**TECHNISCHE UNIVERSITÄT MÜNCHEN**

Institut für Informatik

# The Impact of Architecture on Quality of Electronic Control Systems

The Cost of Architectural Complexity

Witold Drytkiewicz

Vollständiger Abdruck der von der Fakultät für Informatik der Technischen Universität München zur Erlangung des akademischen Grades

Doktor der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Die Dissertation wurde am 03.07.2012 bei der Technischen Universität München eingereicht und durch die Fakultät für Informatik am 11.12.2012 angenommen.

## Abstract

Architecture in automotive electrics-electronics (E/E) systems is influenced by a wide range of concerns and trade-offs: functional and non-functional requirements, historical decisions, organizational responsibilities as well as economic aspects of production and development. It has been recognized as early as in 1976 (Parnas, 1976) that architecture embodies the earliest decisions regarding these trade-offs. Architectural decisions are the hardest to change (Fowler, 2002) and have the highest impact in subsequent life-cycle cost structures (Zehbold, 2001). Evaluation of architectural decisions, in the automotive industry at least, is still focused on short-term quantifiable goals and based on the experience of system architects.

The strategy of optimizing automotive E/E architecture to reduce unit costs, in connection with growing variability of products and functional size has in the past led to an often criticized explosion of complexity (Negele, 2006), reflected in an exponential increase of variability, heterogeneity, distribution and interdependencies in automotive E/E systems (Broy, et al., 2006). The costs of complexity, such as increased integration costs, higher error rates and missed opportunities due to limited design space, are responsible for high expenses in architecture development which do not contribute to the realization of requirements or perceivable increase in vehicle value. As neither the automotive market, nor the customer’s willingness to pay for in-vehicle electronics grows as fast as costs related to E/E functionality, it is foreseeable that future innovations will be even more limited by the costs of complexity.

The following work aims at improving the quality of architecture analysis and prediction of consequences resulting from architectural decisions. We present a quality model for E/E systems reflecting costs, risks and opportunities in its life-cycle. We discuss the life-cycle of automotive E/E systems at distinct levels of abstraction and variability and derive quality goals for their architecture. We relate these quality goals to structural properties of architecture and external factors driving the costs of development, maintenance, production and operation. Finally we analyze selected architectural design decisions with respect to their long term effects on life-cycle costs.

# Kurzfassung

Architektur in Elektrik-Elektronik Systemen im Fahrzeug spiegelt eine Reihe von Einflüssen und Abwägungen wider: etwa zwischen funktionalen Anforderungen, Kosten, historischen Entscheidungen und organisatorischen Strukturen. Bereits (Parnas, 1976) stellte fest, dass Architekturentscheidungen zu den frühesten Entscheidungen bezüglich dieser Abwägungen gehören. (Fowler, 2002) erwähnt, dass Architekturentscheidungen diejenigen sind, die in der weiteren Entwicklung am schwierigsten zu verändern sind, während (Zehbold, 2001) aus wirtschaftlicher Sicht Architektur als den größten Hebel für Kostenstrukturen im späteren Lebenszyklus identifiziert. Zumindest im Automobilbau wird gerne der Vorwurf erhoben, dass solche Architektur-Entscheidungen anhand von kurzfristigen Wirtschaftlichkeitsaspekten bewertet und getroffen werden und auf der Intuition von Systemarchitekten basieren.

Die Fokussierung auf Produktionskosten und die zunehmende Anzahl von Sonderausstattungen und Fahrzeug-Baureihen hat in der Vergangenheit zum häufig bemängelten, explosionsartigen Anstieg der Komplexität der Architekturen geführt (Negele, 2006). Messbare Größen dieser Komplexität sind z.B. Heterogenität, Verteiltheit und Abhängigkeiten innerhalb von E/E Systemen im Fahrzeug (Broy, et al., 2006). Folgen dieser Komplexität sind erhöhte Kosten für Integration, höhere Ausfallraten und verpasste Marktchancen durch eingeschränkte Erweiterbarkeit des Produktes. All diese Nachteile führen zu höheren Aufwänden in der Entwicklung, ohne dass zusätzliche, kundenwerte Funktionen realisiert würden. Da weder der Automobilmarkt, noch die Bereitschaft des Kunden, den zusätzlichen Aufwand zu bezahlen, so schnell wächst wie die Kosten für Elektrik/Elektronik, ist es abzusehen, dass in Zukunft Innovationen durch die Komplexität der Systeme weiter eingeschränkt werden.

Ziel dieser Arbeit ist es, die Systematik und dadurch auch Qualität der Architektur-Bewertung zu verbessern und die Konsequenzen der zu treffenden Architektur-Entscheidungen vorherzusagen. Wir präsentieren ein Lebenszyklus-Modell für E/E Systeme und leiten daraus Qualitätsziele für die Systementwicklung ab. Ferner diskutieren wir den Einfluss struktureller Eigenschaften der Architektur auf die Erreichung dieser Qualitäts-Merkmale. Schließlich bewerten wir ausgewählte Architekturentscheidungen anhand der vorgestellten Methodik.

# Danksagung

Mein tiefer Dank gilt zuallererst Prof. Dr. Dr. h.c. Manfred Broy, der mir die Möglichkeit eröffnete, an diesem anspruchsvollen und relevanten Thema zu arbeiten, mich das Thema hat frei ergründen lassen und der mich an den entscheidenden Stellen in die richtige Richtung gewiesen hat. Ebenso möchte ich Prof. Dr. Andreas Herkersdorf für die wertvollen Hinweise und Anregungen zu Beitrag und Methodik der Arbeit danken.

Ebenfalls danken möchte ich den vielen Kollegen bei BMW, die mich mit ihren Hinweisen, Erfahrungen und ihrem Wissen unterstützt haben und zum erfolgreichen Abschluss dieser Arbeit beigetragen haben. Persönlich erwähnen möchte ich an dieser Stelle Dr. Wolfgang Hopfensitz, Dr. Hans-Jörg Vögel, Hans-Ulrich Michel, Dominik Schnieders, Hans-Joachim Wetzel und Rinat Abdulkarimov, sowie Mitdoktoranden bei der BMW Group, Stéphane Turlier, Florian Burgdorf, Tuan Le und Jens Kohl, für ihre Hilfe bei der Recherche, wertvolle Hinweise und Korrekturen.

Nicht zuletzt möchte ich auch meiner Familie und Freunden für die Unterstützung und ihren Glauben an den erfolgreichen Abschluss dieser Arbeit danken.

# Table of Contents

# List of Figures

# List of Tables

# 1 Introduction

This chapter gives an overview of the topic covered and outlines the approach taken in this work. We present the motivation and relevance of the investigated problem, state the question to be answered and define the abstractions and concepts by which to approach the subject. Finally we discuss the methodology used in order to answer the questions asked and verify the hypotheses postulated in the process of this work.

## 1.1 Motivation

The value provided by electronics and software in modern vehicles and the number of services realized by E/E systems grows constantly. At the same pace dependency of vital functions on software and electronic components increases, as does the share of development and production costs of hard and software in the overall costs of a vehicle. Various sources mention a number of 40% of value[1] and 90% of innovation[2] within a vehicle being generated with electronics and software.

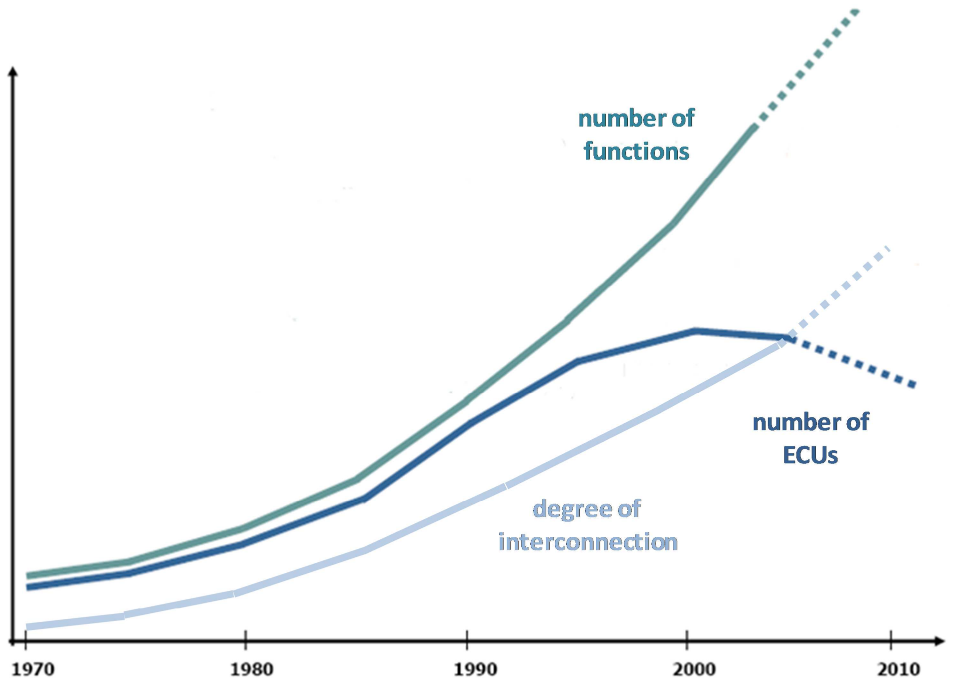


Figure 1 Structural evolution of automotive E/E systems, adapted from (Negele, 2006)

Manufacturers and suppliers face a growing demand to implement new functionality or shift functionality from mechanical parts to electronics and software, in order to increase comfort, safety, energy efficiency and distinguish their products from competitors. At the same time pressure to lower unit costs imposes narrow bounds for implementation. Figure 1 depicts this trend – a growing number of functions accompanied by

[1] e.g. (Magna, 2005), (Blanco, 2010), (Rhines, 2007)
[2] e.g. (Negele et al., 2005), (Bielefeld, 2005), (Fröberg, 2007) referring to a Siemens VDO study

increasing interconnectedness and functional density of electronic control units.

In the past, automotive E/E architectures have been primarily optimized to reduce unit costs in production which was justified by scaling effects according to which small savings at the vehicle level almost always outweighs development costs (Broy, et al., 2006). This strategy has led to what is commonly referred to as *explosion of complexity* or more precisely an exponential increase in variability, heterogeneity and interdependencies of in-vehicle E/E systems.

