ABSTRACT
The requirements on increasing functionality, quality, and customisation, while reducing cost has lead to the introduction of an architecture centred development process for electronic systems at Volvo Cars. This process enables better control of system integration and achieving non-functional requirements, such as reusability, understandability, etc. The result of the process is a reference architecture that includes strategies for implementing the balanced requirements, architectural views that provide means for reasoning about all the concerns of all stakeholders, and a top-level design of the architecturally significant parts. The reference architecture guides the design of several projects, and thus, cost is optimised accordingly. The main contribution of this paper is that we present experiences from introducing the architecture centred process. The main conclusions are that disseminating and maintaining the reference architecture actually require more resources than developing it. Furthermore, experience shows it is difficult to create an architecture that enables a lot of different variants that is also strategically useable in the long term.

Categories and Subject Descriptors
D.2.11 [Software Engineering]: Software Architecture – Domain specific architectures.

General Terms

Keywords
Automotive electronics and software, reference architecture, product line architecture, non-functional requirements.

1. INTRODUCTION
Over the last decades there has been a drastic increase in the functionality implemented in electronic systems in vehicles. This affects both hardware and software, and increases the importance of system integration activities, requiring more staff as the complexity increases. The systems are highly distributed, which also adds to the complexity and development effort.

In all development involving more than one group, there is always the problem of how to control the conceptual integrity of the design and implementation among the groups, to ensure that a homogeneous design is obtained [3]. For distributed embedded systems, the implications of an uncontrolled architecture may be:

- Unnecessary design dependencies resulting in domino effects when components are redesigned.
- Redundant computations resulting in inefficient use of the computational resources.
- Complicated solutions decreasing the possibility of robust designs, making fault tracing and repairing difficult, and increasing the cost.

The use of an architecture centred, top down development process for embedded system was first suggested in the beginning of the 1970s, e.g. in [3]. However, reference architectures, i.e., a common architecture guiding several projects, were first successfully applied in the beginning of the 1990s, e.g. [4]. Not only has architectural design evolved over time, but the description of an architecture has also matured. A standard for documenting software-intensive systems is described in [7]. It recommends defining the system's stakeholders, their concerns, and architectural views to address the concerns [8].

Architectural efforts are however not easily transferred to the automotive industry since the operating environment is quite extreme. Some key characteristics are:

- Safety critical consumer products
- Product cost is an important factor
- Mass production (Volvo Cars: 460 000 cars 2004)
- Long operational life and unreliable maintenance
- Complex supplier structure (75% of material value)
- Sold world wide: different market requirements
- Almost unlimited number of variants

Existing development processes must be adapted to these conditions. Figure 1 illustrates how this adaptation was performed in practice, but also how the process has evolved over the years. First, best-practice processes were adapted to the requirements of the Original Equipment Manufacturer's (OEM) requirements.

---

1 We use the term Reference Architecture as a synonym to Product Line Architecture, in order to avoid confusion with the production line in manufacturing.
Thereafter, the resulting architecture development process was introduced, but in reality the real process will always deviate due to traditions, inconsistencies, organisation, limitations, etc. These factors can be regarded as a kind of noise. Over time, experience is gained on what works well or not. This experience is used to refine the development process, also considering changed prerequisites, requirements, and the advancements in the state of the art.

Figure 1. Overview of the development process evolution.

1.1 Scope of the Paper

In this paper, we describe the evolution of the architecture development process at Volvo Cars. Specifically, we explain the background to why we introduced an architecture-centred development process and how we apply this in practice.

The paper is structured as follows: Section 2 gives a brief historical overview of electrical architectures at Volvo Cars. Section 3 describes the motivations for moving to an architecture-centred development process. In Section 4, we describe the work we are doing today to develop the next electrical architecture, and the experiences we have gained so far during this work. Finally, in Section 5, we present some important conclusion.

