within EESE since the study was conducted and the study influenced that activity indirectly, resulting in co-organisation of architecture related disciplines into the Electrical Systems Design department."

This quote goes one step further by illustrating the impact the case study has as initiator for one of the goals in the re-organisation of the EESE unit—a re-organisation that resulted in strengthening the role and impact of architecture by defining a department dedicated to this. The initial study, and in particular the development responsibility scenario, played a role as one of several triggers for understanding and change.

## 4.4. Reflections on the Architecture Business Cycle

In our study we aimed to capture an instance of the architecture business cycle for the comprehensive software architecture in a modern vehicle, but our dry-run interview did not reveal much information useful to Volvo Cars beyond what was already described in the general architecture business cycle [6].

In the study we therefore focused on a subset of the architecture and as a result captured a separate cycle for each of three architecture scenarios. Our conclusion is that each architecture scenario in itself had a set of forces and feedback according to the theory behind the general architecture business cycle.

**4.4.1. Difficulties in Capturing the Cycle.** The interview responses for one of the architectural scenarios investigated varied too much to be captured in a single diagram of a cycle. We are at this time hesitant to conclude if this means the theory of the architecture business cycle needs to be expanded or if our methodology to capture the cycle must be improved to resolve ambiguities. We can think of three possible explanations why we could not capture a single architecture business cycle for the last architecture scenario:

1) The captured architecture business cycle is very dependent on the respondent, or set of respondents.
2) It is an inherent fact that the results from some decisions are not easily captured in single diagram of an architecture business cycle.
3) The particular decisions leading to the scenario for which we could not capture capture a coherent cycle, did not involve the architect (or the architecture), at least according to some respondents.

It was not possible for us to determine the exact cause of why it was so difficult to capture a coherent cycle for the last scenario of development responsibility, as this would have required an extended study and additional run of interviews.

**4.4.2. Additional Forces.** In the course of the interviews some of the respondents mentioned forces influencing the architect and the architecture not readily categorised according to the general forces of the architecture business cycle in [7]. The students who performed the interviews suggested two additional categories of forces: Developing Organisation and Legacy, besides the seven forces proposed in [7]. We agree with this analysis and note that the development organisation category can be seen as a result of Conway's law [20]. The development responsibility is likely shaped by the existing organisation, where different teams have their areas of expertise, and not only by the other seven forces. Legacy is the result of the strong demand to re-use existing and proven system solutions when developing new vehicles, i.e. a domain specific category. If a similar study of using the architecture business cycle would be performed in other particular domains it is reasonable to expect that

more categories would need to be added besides the seven original categories of forces.

### 4.5. Generalisation of the Case

We believe the presented method of capturing an architecture business cycle by scenarios defining a subset of the software architecture is possible to generalise beyond our case in the automotive domain. We found the mapping of the feedback in the cycle valuable in understanding the role of the architecture in the development process and we expect this understanding would be equally valuable for other architectures studied.

The nature of the architecture business cycle for a specific architecture make the cycles we captured unlikely to generalise to other architectures or organisations. However, some of the forces identified in our case study may be similar also for other automotive manufacturers with comparable products operating in the same markets, most likely forces in the following categories:

- *Legal/contractual issues*
  The automotive domain is very regulated and the OEM-supplier relationship does not vary much between European companies. Also the OEM-end customer relationship is very similar between car brands on the same markets.
- *Commercial/competitive pressures*
  Premium car manufacturers are competing for the same customers on the same global market.
- *Technical environment*
  Some automotive technology are de-facto standards, such as CAN [10], and other is defined by legal requirements (e.g. diagnostics).

Similarly, we would expect that if identical architectural decisions have been made by other architects in other automotive companies, they would assert the same influencing feedback on the original forces according to the architecture business cycle.

The last generalisation we see is based on our experiences with working with architecture business cycle "as an initial guide to design and data collection" [17]. We think the architecture business cycle worked quite well in this respect and believe it would be useful as a theoretical framework in the design of other case studies and in the collection and organisation of the data, even if some hands-on adaptations are needed.

## 5. Conclusions

There were several, both theoretical and practical, benefits resulting from our study:

From a research perspective we found that the architecture business cycle worked very well in defining the context for the interviews. The cycle also worked well as a guiding model to understand the role of the architecture in the software development process. So both the theoretical framework and the interview methodology used in this case should be possible to generalise for studies at other organisations.

However, when trying to capture the a comprehensive architecture business cycle in a single interview situation the answers we got were too general to be of use to the organisation studied. We therefore tried to capture a separate cycle for a subset of the architecture, defined by three major architecture scenarios. Our conclusion is that these scenarios in themselves have a set of forces and feedback which could be described according to the architecture business cycle.

The Electric and Electronic Systems Engineering unit at Volvo Cars gained several benefits from participating in this study: The architects gained a better understanding of how the identified forces affected the decision process and the architecture resulting from these decisions. The management at the EESE unit discovered the architecture team may not be fully utilised and as a result the organisation increased the focus on strategic architecture issues in management meetings. Finally findings in the study indirectly influenced a subsequent re-organisation at the EESE unit. We believe that studies at other organisation could also gain similar benefits.

Finally we find that the architecture business cycle is useful, not only in theory but also in practice, in understanding the relationship between the will of stakeholders, the architects, the decisions shaping that architecture and how these decisions in turn affect the stakeholders, as shown in this case study.