The strategy to optimize E/E architecture to unit costs already reaches its limits when considering costs at a more holistic perspective:

- Integration and tests consume a significant part of development costs of the E/E system of a new product line.
- Maintenance of a large number of distinct platforms and variants binds resources in architecture and development departments without adding value to the product.
- Errors remain undiscovered until delivery, as variability of implementation prevents all possible feature configurations to be tested.
- Generation changes of platforms consume more resources than necessary, due to high coupling between functionality and implementation.
- The large number of component variants prevents scaling effects to be realized in purchase and production.

All of these issues document the paradox situation that the more value is generated with in-vehicle electronics and software, the fewer resources remain to develop innovative features and realize efficiency gains. It is foreseeable that in future budgets for innovation will be eaten up by costs of integration and maintenance, unless significant effort is made to limit the complexity of current in-vehicle E/E architectures and increase the communality between product lines and models. This can only be achieved, by improving the understanding of long-term effects of architectural decisions.

The main challenge of automotive E/E architecture design is to manage variability of requirements and constraints and growth of functionality on the one hand, while controlling unit, development, integration and maintenance costs on the other.

To cope with the growing variability of requirements and heterogeneity of solutions OEMs have turned towards platform and construction kit strategies, to leverage on commonalities between vehicle models and classes and reduce variability of solutions to an economically justifiable level. Construction kits and platforms however have decoupled architectural decisions from the evolution of requirements. Architectural decisions on the construction kit level are made long before concrete requirements are elicited and bear major uncertainty regarding their value for concrete sets of requirements and constraints.

Despite uncertainties regarding concrete requirements and take-rates at the time decisions are made, a structured, systematic comparison of architecture alternatives is certainly useful and necessary. Architecture evaluation is currently mostly based on the responsible architect's experience. Even though most engineers do have the experience and obviously do a good job, an unstructured evaluation and decision process is apt to trends and subjectivity, applying criteria which are easy to evaluate or favoring some criteria over others.

The premise of this work is that criteria and binding assumptions can be made transparent in a quality model and methodology for architecture evaluation.

## 1.2 Problem statement

This work aims at improving the quality of evaluation of architectural decisions. It does so by investigating the **impact of architectural structure** on **costs and quality of E/E system**s and thus to prevent decisions to be made in practice which locally reduce one type of costs, but globally lead to a dead end of unmanageable complexity. Apparently current practice does not adequately account for complexity introduced through architecture decisions or does not sufficiently help to trade off additional complexity against immediately quantifiable goals such as production costs efficiency.

Intuitively, the key to sustainable architecture for automotive E/E systems is to capture the short and long term effects of architecture decisions and to determine a cost-optimal partitioning of subsystems and components and allocation of functions. We subsume the suitability with respect to activities in the life-cycle from the perspective of any stakeholder as quality of the system.

The first part of this work is to break down the general concept of quality into a set of well defined, independently assessable and quantifiable goals. Secondly the achievement of quality goals by candidate architectures needs to be assessed in early stages of the life-cycle based on structural properties. Finally, we validate the presented model and methodology by evaluating a number of architectural measures with respect to their consequences on costs, risks and opportunities in the life-cycle of the E/E system.

By considering a larger number of quality goals than currently we increase the probability that architectural decisions taken actually help to achieve an economic optimum. By investigating the relationship between structure and the achievement of the economically dominant quality goals we help to appropriately weight and trade off conflicting goals.

### 1.2.1 Definition of Evaluation Criteria

To derive the set of desirable properties and optimization criteria for in-vehicle E/E systems we discuss their life-cycle of and point out cost drivers in different life-cycle phases. A basic life-cycle model is presented in Figure 2 and spans development, manufacturing, use and maintenance.

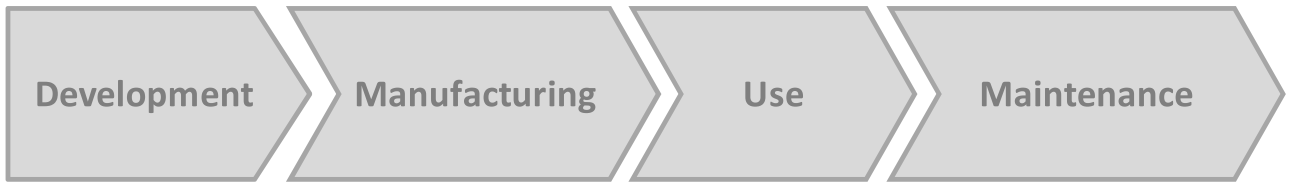


Figure 2 Generic life-cycle model for engineering systems

In addition to the chronological separation of life-cycle phases we need to consider the subject of evaluation and the scaling factor of potential quality properties. Whereas production, service and warranty costs of a vehicle scale with the number of units produced, the development and integration costs of a new component occur only once per vehicle product line. Finally decisions regarding the construction kit and architectural blueprint affect potential synergies and thus benefits in development effort and correctness of each product line.

With these initial considerations we define a hierarchic life-cycle model for the definition of evaluation criteria:

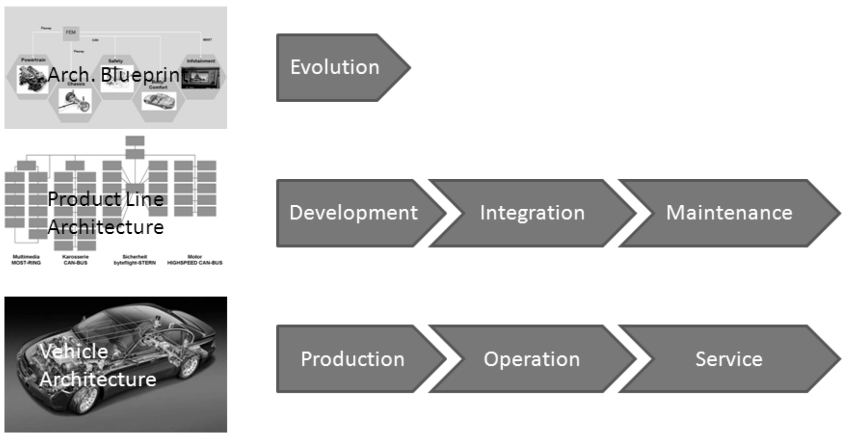


Figure 3 Hierarchic life-cycle model for In-vehicle E/E systems

The **vehicle architecture** level contains all properties related to a single vehicle, both in terms of individual feature configuration as well as its life-cycle, i.e. production, operation and service.

The **product line architecture** level targets properties related to development and integration, but also at the average cost efficiency of individual vehicles, a.k.a. scalability.

Finally the **architectural blueprint** level describes the communality, thus benefits of reuse between individual product lines, as well as issues of functional evolution of architecture from one product line generation to another.

Since the lower levels inherit the architectural design decisions from the higher levels, the architecture blueprint level can be evaluated with respect to the quality properties of product line architectures derived from it. Analogously the product line can be assessed by the properties of derived vehicle architectures to the extent their structure is defined by the product line.

### 1.2.2 Definition of Architectural Means

The achievement of the evaluation criteria defined up front can be measured in the resulting system as well as in economic key-figures. However, it is the properties of architecture as an artifact that has to be controlled by the system architect in the design phase. To define the scope of architecture and the architectural means of optimization we relate to the model defined by (Wild, et al., 2006) and presented in Figure 4.

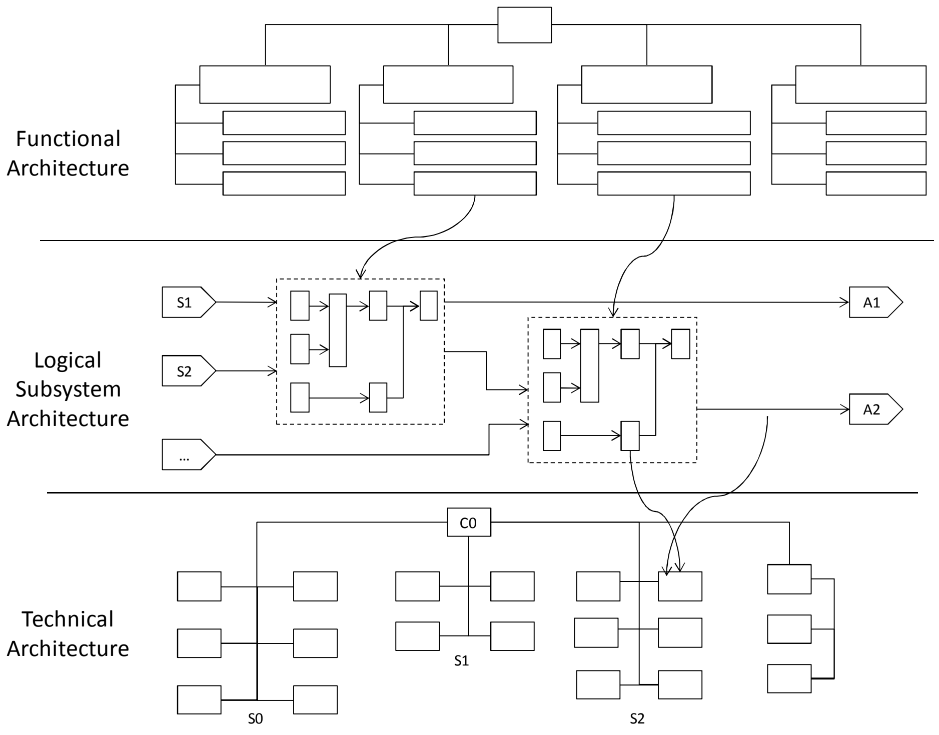


Figure 4 Architecture description model

The model described therein consists of three layers: the functional architecture, the logical subsystem architecture and the technical architecture. Whereas the functional architecture defines the set of requirements and is thus not subject to architectural design, the architectural decisions made in the design phase affect the **realization of functions by logical components and signals**, the **subdivision of the vehicle's E/E system into systems and components** and the **allocation of logical components onto physical components** and signals on communication buses.

Based on this model, as well as documented architecture changes we identify typical means affecting the performance of architecture with respect to costs and quality. These means are:

**Component Integration and Segregation** aims at the total number of components in order to leverage efficiency gains in production. The measurable structural key-figure is the **functional density** of individual components, i.e. the average number of logical functions provided by a component.

**System Integration and Segregation** aims at the total number of subsystems (independent communication buses) in order to reduce the extent of wiring and gateway functionality within the vehicle, and ease verification and validation through functional separation.