The main contributions of this paper are that we present experiences of using architectures to guide and control the design of automotive electronic systems, and how the concept of reference architectures is now used to improve the development process. This paper supports the conclusions made in [10], but goes further by reasoning about the necessity of an architecture-centred development process, and also describing how Volvo Cars applies such a process.

2. THE ROLE OF ARCHITECTURE AT VOLVO CARS

With the start of development of the present Volvo S80, V70 and S60 (start of production 1998) there was a significant change in the electrical architecture compared to previous vehicles at Volvo Cars. The electrical system became a network of ECUs connected on two multiplexed buses rather than a set of stand-alone systems that happened to share space in the same EDS harness [9].

In order to facilitate this change, two architectural views were established that guided subsequent component design: a network topology view and an ECU software view. The first view defined what ECUs the electrical system would consist of (and is an abstraction of the electrical schematics). The second view defined the basic software components an ECU should implement and their internal relationships: Real-time operating system, CAN communication driver, software bootloader and a diagnostic kernel. Together with these views a number of associated strategies were identified, e.g. that all ECUs should use flash memory instead of ROM, and use the same protocol for ECU programming (This strategy has since then also guided how Volvo Cars handle software upgrades after manufacturing, including the aftermarket infrastructure and tool chain). The views were captured in a single architecture description: System Requirements Description - Complete Electrical System, with the last, and also the largest chapter containing the specific requirements the architecture placed on the major components in the electrical system. This original architecture for S80/V70/S60 did not have any requirements regarding feature integration.

In the following projects, more detailed requirements were gradually defined, especially those guiding the implementation of basic software modules, since these were developed by, and purchased from 3rd party suppliers and integrated with application software in ECUs. The number of variants of network topologies also increased. Volvo Cars therefore started to describe the functional content of the electrical system with the help of modelling tools: StateMate for validating the right behaviour and UML to define the functional design of features [11]. The latter could be considered an architectural view, but was only to a small extent used to identify and validate strategies for feature interaction.

With the introduction of a MOST-based infotainment system in Volvo XC90 we developed a reference architecture to handle certain aspects which were troublesome: e.g. design responsibility, prioritisation of resources in the system and some non-functional characteristics like “IT and other functions in the car must be possible to upgrade even after the car has left the factory”, “part of the electronics can also be built in modules which can be replaced”, and “...the interface to the driver is becoming an important competitive advantage”. The architecture focused on a logical view, showing interaction between components in the infotainment system to realise customer features, which was a new aspect for an architecture developed by a small core team. Today, we extend the role of the reference architecture to guide all domains (infotainment, chassis, powertrain, body, and passive safety).

In the next section we will give the motivations to why this architecture-centric development process is introduced.

3. MOTIVATION FOR ARCHITECTURE CENTRED DEVELOPMENT

As described in the previous section, the development process at Volvo Cars has evolved during the years to an architecture-centred development process using reference architectures. The evolution is obviously guided by the wish to improve functionality, customisation, and quality while reducing cost. In this section, we point out in more detail how an architecture-centred development process supports this balancing. Volvo Cars is not the only manufacturer to reach these conclusions; a similar position is brought forward by BMW [10].

3.1 Handling of Distributed Functionality

The traditional vehicle design has enjoyed the two axioms that each component mainly or exclusively serves one system, and that the components of a system are stable over the years. This is still true for some systems in the vehicle, such as the wheel system.
which consists of a rim and a tyre. However, some systems which we perceive as being stable are slowly drifting into new domains; the electrical power supply system not only consists of a battery, an alternator, a regulator and some heavy wiring; it will now also include things like battery monitors, heavy load schedulers, alternator torque strategies etc. These will either be located in dedicated ECUs, or just as often as algorithms hosted by existing ECUs for body control or engine management. The point is that this allocation can and will differ between car models.