## 6. Future Work

The obvious follow up to this study would be to see if the "more focus on strategic architecture issues in management team meetings" and "the co-organisation of architecture related disciplines into the Electrical Systems Design department" had the intended results of better utilising the architecture team.

One thought is to use the architecture business cycle for predicting change, i.e. how would the feedback cycle look like and how would the originating forces be affected if a specific software architecture would be introduced. A current example would be AUTOSAR in the automotive industry. A first attempt to do this can be seen in [16], but to support this a study would need

to investigate more scenarios of the architecture than covered there.

## Acknowledgements

VINNOVA funded this work as a part of project 2009-00091. We are grateful for all the time the participants from Volvo Car Corporation have willingly contributed. The authors would like to thank Prof. Thomas Arts and the three anonymous reviewers for their useful comments.

The authors would also like to extend huge thanks to David Smallbone Tizard, John Wallmark and Thomas Warrby, students at the IT University 2004-2007, for their invaluable work in conducting and transcribing the interviews and providing useful ideas as part of their bachelor thesis project. We are not surprised their work was nominated as best bachelor thesis in Sweden 2007.

## References

[1] G. Reichart and M. Haneberg, "Key drivers for a future system architecture in vehicles," in *Proc. Convergence 2004*. Detroit, MI, USA: Society of Automotive Engineers, oct 2004. [Online]. Available: http://www.sae.org/technical/papers/2004-21-0025

[2] A. Pretschner, M. Broy, I. H. Kruger, and T. Stauner, "Software engineering for automotive systems: A roadmap," in *2007 Future of Software Engineering*. IEEE Computer Society, 2007, pp. 55–71. [Online]. Available: http://portal.acm.org/citation.cfm?id=1253532.1254710

[3] M. Broy, "Challenges in automotive software engineering," in *Proceedings of the 28th International Conference on Software Engineering*. Shanghai, China: ACM, 2006, pp. 33–42. [Online]. Available: http://portal.acm.org/citation.cfm?id=1134285.1134292

[4] J. A. McDermid, "Complexity: Concept, causes and control," in *Proceedings. Sixth IEEE International Conference on Engineering of Complex Computer Systems, ICECCS*, 2000, p. 29. [Online]. Available: ftp://ftp.cs.york.ac.uk/pub/hise/Complexity-Concept,Causes&Control.pdf

[5] L. Bass, P. Clements, and R. Kazman, *Software Architecture in Practice*, 2nd ed., ser. SEI Series in Software Engineering. Addison-Wesley, 2003.

[6] R. Kazman and L. Bass, "Categorizing business goals for software architectures," Carnegie Mellon Software Engineering Institute, Tech. Rep. CMU/SEI-2005-TR-021, dec 2005. [Online]. Available: http://www.sei.cmu.edu/pub/documents/05.reports/pdf/05tr021.pdf

[7] R. Kazman, L. Bass, P. Clements, and R. Nord, "The architecture business cycle revisited: A business goals taxonomy to support architecture design and analysis," http://www.sei.cmu.edu/news-at-sei/columns/the_architect/2005/2/architect-2005-2.htm, Carnegie Mellon Software Engineering Institute, 2005.

[8] T. Noergaard, *Embedded Systems Architecture: A Comprehensive Guide for Engineers and Programmers*. Newnes, feb 2005.

[9] AUTOSAR, *AUTomotive Open System ARchitecture (AUTOSAR)*, http://www.autosar.org, AUTOSAR development partnership Std., 2009.

[10] ISO - International Organization for Standardization, *Controller Area Network (CAN)*, International Organization for Standardization Std. 11 898, 2003. [Online]. Available: http://www.iso.org/

[11] LIN Consortium, *Local Interconnect Network (LIN)*, http://www.lin-subbus.org/, LIN Consortium Std., 2008.

[12] MOST Cooperation, *Media Oriented Systems Transport (MOST)*, http://www.mostcooperation.com/, MOST Cooperation Std., 2008.

[13] F. Bachmann and L. Bass, "Managing variability in software architectures," *SIGSOFT Software Engineering Notes*, vol. 26, no. 3, pp. 126–132, 2001. [Online]. Available: http://portal.acm.org/citation.cfm?id=379377.375274

[14] B. L. Berg, *Qualitative Research Methods for the Social Sciences*, 6th ed. Allyn & Bacon, mar 2006.

[15] *IEEE Recommended Practice for Architectural Description of Software-Intensive Systems*, IEEE Std. 1471, 2000. [Online]. Available: http://standards.ieee.org/reading/ieee/std_public/description/se/1471-2000_desc.html

[16] D. Smallbone Tizard, J. Wallmark, and T. Warrby, "Architecture and change: A case study using the architecture business cycle for assessing an organisation facing a major architectural change," Bachelor thesis, IT University, Göteborg, Sweden, jun 2007.

[17] G. Walsham, "Interpretive case studies in IS research: nature and method," *European Journal of Information Systems*, vol. 4, pp. 74–81, 1995. [Online]. Available: http://www.palgrave-journals.com/ejis/journal/v4/n2/abs/ejis19959a.html

[18] R. K. Yin, *Case Study Research: Design and Methods*, 3rd ed. Sage Publications, 2003.

[19] K. Melin, "Volvo S80: Electrical system of the future," *Volvo Technology Report*, vol. 1, pp. 3–7, 1998. [Online]. Available: http://www.artes.uu.se/mobility/industri/volvo04/elsystem.pdf

[20] M. E. Conway, "How do committees invent?" *Datamation*, apr 1968. [Online]. Available: http://www.melconway.com/research/committees.html