**Function Allocation** aims at deployment of units of functionality defined at the logical layer in order to reduce inter-component communication, as well as potentially ease development by co-locating related functions on one component. Measurable structural properties related to allocation are **low inter-component coupling**, **alignment with organizational structure** or with **alignment with patterns of use**.

**Component Allocation** aims at deployment of components to communication buses in order to optimize the inter-domain communication spanning more than one communication bus and gateway nodes.

**Variability of Components and Systems** aims at adaptation of physical resources installed in the vehicle to the set of functions installed in order to reduce the average extent of resources and thus production costs. Variability can be introduced through additional component variants or through optional components resulting in a variable infrastructure.

### 1.2.3 Definition of Scope

Given the set of evaluation criteria of E/E systems and a set of architectural means and metrics we relate structural properties of architecture to the quality goals and study their correlation. To limit the scope and complexity of investigation, we make a number of simplifying assumptions and exclusions regarding the subject of study.

The first simplification concerns the degrees of freedom of architecture development. We assume both the set of functional requirements, i.e. the functions to be provided by the vehicle and their design, represented by the logical subsystem architecture, to be fixed. Thus, the means of optimization are limited to the ones listed in section 1.2.2 . Conversely we do not discuss the elimination of functionality, or re-design of the logical component-signal chains in the logical subsystem architecture as means of optimization.

The second exclusion concerns the means of architecture design and optimization. We restrain our analysis to the *structure* of the E/E system as opposed to *semantics*. Architectural means discussed can be expressed by the number of components, systems and allocation of functions to components and components to systems respectively, not by the abstraction or granularity of interfaces between components or systems. Likewise we do not discuss the structuring of functional layers of software within a single component in order to reduce the extent of individual development required to implement a customer function.

## 1.3 Methodology

In order to bring some structure and methodology into architecture evaluation, we need to define the quality goals and identify the architectural means and trade-off points. In the process of this work we apply both empirical and analytical methods.

In the first part we analyze documented architecture changes and their evaluation to gain an understanding of the economic relevance of architectural decisions and assessment criteria defined so far. In addition to illustrating the economic dimensions of architecture decisions, the results will help to identify the issues in architecture evaluation models and methodology to be addressed by this work. The changes investigated and documented therein are by definition bound to concrete functional and organizational constraints and can thus only partially be generalized, however such ‘anecdotal evidence’ can support the discussion of relationships derived in the subsequent part or falsify them.

In order to define a methodologically consistent, holistic quality model we will subsequently analyze the life-cycle of in-vehicle E/E systems and define measurable goals for their quality. The derivation of measurable quality goals in distinct life-cycle phases is an analytical process backed by deduction and common sense. Likewise, the relation of structural properties to quality goals is a process of invention.

Since the subject of study is actual vehicle projects and components, the postulated relationships cannot be tested in an experimental setup where factors extrinsic to the architectural structure are normalized for comparability. In this work we can only relate to documented analyses in the first part and expert interviews for validation.

As a proof-of-concept of the given methodology, we apply the criteria defined previously to concrete architectural decisions to verify, whether founded and systematic conclusions about system quality can be drawn from structural analysis.

## 1.4 Structure

The rest of this thesis is structured as follows:

**Chapter 2**

We summarize related work in the fields of quality models and quality goal definition. The case for an in-depth study of quality of automotive E/E systems has been made by (Broy, et al., 2006) and (Wallin, et al., 2008). We elaborate on the work on model based, structural architecture analysis (Fröberg, et al., 2006), (Larses, 2005) and (Chen, et al., 2004) and (Thiel, 2005) and identify the research gap to be closed by this work.

For a methodology for architecture analysis we refer to work by Kazman et al. particularly the Architecture Tradeoff Analysis Method (Kazman, et al., 2000) and Architecture Options (Engel, et al., 2008), as well as modularity analysis (e.g. (Baldwin, et al., 2006)).

**Chapter 3**

We present a study on architectural decisions and their evaluation for economic benefits and long-term effects. We derive a list of evaluation criteria and relate these to typical architectural changes. Finally we discuss a possible prioritization of the derived criteria. Based on this analysis we identify the deficiencies in the current approach to architecture evaluation to be targeted by a comprehensive and structured quality model.

**Chapter 4**

We analyze the levels of architecture definition and discuss the activities constituting their life-cycle. Using the life-cycle phases for the categorization of quality goals, we describe the cost structures and constraints related to E/E system architecture at each level.

The cost, risks and opportunities in the life-cycle yield desirable properties to optimize the architecture for. The resulting list of properties will serve us in the following chapter to define structural properties contributing to the achievement of these quality goals.

**Chapter 5**

We define quality goals for each level of architecture instantiation and life-cycle phase and discuss the relationship between structural properties and quality goals. We illustrate our analysis by empirical data from a number of vehicle development projects. We discuss distribution and redundancies as a negative properties leading to disadvantages in vehicle related qualities, and variability as a negative product line related qualities leading to increased verification and validation costs and risks regarding system correctness.

**Chapter 6**

Having identified the quality goals for E/E systems and discussed the relationship between structural properties and the achievement of these goals, we investigate the differences between two common strategies applied in E/E system design: modularization and centralization accompanied by component variability.

**Chapter 7**

We document several architectural case studies aiming either at adding functionality to an existing in-vehicle architecture or improvement of quality properties by modifying the structure of the E/E architecture. Based on the quality goals and methodology defined previously, we discuss the consequences of these measures on quality properties of the respective and derived architecture instances.

We compare the benefits of centralization and the resulting savings at the production level against the costs of development, integration and evolution due to decreased modularity and increased functional density.

**Chapter 8**

We summarize our findings and extrapolate them for a scenario of increasing requirement variability and number of derivatives and outline open issues for further research.

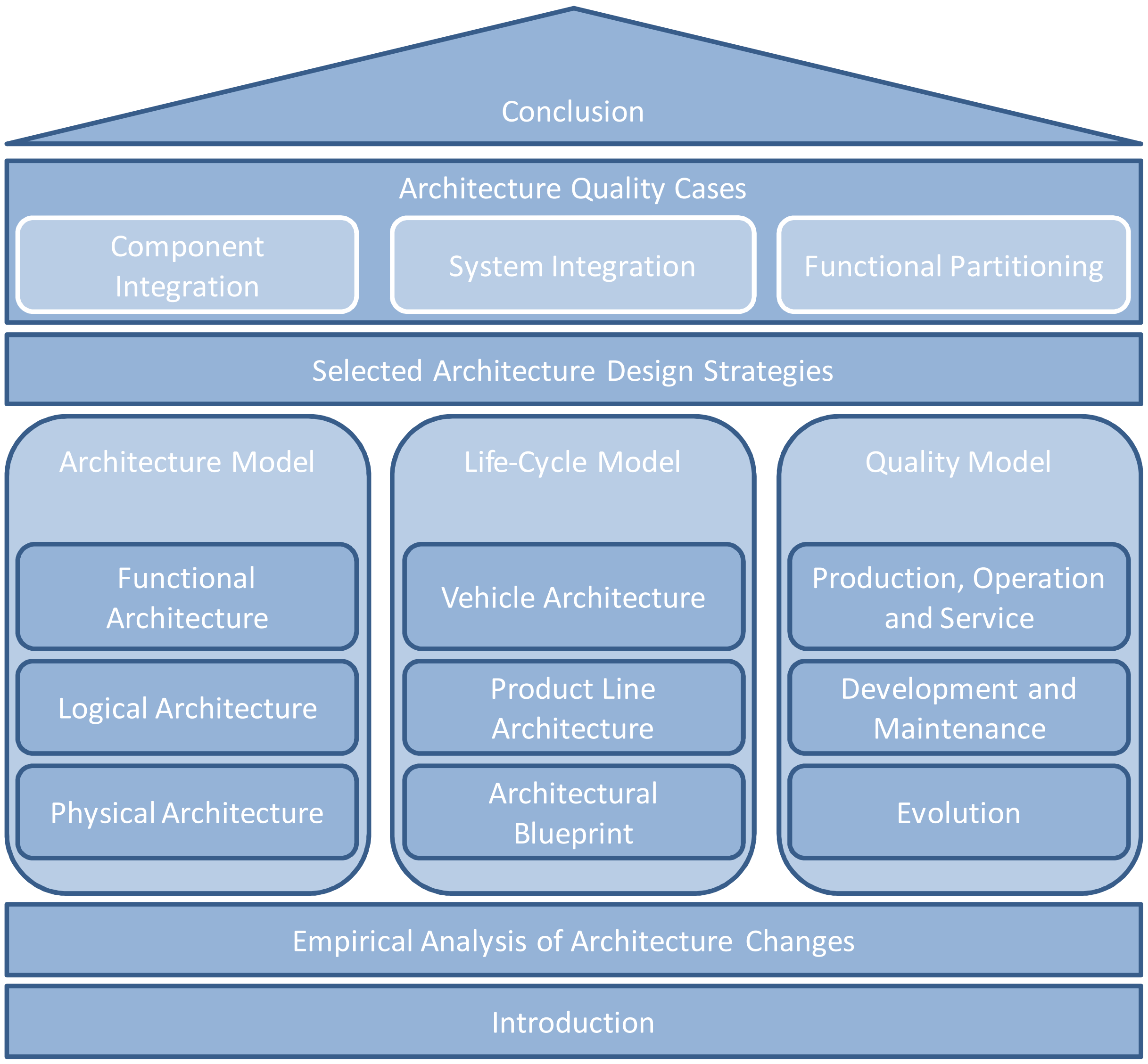


Figure 5 Structure of this thesis

# 2 State of the Art

We are about to investigate the impact of *architecture* on *quality* of automotive E/E systems. To approach the state of the art in subject we discuss the concept of quality in general as well as the specific questions of quality in automotive E/E systems and methodology of architecture evaluation.

## 2.1 Concept of Quality

Even though the concept of quality may seem intuitively clear, its interpretation and concrete definition varies widely among researchers. The earliest work on quality to be referenced in this work is (Shewhart, 1931), which is still considered the most holistic and thorough in its discussion, and will be picked up later in this chapter. At least two distinct interpretations of quality can be distinguished among researchers since then:

- Quality is conformance to specification, and can be measured as the number of defects (non-conformance)
- Quality is meeting customer's needs, thus can be judged subjectively by the consumer or any stakeholder of the end product

Prominent proponents of the first definition are (Crosby, 1979) and (Juran, 1988). Crosby advocates an objective definition of quality as opposed to common meaning of "goodness" or "elegance" as these notions cannot be measured nor systematically managed. He writes:

*"Quality must be defined as 'conformance to requirements' if we are to manage it. Consequently, the nonconformance detected is the absence of quality, quality problems become nonconformance problems, and quality becomes definable."*

He states that quality must be achieved by preemptive measures, not by inspection. The means to achieve quality are therefore an engineering and production process aiming at zero defects. Despite using the term "fitness for use", (Juran, 1988) likewise tends to the "conformance to specification" perspective and points out that satisfying customer expectations is indeed very hard to achieve. Any directed action towards improving quality obviously needs verifiable goals. His recommendations are planning, control and improvement of quality, thus aiming at the process aspect of engineering and production.