It becomes even worse when we look at newer or less established systems. Consider a Locking or Alarm systems. Ten years ago, every vehicle with central locking would have had a dedicated lock control module. At most, this module would also handle alarm. This is not financially feasible any more, since take rates are close to 100% and these systems have to be as cost efficient as any other. Therefore the functionality will naturally evolve to become distributed across other components and ECUs. However, this causes three major problems:

- Fault tracing is complicated and differs between cars.
- Relations between components may prohibit re-use, since component A requires components B and C to be present, or it will not work.
- The complexity of system integration is increased.

An architecture centred development process, where each partitioning or distribution decision is consciously made, always considering the whole vehicle as well as long-term strategies, will help minimising these three problems. A reference architecture provides the guidance for harnessing solutions and driving them to commonality on all levels.

### 3.2 Balancing of Attributes

Another reason to introduce reference architecture based design into the process is to prove how the selected solution delivers to the high level attributes of the vehicle. OEMs will initiate the development of a new vehicle by setting some kind of high level attributes. These will include Handling, Cost, Reliability etc., and typically number between 20 and 50 attributes. Each will be given a priority for the vehicle at hand, and they will be balanced to each other. This set of attributes will then govern all design activities.

This seems straightforward enough, and it sometimes is too. One can well imagine that changing the properties of a strut may improve Handling. However, it becomes much more difficult when designing new, complex, distributed or heavily changed components. Without an architecture centred development process, it is almost impossible to prove that "sensor A has to have better resolution because of attribute C". A development process that starts with the set of balanced attributes, makes design decisions whilst step-by-step moving down into deeper detail, but always tracing each decision back to the higher levels, and always considering impact across the whole vehicle, will at the end allow the designers to state with confidence that "this piece of software code will have to follow MISRA guidelines because of attributes E and F". With a reference architecture these design decisions and their traces can be reused over and over.

### 3.3 Using International Standards

New initiatives like AUTOSAR [1], [6] are providing common standards for low-level software and hardware. These will be applied by most OEMs and automotive suppliers. But the strategy of having standardised basic software components from 3rd party suppliers is not new to Volvo Cars. They have been used across ECUs since 1998 [5], [9]. The main difference between AUTOSAR compared to Volvo Cars in 1998 is that most of these basic software components were unique to us. The new standard of ECU software is a change in design, but not in what we as OEM require from our suppliers.

The standardisation of basic software will in itself constitute a part of a complete reference architecture. To make full use of this, all higher levels of the vehicle definition process need reference strategies. Furthermore, since the technical concepts now will be common among all car brands, Volvo Cars, to be competitive, must design systems efficiently, e.g., determine when to reuse and when to develop from scratch.

### 3.4 Business Environment

As long as Volvo Cars only worked with one platform simultaneously (e.g. S80/V70/S60) it was manageable to work without a reference architecture guiding the product line engineering over time. Today, we are not only developing electrical systems to suite Volvo Cars, but all of the Premier Automotive Group2. When we work with three or more platforms simultaneously across brands within PAG it is necessary to guide the development with a reference architecture, otherwise the effort of maintaining so many separate designs is just too great. Another conclusion is that the project of developing an electrical architecture should follow a long-term strategy and not just meet the needs of a specific car project.

### 3.5 Improving Understandability

The more complex the electrical system becomes, the more important understandability becomes as a non-functional requirement when shaping the architecture; both for choosing the views to describe the architecture, but also for shaping the design in itself. The architects at Volvo have had difficulties in asserting how well the original strategies and design guidelines of the architecture were adhered to while the vehicle project progressed. This is caused by a combination of increased feature content, more complex design, and inadequate methods of describing the top-level design.

The worst scenario occurs when the architecture description is used as a document to capture what the design ended up looking like, rather than guiding and controlling the development of the system. This often happens when the architecture is developed late in the project, concurrent with component design. Even though this improves understanding of the complete electrical system compared to having no description at all, it usually captures an unnecessarily complex solution.