Proponents of the "customer needs" standpoint are more numerous: e.G. (Deming, 2000), (Feigenbaum, 1983) and Ishikawa (Ishikawa, 1985) . Deming states that:

"*The difficulty in defining quality is to translate future needs of the user into measurable characteristics, so that a product can be designed and turned out to give satisfaction at a price that the user will pay.*"

Thus quality is not a task for engineering and manufacturing, but a goal for the whole organization and should be introduced as a philosophy and a culture. His recommendations are stated as 14 "Points for Management" and focus on promoting the goal and instantiating a culture of constant improvement, by leadership, responsibility, training and so forth. Feigenbaum's definition is likewise very holistic, as the observation that quality is in fact experienced by the customer, leads to the conclusion that improvement of quality involves equally management of the product itself, as well as of customer expectations. Ishikawa likewise extends the concept of quality to basically all aspects of the organization, that is: service, information, people, systems and so forth.

Comparing both of the definitions stated above, i.e. the “customer needs” perspective and the “conformance to specification” perspective, it becomes apparent, that the key difference between the two positions is whether customer expectations can be stated up front, as clearly in that case, the specification and the customer needs are identical. Whereas the advocates of the “customer needs” philosophy emphasize the elicitation (and also management) of customer needs and constant change as key to improvement, the proponents of the “conformance” philosophy stress the importance of processes and prevention.

A theory which brings both definitions together has been formulated by Shewhart (Shewhart, 1931). Interestingly it is the oldest publication on quality referenced, and evidence that questions re-occurring nowadays may have been studied and answered some tens of years ago. Shewhart writes:

*“If we are to talk intelligently about the quality of a thing or the quality of a product, we must have in mind a clear picture of what we mean by quality. Enough has been said to indicate that there are two common aspects of quality: one of these has to do with the consideration of the quality of a thing as an objective reality independent of the existence of man. The other has to do with what we think, feel, or sense as a result of the objective reality. In other words there is a subjective side of quality. (…)*

*From the viewpoint of control of quality in manufacture it is necessary to establish standards of quality in a quantitative manner. For this reason we are forced at the present time to express such standards, insofar as possible, in terms of quantitatively measurable physical properties. This does not mean however, that the subjective measure of quality is not of interest. On the contrary, it is the subjective measure that is of commercial interest.*

*(…) The first step of the engineer in trying to satisfy these wants is therefore, that of translating as nearly is possible these wants into the physical characteristics of the thing manufactured to satisfy these wants.”*

In other words it is the subjective quality which we need to improve, however it is only the objective aspect one can control. By specifying the quality of the product under development in terms of objective attributes, we are trying to come as close as possible to the subjective ideal customers are paying for. The measurable properties we need to specify are the subject of the following chapter.

## 2.2 Software Quality Models

The aforementioned statements of quality gurus are certainly insight- and helpful from a management standpoint. The advice given there, particularly to understand the customer, to focus on people and instantiate a culture and to aim at continuous improvement is one of permanent validity and importance. However from an engineer’s standpoint the lack of concrete criteria to assess the product against and guide the design process make these philosophical works interesting to read, but of little help in day-to-day work.

What are the characteristics of the product by which we are trying approximate subjective experience of quality? At least for software products, a number of such *quality models* have been defined, the most renowned ones being (McCall, et al., 1977), (Dromey, 1995) and finally the ISO 9126 standard and its successors.

**McCall**'s quality model intends to bridge the gap between quality perceived by the user and quality designed by the engineers. McCall derives quality attributes from three "perspectives", i.e. product revision (changes to the product), product transition (adaptation to a new environment) and product operation (its perceived characteristics). To support the design and evaluation process from these perspectives, McCall defines a hierarchy 11 factors (to specify) 23 quality criteria (to build) and a number of Metrics (to control). The quality evaluation by McCall is conducted by evaluating the metrics on a yes/no scale. Thus answering an equal number of evaluation questions with yes and no would result in a 50% achievement of quality (within the respective criterion).

**Boehm** likewise presents a hierarchic model of properties. The top level categories address the three main questions related to software as a product of economical use: *as-is-utility*, *maintainability* and *portability*.

The parallels to McCall's model with its distinction into product operation, revision and transition are obvious. However the refinement into intermediate- and primitive characteristics yields a slightly different list of properties. Since both models were developed about at the same time, there is no substantial difference in the philosophies behind them. One could say, that Boehm concentrates on software maintenance cost-effectiveness which, as he states is the primary payoff of increased quality of software (Berander, et al., 2005).

**ISO 9126** is somewhat the unification of the models listed previously and is structured in the same manner, i.e. in a hierarchical utility tree. A comparison of McCall's, Boehm's and the ISO 9126 model is listed in Table 1.

| Criterion / Goal | McCall (1977) | Boehm (1978) | ISO 9126 (2001) |
|---|---|---|---|
| Correctness | * | * | maintainability |
| Integrity | * | * | * |
| Usability | * | * | |
| Efficiency | * | * | * |
| Maintainability | * | * | * |
| Testability | * | | * |
| Interoperability | * | | maintainability |
| Flexibility | * | * | |
| Reusability | * | * | |
| Portability | * | * | |
| Clarity | | * | |
| Modifiability | | * | |
| Documentation | | * | maintainability |
| Resilience | | * | |
| Understandability | | * | |
| Validity | | * | maintainability |

| | | | |
|---|---|---|---|
| Functionality | | | * |
| Generality | | * | |
| Economy | | * | |

Table 1 Software quality criteria (McCall 1977), (Boehm 1978), (ISO 9126) based on (Lundeberg 2005)

In addition to unifying Boehm's and McCalls model, ISO 9126 includes Functionality as a top level factor and distinguishes between external quality and internal quality.

**Dromey**'s model is one of the more recent ones. His analysis is based on the observation that the discussion of (software-) quality has stalled among others due to the difficulty of attributing high-level quality attributes down to tangible properties. According to Dromey the problem has two underlying reasons: primary there is an abundance of properties affecting the quality of the product, secondly, there is no formal basis to relate product properties to quality attributes.

His main contribution is that quality evaluation differs for each product and the properties affecting quality differ depending on the artifact under consideration. Thus quality metrics such as the ones defined in the models presented above are too generic or inadequate for practical purposes. He suggests the use of the ISO 9126 as a starting point to identify quality attributes of each set of artifacts and relate the properties of components in these artifacts to the quality attributes.

Dromey is focusing on the relationship between the quality attributes and the sub-attributes as well as attempting to connect software product properties with quality attributes. In that respect Dromey's approach is similar to the one applied in this work. By inspecting (structural) product properties we would like to reason about quality properties. Whereas Dromey's model applies explicitly to software, and thus contains properties such as initialization of variables and computability of expressions, our work concentrates on structural properties of the architecture model.

Dromey's evaluation method is based on a 5 step elicitation process consisting of the steps described below:

1. Define quality attributes to be satisfied by the product
2. Describe the structure of the system
3. Identify quality carrying properties for the components/modules
4. Determine how each property affects the quality attributes
5. Evaluate the model and identify weaknesses

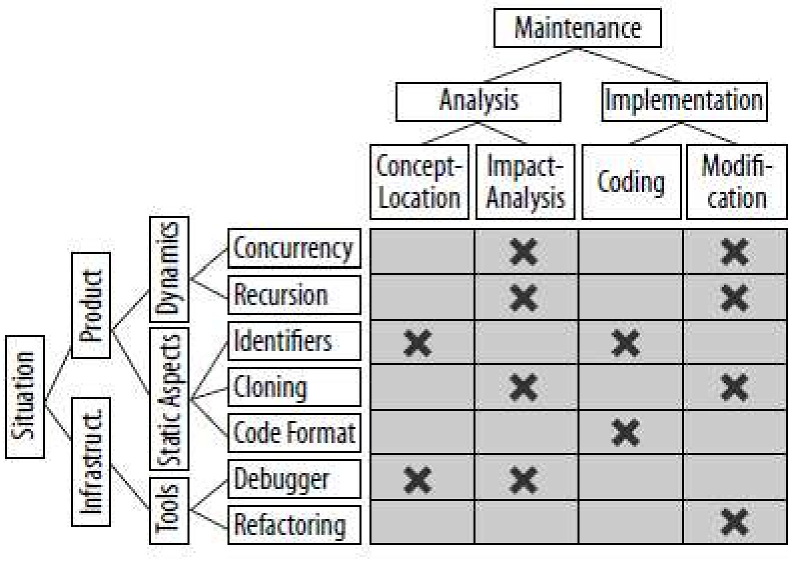
The interesting point here is mainly that quality is affected by structure (quality carrying properties) and that quality evaluation can be performed by identifying inefficiencies within the structure. We will return to that when discussing evaluation of automotive E/E systems in the chapters 5 6 and 7 . For the current discussion, we should emphasize Dromey's point that a quality model should be developed individually for each product. Elements of the artifacts under evaluation should be identified and their properties related to high level attributes. Components in our discussion are the elements of the architecture model defined in chapter 4. Quality attributes will be derived based on a discussion of the life-cycle of the system under development.

Keeping in mind the models presented so far we have to deal with two mainly independent questions:

- What is the set of characteristics which approximates perceived quality and how do we know the characteristics comprehensively describe quality of the product
- How can we categorize these characteristics and what do we gain by choosing one categorization over another

One possible approach is described in (Wagner, et al., 2007). The quality model, or rather - meta-model for describing quality distinguishes *activities* to derive desirable properties of the product from and *facts* to guide design and evaluation. These correspond to point 1 and 3 of Dromey's evaluation process. Drawing the activities and facts in a two dimensional diagram yields a quality matrix indicating the dependencies between measurable properties of the product structure and quality goals we are trying to achieve. An example fact-activity diagram for maintenance activities as cited by (Wagner, et al., 2007) is depicted in Figure 6.

An alternative approach is to derive quality criteria from top-level strategic or economic goals of the organization and relate them to measurable properties supporting these goals, commonly known as the Goal-Question-Metric methodology (Basili, 1992), (Basili, et al., 1994). Whereas goals identify the benefit of achieving quality defined in this way, and metrics denote controllable properties of the product under development, questions are intended to create an argumentative chain between the goals and the metrics. An application of GQM to the automotive domain is presented in (Schorer, 2007) and is depicted in Figure 7.



| | Concept-Location | Impact-Analysis | Coding | Modifi-cation |
|---|---|---|---|---|
| Concurrency | | ✗ | | ✗ |
| Recursion | | ✗ | | ✗ |
| Identifiers | ✗ | | ✗ | |
| Cloning | | ✗ | | ✗ |
| Code Format | | | ✗ | |
| Debugger | ✗ | ✗ | | |
| Refactoring | | | | ✗ |

Figure 6 Fact-activity-matrix for maintainability (Wagner, et al., 2007)

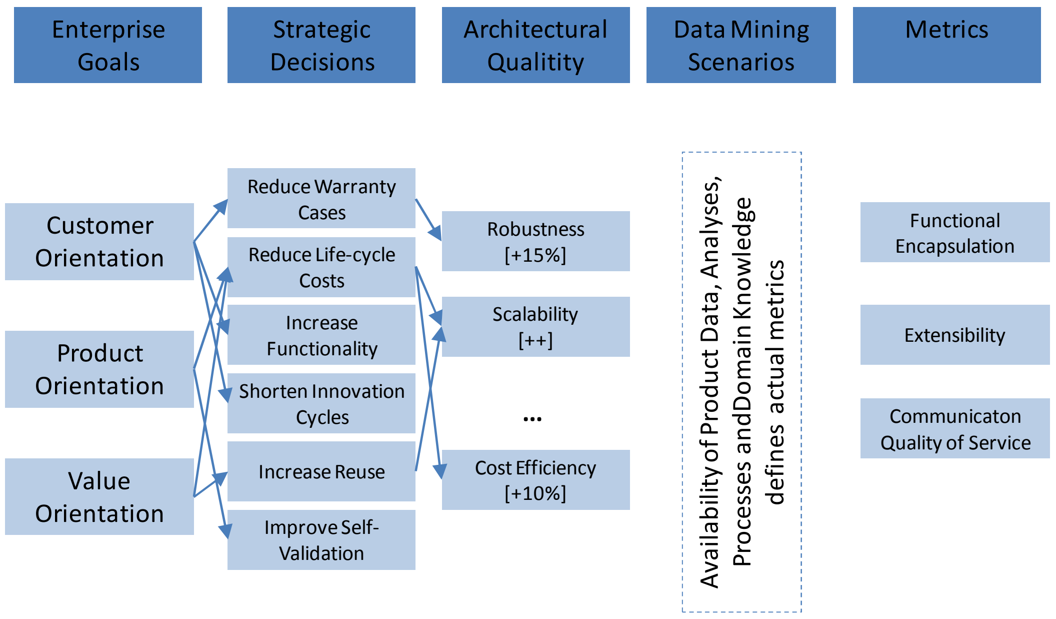


Figure 7 Application of GQM to automotive E/E systems (Schorer, 2007)

Both of these approaches exemplify the fact that the relation of structural properties of the product to quality goals is a process of invention. A correlation between metrics and goals can be more or less intuitive. Empirical evidence that the optimization of a particular metric in fact leads to the achievement of a goal is in practice hard to find, due to external factors affecting the achievement of goals. The binding of properties to economically measurable quality goals is mostly conducted argumentatively.

For the definition of a quality model for automotive E/E systems, we will adopt both of these approaches – on the one hand by investigating the life-cycle of the E/E system architectures to identify relevant activities and by refinement of these activities into quality goals in chapter 4 and 5 . On the other hand, we will define structural properties based on an architecture model representing the most fundamental decisions to be made by a systems architect, i.e. system structure, component definition and function partitioning.

The quality model resulting out of this work can then be represented as a matrix similar to the one presented in Figure 6. The quality with respect to individual activities is defined as key-figures which in turn are positively and negatively related with structural properties. The actual kind of relationship between architectural structure and related activities can take different forms such as increased effort in development, integration, risk of errors undetected until delivery or costs in production, operation and service.

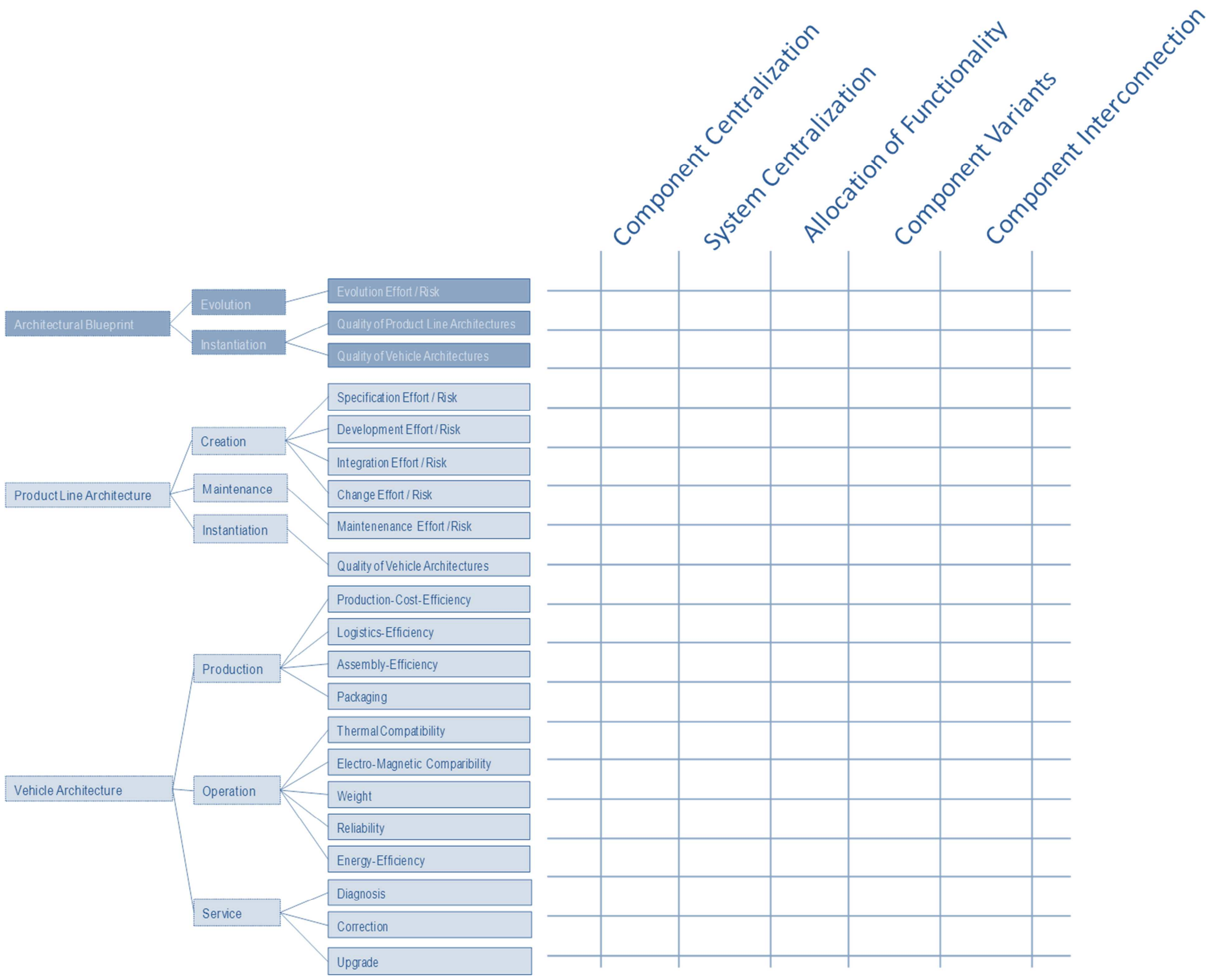


Figure 8 Quality-structure matrix for automotive E/E systems

## 2.3 Automotive Electronic Systems

(Dromey, 1995) concludes that generic quality models are too abstract to adequately guide the design process and suggests creating a domain specific quality model for each product. This becomes apparent when trying to apply the ISO 9126 Model to automotive software and electronic systems. Criteria such as efficiency or testability can, with a little stretch, be adapted to automotive electronics, while portability can hardly be evaluated in a mechatronic system. Conversely, the complete set of quality properties arising from the production process and the fact that the produced system is a physical entity do not fit into that grid.

The specific requirements and quality criteria of automotive E/E systems have been discussed by (Axelsson, 2009), (Larses, 2005), (Fröberg, 2007) and (Wallin, et al., 2008) besides, individual sets of evaluation criteria exist within the architecture departments, such as the one presented in Chapter 3. We focus on the quality goals and evaluation criteria, and the grouping of such criteria into categories.

(Axelsson, 2009) distinguishes *feasibility*, *consistency*, *optimality* and *modifiability* as top-Level criteria or concerns that govern architectural design.

- *Feasibility* is the possibility to implement the specified functionality by the available computational resources
- *Consistency* is the requirement of completeness of component and interface definition
- *Optimality* reflects the efficiency of implementation with respect to important quality attributes (including costs)
- *Modifiability* reflects the ease of future evolution

This categorization subdivides quality requirements by the kind of acceptance criterion or method of evaluation: Feasibility expresses an acceptance criterion on the system design in terms of constraints on the system itself, available packaging space or memory and processor capacity fall into this category. Feasibility requirements result in yes/no criteria to be satisfied by candidate architectures, but do not define a ranking among possible designs satisfying these constraints.

Consistency likewise defines a requirement on the result of the design process. The subject of evaluation however is not on the engineered system itself but the design artifacts. It excludes candidate architectures which are not sufficiently defined, for instance whose requirements do not have a representation in the design artifacts, rather than those physically not implementable.