### 3.6 Formalising the Process

The process of designing the architecture has been rather informal. The drawback of having an informal process has been alleviated by two factors; the architecture is usually defined by a relatively small team, and the architecture design process is as much art as science. However, due to the decreased lead times, the increasing number of variants required variant to be handled.

2 The Premier Automotive Group (PAG) of Ford Motor Company consists of Volvo Cars, Jaguar, Land Rover and Aston Martin
increased interest in cross brand reuse within Ford Motor Company, and increased system complexity, there is a need for formalising the process.

### 3.7 Clarifying Requirements

More difficult than capturing a requirement is to determine whom it concerns. For a sub-system/function/component developer it is not vital to know how the architecture was developed, e.g. if different strategies and views were developed by different persons or whether they are new or reused. The important factor is that the relevant input requirements to the component are presented in as a clear manner as possible. Equally important is that the sub-system/function/component requirements are complete and stable enough for carrying the design to the next level. However, it is also necessary to have a clear overview of the total architecture, i.e., to understand the design (this is related to Sec. 3.5). This also enables evaluating the underlying principles that unify the design of the complete electrical system at all levels. The architecture should do similar things in similar ways [2].

### 3.8 Simplifying Reuse

The first attempts at reusing components, and therefore decreasing development efforts and manufacture larger series was an ad hoc approach, seen in Figure 2. If the specific design of a component or a sub-system seemed to fit in the new vehicle it was attempted to carry over that solution. This indirectly also defined the architecture of the new project, manifesting inconsistent architecture principles and strategies rather than making them the result of an active design decision based on given attributes and prerequisites.

![Figure 2. The first architecture process: Ad hoc reuse.](image)

### 4. ARCHITECTURE CENTRED DEVELOPMENT

In Section 3, we discussed the reasons to why we introduced an architecture centred development process for the electrical system at Volvo Cars. An important decision was to use the same basic architecture for electrical systems in more than one vehicle. This is business decision, it will impact both work processes and organisation. If the management on all levels are not backing up such a change, or if it is perceived as something that will only affect the design of the electrical system, working with a reference architecture will be very difficult to implement successfully.

The reference architecture should alleviate the effort of developing and maintaining several detached electrical systems in parallel. It highlights that the engineering of the electrical systems should use a product line approach, where important assets are developed already from the beginning to fit several platforms rather than a specific project, seen in Figure 3.

![Figure 3. The proposed electrical and electronic systems engineering process](image)

In this section, we present how we have worked with the reference architecture for the infotainment domain and how we now work with the reference architecture for the complete electrical system. The process can be divided into the following steps:

1. Analysing the design prerequisites
2. Designing the reference architecture
3. Verifying the reference architecture
4. Disseminating the reference architecture
5. Maintaining the reference architecture

The relationship between the steps and the resulting artefacts are seen in Figure 3. The steps are not discrete events, but are closely tied together. For instance, when gathering the design prerequisites from different stakeholders, this is also the first step in disseminating the architecture.

### 4.1 Analysing the Design Prerequisites

The gathering of design prerequisites is managed by the architectural team and starts by identifying the different stakeholders of the electrical system, and with capturing their specific prerequisites. This task has become more difficult as the stakeholders nowadays are not only located only at Volvo Cars, but over all sites of Ford in Europe.

First, we need to understand all characteristics that the design must follow, i.e. legal requirements. We must also understand the functional requirements on the system. At early phases these may only consist of a well-defined feature content (from product planning), together with use cases that can be reused from previous projects, describing well-known functionality. Equally important are the non-functional requirements on the system (e.g. brand identity attributes such as safety) and design constraints (which include management decisions on commonality and reuse). Further constraints include the use of standards (e.g., AUTOSAR and ISO 14229) and supplier implications (each supplier has its own architecture and product line).