The remaining two concerns represent the common notion of quality as a mean to rank candidate architectures according to their suitability: Optimality characterizes properties which exhibit some form of functional relationship with costs or - more general – with resource consumption and with value beyond pure functionality. This is probably the most intuitive definition of quality, as efficiency given, that all requirements are fulfilled. Likewise modifiability reflects hypothetical costs and risks, which are bound to a specific modification scenario. Optimality and modifiability are quality characteristics in the sense of previously mentioned quality models, thus our quality analysis will focus on these two categories, however as we will see, a number of criteria fall into the *Feasibility* category, simply because an optimization of such criteria beyond a certain threshold does not result in gains or savings, but their violation renders an architecture alternative infeasible as will be shown in Chapter 7.

This observation that various quality goals exhibit distinct characteristics when it comes to evaluation is indeed helpful in formulation and discussion of evaluation criteria, but does not answer the question which resources are to be considered for feasibility evaluation and what costs, risks or value are to be calculated for optimization. More specific models in that respect are described by (Fröberg, 2007) and (Larses, 2005).

(Fröberg, 2007) relates his analysis to a generic life-cycle model, and concentrates on the integration activity within the development phase of the E/E system life-cycle. Thus the evaluation is limited to the success of the integration process. Regarding the set of quality goals, Fröberg refers to a (proprietary) quality model depicted in Figure 9. The model resembles the, likewise proprietary, model discussed in chapter 3, unfortunately there is no further discussion or definition of the quality goals mentioned therein.

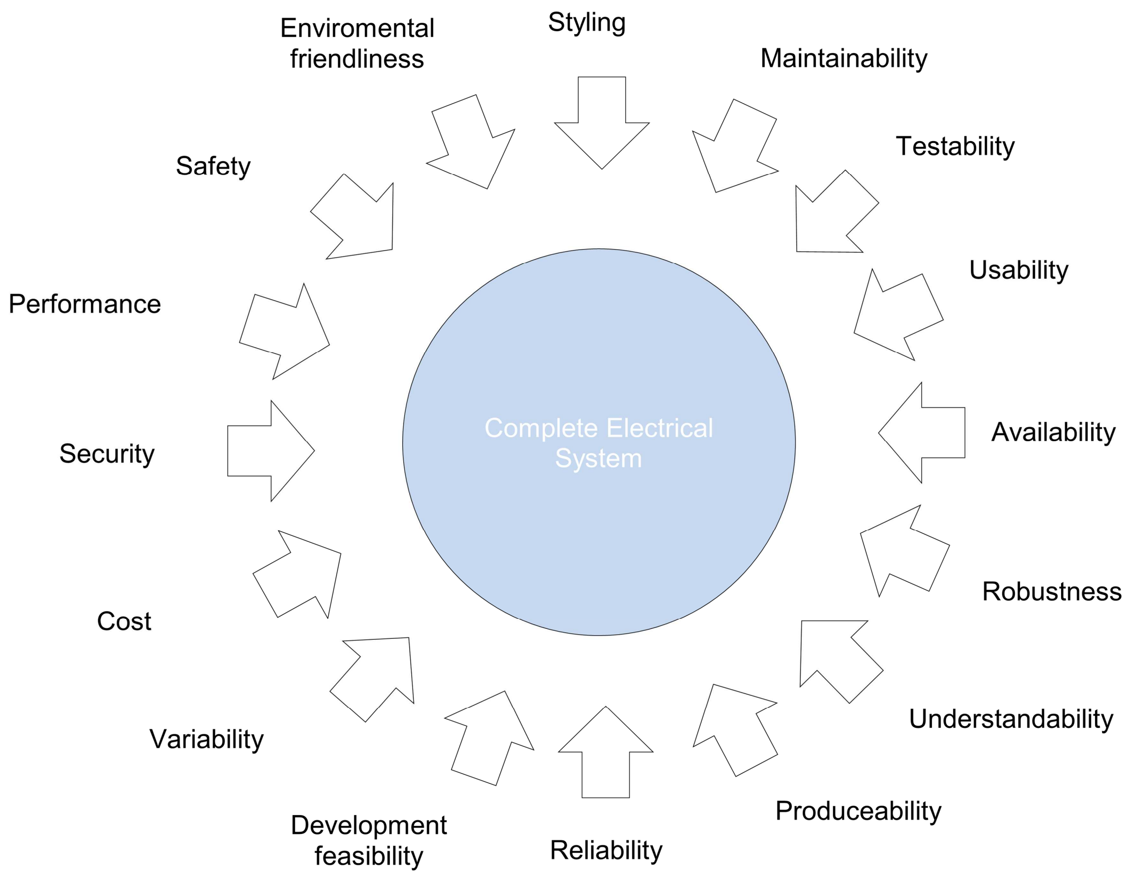


Figure 9 Stakeholder requirements for E/E systems (Fröberg, 2007)

Regarding the methodology, Fröberg investigates several integration projects, with respect to factors that affect project success. Integration projects in that context are changes to an existing architecture such as replacement or extension. Thus the assumed quality goals are defined in the integration project statement, and reflected by a graded assessment of project success. The factors relate to the management and process of integration and include such concerns as completeness of specification, knowledge about the constrains on the project imposed by the system (platform), and involvement of stake-holders and clear responsibility. However none of these criteria is in fact related to architectural structure.

Our work is related to Fröberg's in its goal, i.e. to make decisions in the development or E/E system architectures transparent and understandable. The goal of evaluation in our work is broader in scope, including not only successful completion of the architecture change, but also quality with respect to cost efficiency, scalability, modifiability of the architecture etc. On the other hand, regarding the factors contributing to architecture quality, we concentrate on structural properties i.e. properties which can be derived from an architecture model representing the services, logical design and physical components (Broy, et al., 2008).

(Larses, 2005) subdivides quality aspects of electronic control systems into *cost-efficiency* and *dependability*. Cost efficiency covers development costs, as well as production, operation and maintenance costs compared to the total product value, defined by the equation:

Cost-efficiency can thus be improved either by reducing costs associated with a given product or by improving the value of the product without incurring a higher cost. Larses admits, while costs related to a product can be accounted throughout its life-cycle, increased value, i.e. the subjective perception of user-related quality, is hard to quantify. Even if the resulting cost savings for the customer, such as increased fuel efficiency through reduced vehicle weight can be directly measured, the perceived value may differ from the actual costs. The value of other qualities such as increased vehicle safety or infotainment services is even more subjective and can be only estimated by customer inquiry.

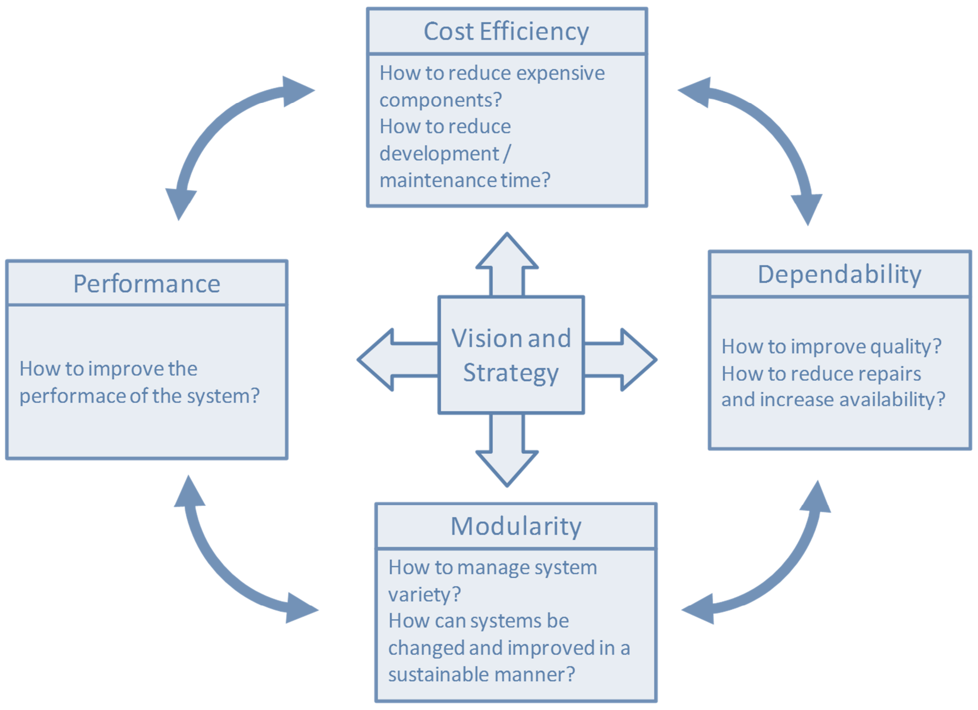

Cost Efficiency
How to reduce expensive components?
How to reduce development / maintenance time?
Performance
How to improve the performace of the system?
Vision and Strategy
Dependability
How to improve quality?
How to reduce repairs and increase availability?
Modularity
How to manage system variety?
How can systems be changed and improved in a sustainable manner?

Figure 10 Design goals of architecture development (Larses, 2005)

The complementary dimension of vehicle quality in Larses' discussion is *dependability.* Dependability spans Safety, Reliability, Availability, Confidentiality, Integrity and Maintainability basically following the definition in (Laprie, et al., 2004) and (Avizienis, et al., 2004). In his discussion, Larses monetarizes dependability with dependability costs, which are basically the costs of correction and inavailability of service.

In practice, in this work a given level of dependability the system is to satisfy is assumed. The required level of reliability is the result of a risk and hazard analysis, and the design goal of the system is to attain that level (or individual levels for each service) cost-efficiently. In that respect the requirement for reliability is in fact a feasibility criterion, whereas production and maintenance costs are the optimality criterion. Despite the comment that with “any solution that improves reliability the availability costs decrease and the maintenance cost may also decrease if fewer repairs are necessary”, the actual value of reliability benefits must be weighed against increased product costs.

A further focus of Larses’ analysis lies on maintainability, which is directly related to modularity of architecture. In fact modularity is defined as a top level design goal alongside with cost efficiency. We will discuss evaluation of system architecture with respect to modularity presented therein in section 2.4.2

## 2.4 Architecture Design and Evaluation

Having identified the relevant quality properties of architecture, either by derivation from a generic quality model or from a life-cycle model, architecture candidates need to be evaluated for fulfillment of these criteria. In case of trade-offs the underlying design characteristics must be identified and their impact on quality goals adequately valuated in order to make a decision. The methods of evaluation vary depending on the criterion under evaluation and the information available at the time of evaluation. Most evaluation methods relay to qualitative assessment by domain experts, or to heuristics known to optimize for a certain property, fewer methods apply quantitative analysis based on an architecture model. We discuss a selection of analysis methods applicable to E/E systems subsequently:

Analysis methods can be subdivided into quantitative and qualitative methods. Quantitative Methods involve attribute based, measurement and simulation based methods. Qualitative methods can be further subdivided into scenario based methods, reviews and check-lists.