Non-functional requirements within the automotive industry fall into two categories. The first are the vehicle attributes, which usually are formally defined, e.g. by program attribute teams (see Section 3.2). These are requirements that affect the customers'
perception of the vehicle. This set of attributes differs slightly between brands, which in itself is a difficulty when developing a reference architecture to be used cross-brand within FMC. The set of attributes used at Volvo Cars are to a high degree driven by our core values: Safety, Quality and Environment.

The second type of non-functional requirements is based on the business environment of the development organization (which is not necessarily the same as the business environment for the entire company or industry). These requirements are often less formally described, but nevertheless important for the development organisation in shaping an architecture. Typical examples are:

- **Reuse** - Requirements to use already existing technical solutions or components.
- **Variability** - Enable customized vehicles based on the same electrical system.
- **Testability** - The ability to test the system regarding the fulfilment of system requirements.
- **Understandability** - The ability to understand and reason about the top level design of and technologies used in the system.

In real vehicle projects a complete set of design prerequisites and input requirements is never available when starting the development of a new electrical system. Often those requirements that are most significant in shaping the architecture (including the top-level design of the complete electrical system) are not fully defined until well into the project, and after several iterations.

When the prerequisites of the system have been listed, the process of balancing them can be started. The balancing both includes prioritising between them (usually based on brand attributes), and identifying if there are contradictory prerequisites. An analysis of new and altered prerequisites is also done, to determine if the present reference architecture is applicable for the project at all, or if the prerequisites are too different.

### 4.2 Designing the Reference Architecture

From the balanced prerequisites, we must work out good design strategies and a top level design that facilitates that the balancing is preserved in the design. To be able to perform the top level design, and simplify verification of the architecture, we also define different views for reasoning about various concerns.

#### 4.2.1 Strategies

The strategies describe important design rules which have to be followed (deviations must be formally approved). Each strategy also has a rationale which explains why the strategy should be followed. The rationale traces back to the non-functional requirements the system must fulfil, but can also be motivated by having a uniform design (there can exist equally good alternative solutions, but this particular one was chosen).

One of the main lessons learnt during the last 3 years is the importance of presenting the rationale for each and every strategy in the same architecture description (including previously implicit strategies). This makes it much easier to determine if a specific strategy from the reference architecture is applicable for a specific project. In some cases we had to reverse engineer existing architectures to determine a rationale for some strategies. Strategies can be either design related; e.g. decomposition strategy or power management strategy, or process related; e.g. modelling strategy or safety process strategy.

#### 4.2.2 Views

The design views should provide means for reasoning of all the concerns for all stakeholders. Thus, they can be seen as the basic tool for determining the effects of specific designs on different requirements (e.g., safety, availability, etc.) and for that reason it is important to determine and maintain the consistency between the different views. Examples of views used in the reference architecture are the network topology view and the mechanical view. Three to seven views are a reasonable number to maintain and keep consistent.

#### 4.2.3 Top-Level Design

The architectural design is the first attempt at a top-level design. The design should be performed in the suggested design views and be in accordance with the defined strategies. It should identify which architecturally significant components are required and the necessary dependencies between these components (and in some cases forbidden dependencies).

The components can at this stage already be explicitly defined as software or hardware, but also be more abstract (i.e. logical or functional components, which later may be realized by a combination of software and hardware). Similarly to defining strategies, it is often possible to perform this design in many different equally good ways, but one must be chosen and followed. Included in the design work is also allocation of requirements on subsystems and components.

In our work, we have in many cases performed reverse engineering to capture the design of previous solutions, followed by a redesign according to the architectural strategies.

#### 4.2.4 Deriving the Function/System/Component Requirements

The strategies, views and architectural design constitute the rationale for subsequent design. Thus, the final step of the process is to specify detailed requirements for the components that constitute the system. The architecture typically defines:

- More detailed design requirements on each component.
- The difference between information (which enhances understanding) and requirements. It is vital that these are clearly marked and kept apart.
- Interfaces between logical, physical and electrical components.

An important principle is that the documentation should be kept small: few specifications are preferable to many.

### 4.3 Verifying the Reference Architecture

When we have designed a reference architecture, the architecture is verified through reviewing (to see that it follows the prerequisites balancing), scenarios (e.g., adding some new functions and verify that the architecture supports them), and box cars (used to test basic software and some basic architecture strategies).

### 4.4 Disseminating the Reference architecture

Our experience is that the effort of developing an architecture is much smaller than the effort of disseminating it in the organization. For the previously mentioned Infotainment Architecture it took a dedicated team of 4 persons 7 weeks to identify the prerequisites and develop 90 % of the basic
architectural strategies. The dissemination after that to concerned developers, and the development of the final 10% of the architecture, took about 18 months. This second stage was done by 2 to 3 of the original architects together with 6 function owners and 7 component developers at Volvo Cars.

Various stakeholders need to achieve sufficient understanding to successfully implement the architecture, and developers design their component according to the architecture which in turn requires them to accepting the benefit of this architecture. In our experience the best method of dissemination is to have the engineers who developed the architecture work in close cooperation with component developers. Furthermore, having component developers review the relevant parts of reference architecture that concerns their components also simplifies the dissemination.

The present method of handling development across brands at geographically separate locations is to establish a trio (if 3 brands are involved in a platform) with one representative from each brand. One of the brands is the dedicated leader with responsibility to develop a system solution that suits all concerned, or identify if and why, this is not possible. A similar trio exist for the electrical architecture of the platform with the same responsibility. One implication of this method is that it assumes that the product structure is identical at the involved brands, and the organisation at each brand follows the product structure. In the latest platform project of Ford in Europe it took more than a year to identify the discrepancies in product structure since the structure mostly was informally defined, and not explicitly seen in any architecture. It was the evolution of the brand organisations that had defined the product structure and not the other way around.

4.5 Maintaining the Reference Architecture
Developing a reference architecture is only one part of a successful architecture centred development process. An equally important part is how to maintain it during its life cycle. The importance of this can be better understood when one is reminded that the cost optimisation of the architecture is not performed for each project, but for a number of projects. Thus, if the architecture is not maintained, the cost of using a reference architecture is generally higher than project based development processes.

The main key to a successful reference architecture is to find the right level of detail (which decisions are generic and which should be project specific), and to continuously evolve of the reference architecture. Finding the right detail level can only be gained by experience and through tight communication. Continuous architecture work must be supported by a dedicated team. To further support that the conceptual integrity is maintained as well as possible, an architect should follow each new project to its end. This architect is responsible for verifying that the reference architecture is respected, documenting all deviations, and for feedback of information on changes and extensions to the architects responsible for the reference architecture.

5. CONCLUSIONS & CONTINUED WORK
The first important point of this paper is the necessity of a centralized development process, as described in Section 3. More and more functions are integrated and the process to do this must be centrally controlled. There is also a need to show that high level attributes are satisfied.

The second important point is the change from project based architectures to platform architectures. Earlier we had architectures for each specific vehicle project, now a core architecture team is responsible for an entire cross-brand platform. The architecture should support the advantages of this, e.g. increase the possibility for strategic reuse (we plan to write future papers describing the technical content of the architectures).

The architecture for the infotainment domain has so far been successful, and has guided the design of all MOST-based system in the Premier Automotive Group. The success of this architecture depends heavily on identifying a relevant set of non-functional architecture prerequisites (keeping these stable over time), and that the design was well-structured and based on established software architecture patterns. It is interesting to note that in this project the dissemination of the architecture to the concerned stakeholders relied more on close cooperation between architects and other developers than on written formal requirements. Thus, we believe that the success of using a reference architecture is as much related to the dissemination as the actually design.

The next step is to adopt the reference architecture according to AUTOSAR and to develop and improve the verification methods for assuring the variant handling and long-term validity.

6. REFERENCES