In the process of quantitative evaluation concrete values of a quality metric are generated. These are based on assumed relationships between architectural properties and elements of the system under evaluation and the achievement of certain quality attributes. The postulated relationships can be derived from expert assessment, empirical data or simulation.

With qualitative methods on the contrary no concrete values for the achievement of a desired property are constructed. Rather result of the evaluation is a set of statements with respect to intended use or expected evolution, or more precisely, whether the architecture under evaluation allows the achievement of an intended level of a certain attribute such as performance, maintainability, correctness etc (Bettencourt da Cruz, 2009).

Qualitative methods have the advantage of easy appraisal. A statement on the suitability of a solution based on a 5-point scale is easy to make, provided sufficient knowledge about the system and its interdependencies. Evaluation methods such as ATAM combine the definition of wishful properties as well as the quality carrying properties and interdependencies in one process thus the discovery of quality requirements goes along with an exploration of possible solutions. Through discussion and analysis of solutions and consequences among interested stakeholders a common understanding of the system is established.

The accuracy of the assessment depends on the experience of the architect and involved stake-holders and their knowledge of the system. This indicates the drawback of qualitative evaluation, primarily the requirement of sufficient domain expertise and secondly the low scalability with respect to the number of decisions or alternatives under evaluation.

### 2.4.1 Selected Scenario Based Methods

**Architecture Trade-off Analysis Method**

The predominant representative of scenario based, qualitative methods is the Architecture Tradeoff Analysis Method (Kazman, et al., 2000). The Architecture Trade-off Analysis Method (ATAM) does not only reveal how well an architecture satisfies particular quality goals, but it also provides insight into how these quality interact with each other – how they trade off against each other (Bahsoon, et al., 2003). The method is largely based on expert assessment of the impact of design elements on quality attributes. It defines a process of six steps which consecutively define requirements, the design space and identify sensitivity and trade-off points.

An application of the ATAM to design decisions in automotive E/E systems is presented in (Fröberg, et al., 2006). The exemplary utility-tree discussed in consists of some 16 scenarios subdivided into safety, reliability, modifiability, and serviceability; the authors claim the utility tree not being representative nor complete. According to the ATAM philosophy the definition of quality goals is in fact part of the development process.

The architectural viewpoints in this case are a simplified graphical notation to depict physical the distribution of the E/E system; the alternatives regard segregation and integration of components and choice of infrastructure, i.e. system structure and communication technology. Evaluation of quality attributes is based on expert assessment, and valued on a three-point scale.

**Cost Benefit Analysis Method**

The Cost Benefit Analysis Method (CBAM) is an architecture-centric method for analyzing the costs, benefits, and schedule implications of architectural decisions. The CBAM builds upon the ATAM to model the costs and benefits of architectural design decisions and to provide means of optimizing such decisions. Conceptually, CBAM continues where the ATAM leaves; it adds a monetary dimension to ATAM as an additional attribute to be tradedoff.

The CBAM consists of the following steps: i) choosing scenarios and architectural strategies (AS); ii) assessing Quality Attribute (QA) benefits; iii) quantifying the architectural strategies; iv) costs and schedule implications; v) calculating desirability; and vi) making decisions.

To quantify the architectural strategies benefits, stakeholders are asked to rank each AS in terms of its contribution to each quality attribute. A scale of –1 to +1 is used. A +1 means that this AS has substantial positive effect on the QA (for example, an AS under consideration might have substantial positive effect on performance) and –1 means the opposite. Each AS can be assigned a computed benefit score from –100 to +100. CBAM doesn't provide a way to determine the cost; it considers that cost determination is a well-established component of software engineering and is outside its scope (Bahsoon, et al., 2003).

**Architecture Options**

With respect to the evolution of systems Architecture Options (Baldwin, et al., 2006), (Engel, et al., 2008) or Real Options (Ozkaya, et al., 2007), (Bahsoon, et al., 2003) can be viewed as an extension to the CBAM, in that they aids to determine the value of possible extension of the system.

An option is the right, but not the obligation, to take an action in the future, typically in the financial market. Real options are those that affect nonfinancial assets. Real option analysis helps to determine the value of the ability to extend a system in future, against other quality attributes such as performance or cost-efficiency.

The components of real options analysis include:

- the decision to be made
- a characterization of the uncertainty
- the decision rule

The decision rule is a simple mathematical expression that indicates when the decision should be made and helps identify the critical parameters that architects need to observe during the decision-making process (Ozkaya, et al., 2007). A number of mathematical models exist reflecting the distinct terms of executing the option, the most common ones are: Black-Scholes-Merton, Binomial Option Valuation and Net Option Value model.

The application of Binomial Option Valuation of Real Options in system development has been discussed in (Gustavsson, et al., 2008). The example considered is the extension of an existing ECU vs. the introduction of a new ECU with the periphery being the critical resource by which extensibility is measured (as opposed to bandwidth).

The calculation shows that it is possible to calculate the cost of flexibility i.e. the option price, however the price of option execution depends on the knowledge about future changes. The more precisely future requirements are defined, the easier the calculation what up-front investment in flexibility is justified. This is obviously the reason, why alternatives promising marginal savings on unit costs are often preferred to ones that promise ‘generally better performance’ upon future changes.

### 2.4.2 Modularity and Cluster Analysis

Whereas directly measurable system properties such as performance or efficiency can be easily related to architectural elements and patterns, quality regarding the impact of architecture changes not further specified is harder to quantify.

Scenarios gathered in the process of an ATAM analysis are one possible solution. If no such scenarios are known, ease of future changes is often equalized with modularity. (Eppinger, et al., 2001) postulate that systems exhibiting modularity and alignment perform better.

Likewise (Baldwin, et al., 2006) indentify identify three kinds of benefits of modular designs, they

- make complexity manageable:
- enable parallel work; and
- accommodate future uncertainty.

According to their work, modularity accommodates uncertainty because the particular elements of a modular design may be changed after the fact and in unforeseen ways as long as the design rules are obeyed. Thus, within a modular architecture, new module designs may be substituted for older ones easily and at low cost. Their method used to determine value of modular designs is Net Option Value of real options making the property comparable with immediate benefits such as resource efficiency.

NOV valuation is based on the value of independent experiments that can be conducted on each module, resulting in the selection of the best-fitting design, hence increasing the total option value of a modular design The NOV model values modules from the perspective of design dependencies and flexibility, but the current formulation provides no insights about how quality attributes other than modifiability may affect the valuation (Ozkaya, et al., 2007).

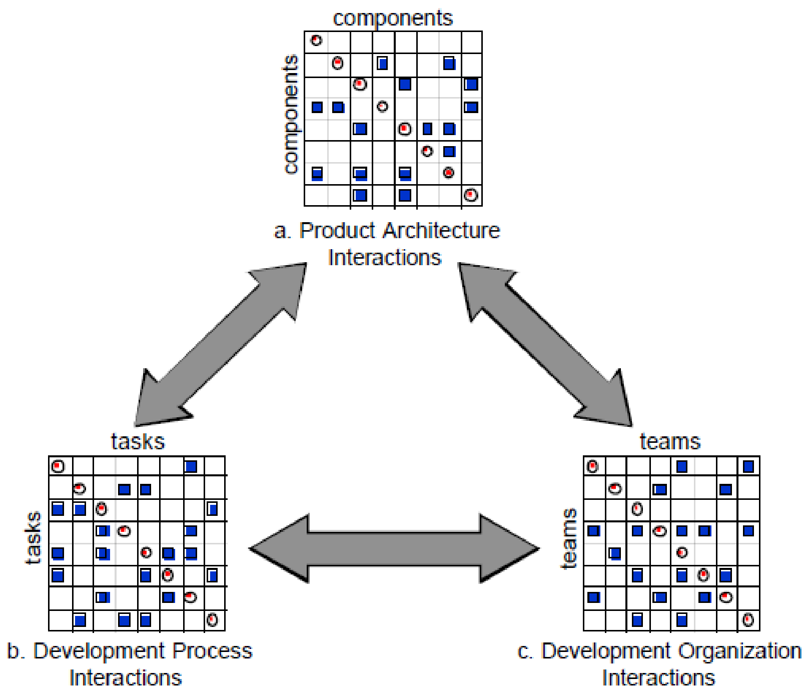

components
components
a. Product Architecture Interactions
tasks
tasks
b. Development Process Interactions
teams
teams
c. Development Organization Interactions

Figure 11 Alignment of product, process and organization structure (Eppinger, et al., 2001)

Following the same argumentation, Larses attributes high modularity to the ease of reuse and evolution in electronic control systems in the automotive domain (Larses, 2005) promoting modularity to a first class design goal (see Figure 1). The method used to analyze architectural designs for modularity and alignment with development and organizational structure as proposed by (Eppinger, et al., 2001):

For a given functional architecture, represented by a graph (similar to the logical architecture described in chapter 4.1 Architecture Model for E/E Systems), and an individual mapping of logical functions onto physical components, organizational components and development activities, the modularity of architecture can be determined as the ratio relationships within a physical, organizational or process entity and the overall relationships of the functions contained.

Modularity and alignment of organization, process and product are certainly important structural properties. However declaring modularity as a first level design goals bears the risk that quantifiable goals such as resource efficiency will be preferred in the case conflict. A structured analysis of costs, risks and opportunities in the life-cycle of automotive E/E systems should improve the weight of arguments for modular designs.

### 2.4.3 Design Patterns and Heuristics

Besides methods to evaluate architectural design towards a specific quality goal, a number of heuristics and design patterns exists which are known to improve certain qualities. (Thiel, 2005) concludes that quality properties must be accounted for in the design process, as architectural drivers. Consequently his work deals with derivation of architectural decisions from architectural drivers. The proposed method to do so is via Architectural Viewtypes, corresponding to the proposed usage of view*points* in an ATAM analysis. A matching of quality goals to view-types defined by Thiel is listed in Table 2.

| View Type | Concerns | Quality Attributes addressed |
|---|---|---|
| **Decomposition** | Resource allocation; encapsulation; configuration control | Modifiability; configurability |
| **Users** | Dependencies among components; separation in processes/tasks | Modifiability; performance |
| **Layered** | Encapsulation; virtual machines | Modifiability; portability |
| **Class** | Instances of components; shared methods | Modifiability; performance |
| **Client-Server** | Distributed operations; separation of concerns; performance analysis; load balancing | Performance; modifiability |
| **Concurrency** | Thread analysis, resource contention analysis | Performance; safety; reliability |
| **Shared Data** | Data producer/consumer analysis | Performance; modifiability |
| **Deployment** | Allocation of software to hardware nodes; safety/security analysis | Performance; availability; safety; security |
| **Implementation** | Configuration control; integration; test activities | Modifiability; integrability; configurability |
| **Work Assignment** | Assignment of software components to development team; best use of expertise; management of commonality | N/A |

Table 2 Architectural viewpoints and quality attributes addressed (Thiel, 2005)

We generally support the approach of identifying architectural drivers and requirements to be met by architectural means. The work is similar to ours in the specification of desired qualities and architectural patterns to support them.

Despite similar goals, Thiel's work differs from the present in several ways. Thiel postulates the elicitation of requirements as part of the design process, structuring the result of the elicitation by ISO 9126 attributes to make them fit his proposed architectural patterns. The premise of our work is that in-vehicle systems exhibit specific quality requirements which can be derived from the life-cycle and valuated as cost and risks of the developed product.

Secondly the architectural means in the automotive domain are in fact limited to hierarchical structuring of the physical architecture into systems and components, as well as partitioning of logical functions onto components. As the quality requirements of E/E systems have not yet been comprehensively defined, the impact of structural decisions beyond design patterns needs to be assessed.

Finally it is the goal of this work to investigate the relationships between architectural structure and quality goals, rather than stating that such relationships can be discovered by using a certain notation of viewpoint.

In a research work conducted at BMW (Felskau, 2006), the advantages and disadvantages of design strategies for automotive E/E system architectures have been investigated. (Felskau, 2006) defines the following evaluation criteria:

- Modifiability
- Extensibility
- Scalability
- Complexity
- Resource preservation
- Reliability

The work analyzes a number of architectural strategies with respect to criteria listed above. The strategies and their characteristics are listed in Table 3.

| Optimization Goal | Design Decisions | Consequences |
|---|---|---|
| Production Efficiency | • Low level of abstraction in application code<br>• Low level of encapsulation of hardware<br>• Simplified diagnosis (and other system services)[3]<br>• Low degree of standardization<br>• Non-deterministic access to communication media<br>• Offline resource management | Advantages<br>• Low production costs<br><br>Disadvantages<br>• High development cost<br>• Low modifiability<br>• Low extensibility<br>• Low reusability |
| Scalability | • Star topology<br>• Application level abstraction<br>• Deterministic scheduling<br>• Sophisticated diagnosability | Advantages<br>• Extensibility<br>• Modifiability<br>• Reusability<br><br>Disadvantages<br>• high development costs |

[3] "simplified" meaning basic in its capabilities, resulting in increased effort of service operations

| | | |
|---|---|---|
| | | • high production costs |
| Openness | • Tracing of component behavior<br>• Dynamic resource management<br>• High degree of standardization<br>• High modularity (integrability)<br>• Mixed access to communication media | Advantages:<br>• Modifiability<br>• Extensibility<br><br>Disadvantages:<br>• High resource usage<br>• High production costs<br>• High development costs |
| Centralization | • Low number of components compared with services provided<br>• Deterministic scheduling | Advantages:<br>• Low production costs<br>• Low infrastructure utilization<br><br>Disadvantages:<br>• Thermal compatibility<br>• Difficult packaging |
| Decoupling of Hard and Software | • Abstraction of communication infrastructure<br>• Abstraction of component hardware | Advantages:<br>• Modifiability<br>• Low development costs<br><br>Disadvantages:<br>• High resource usage<br>• High production costs |

Table 3 Architectural design strategies advantages and disadvantages (Felskau, 2006)

This work finally distinguishes the goals of optimization and the means to achieve these goals. With the means identified, the advantages and disadvantages read: trade-off points are qualitatively assessed.

A qualitative discussion of the advantages and disadvantages of design strategies is certainly helpful to promote the importance of strategic architecture development and management. Likewise any evidence of the impact of architectural decisions on quality is most welcome. However the methodology of this work is not transparent. Particularly the attribution of design characteristics to the design strategies appears arbitrary. Nevertheless, the work at least in its fundamentals relates structural properties to quality attributes.

Contrary to (Felskau, 2006) the premise of our work is that quality goal conflicts are not inherent to the quality model, but arise from the structure of the developed product. In other words, quality trade-offs are dictated by the model of architecture and the degrees of freedom it allows. For instance, having only a physical component (ECU) as a unit of specification and development, the goal of production efficiency is in conflict either with scalability, since the former benefits from increasing functional density, while the latter is being decreased by lowering the granularity of system adaptation, or it conflicts with integration effort and correctness, since component variability increases the number of vehicle configurations to be tested.

## 2.5 Summary

We have discussed the concept of quality and various approaches to the definition and management of quality particularly in automotive E/E systems. We have concluded that quality is indeed a subjective and transcendental goal. It is highly uncertain, because the stake-holders expectations are hard to predict. The notion of quality however can be approximated by specifying measurable criteria to guide the design and evaluation process.

We have reviewed related work in the field of architecture evaluation, particularly work related to the evaluation of automotive E/E systems. Architecture development is both an ongoing process of adaptation and optimization of an existing system as well as a revolutionary process of redesign to eliminate suboptimal design decisions introduced so far (Axelsson 2007). Engineers deal either with short-term decisions in which sub-optimal solutions are tolerated, as they are expected to be revised with the next generation change, or with long term decisions reflecting the current state of requirements but not the expected changes of their lifetime. Both of these design and decision processes involve an evaluation of architecture alternatives, though not necessarily by the same criteria.

The need for a structured and systematic approach to architecture evaluation and development has been recognized by (Broy, et al., 2006) and (Wallin, et al., 2008). However a comprehensive quality model tailored to the challenges of automotive development is yet to be defined. The one cited in (Fröberg, 2007) and depicted in Figure 9 seems promising, however lacks an operational definition of criteria mentioned therein, let alone a discussion of potential conflicts and trade-offs. Other publications, e.g. (Thiel, 2005), (Larses, 2005) postulate a set of quality criteria to optimize for but do not discuss how these criteria have been derived or propose a scenario based definition by domain experts omitting the discussion of evaluation criteria altogether.

Regarding methodology, research work on of quantitative methods of evaluation for automotive E/E systems so far focuses on modularity analysis and design options. Work on qualitative evaluation recommends either Analytical Hierarchy Process for comparison or architecture alternatives which once again leads to expert assessment. In the field of systems engineering and software architecture, ATAM is widely recognized and has been applied to automotive E/E systems (Fröberg, et al., 2006). Its downside is its dependency on expert assessment and low scalability with respect to the number of decisions to be evaluated.

Quantitative, model-based methods can complement scenario based methods especially in evolutionary changes, where gradual worsening of overall quality easily slips out of scope. However their accuracy critically depends on the quality of models captured up front. The inherent risk is that the quality of results does not justify the extent of modeling required.

We see the lack of a comprehensive model of quality goals reflecting the requirements and constraints of in-vehicle systems and indicating internal and external factors of quality is probably the biggest research gap. This work intends to close this gap and combine the quality model with existing evaluation methods. To support a comprehensive analysis, evaluation should be conducted based on a common architecture model and with respect to a quality model properly addressing the cost and value structures in the automotive domain.

This work intends to address the following issues in the area of quality assessment of E/E architecture:

- There is a need for systematic architecture evaluation in the domain of automotive E/E systems.
- Generic quality models, such as ISO 9126 do not describe the problem properly, a quality model reflecting the specific requirements and life-cycle of automotive E/E systems needs to be defined.
- Existing work on architecture analysis lacks a concise argumentative chain between the goal and the architectural means to achieve them.
- To integrate the various methodologies regarding individual quality goals, a discussion of quality properties based on a common vocabulary for architectural means is required.
- To properly weight individual quality goals, an analysis of their respective economic relevance is necessary

# 3 Case Study on Architecture Evaluation

In the previous chapter we have concluded that a quality model for E/E system architecture needs to account for specific requirements, constraints and processes of development and production of in-vehicle E/E systems. Secondly, that the quality goals defined need to be related to architectural means in order to identify conflicts and trade-offs.

The following chapter presents the decision process in architecture development at a premium automotive OEM and describes typical architecture changes as well as the evaluation criteria applied. Finally it presents the findings of a number of documented architectural decisions. Eventually we suggest improvements in the list and structure of the evaluation model, as well as methodology of their appraisal.

The goals of this investigation is to gain an understanding of the architectural means available to the system architect to optimize for certain criteria and the goals which need to be addressed during architecture development as well as their economic relevance and inherent priorities to justify trade-offs. The findings of this analysis will help us to guide the constructive part of this work.

## 3.1 Context of study

Any automotive OEM and particularly manufacturers claiming technological leadership, is faced with the challenges of optimization of architecture to meet profitability goals on the one hand and evolution of architecture in pace with market needs and customer requirements on the other hand. The need to introduce changes to architecture and assess architectural decisions with respect to their consequences arises naturally out of these two objectives.

At the same time the development and decision process at a large OEM is scattered across a number organizational units responsible for product lines, functional domains, components and functions. The problem that each organizational entity optimizes decisions towards its own goals leading to local optima but neglecting synergies at a global level has been identified and led to the concentration of responsibility for architectural decisions at a dedicated systems architecture department.

Standard quality models such as the ISO/IEC 9126 (and its successor ISO/IEC 25000) do not adequately represent the requirements and constraints in the life-cycle of automotive E/E systems, as outlined in section 1. As a consequence proprietary sets of criteria for architecture assessment tailored to the automotive domain have evolved based on the experience of engineers responsible for E/E systems development and lessons learned from previous architecture measures and often from failures.

This pragmatically evolved form of architecture evaluation reflects the experience gained with E/E systems through years of architecture development. A number of design decisions which have been appraised for their consequences on the E/E architecture are documented and pose a viable source of knowledge about the economical dimensions of and the sensitivity of quality goals to structural changes. Hence this is the point to start when trying to holistically define E/E system quality and relate quality goals to structural properties of architecture.

## 3.2 Evaluation Scenarios

Decisions regarding the E/E system architecture occur either during the design and concept phase of product line development or during the maintenance phase and apply either to individual product lines or the architectural blueprint defining guidelines and common aspects of all product lines.

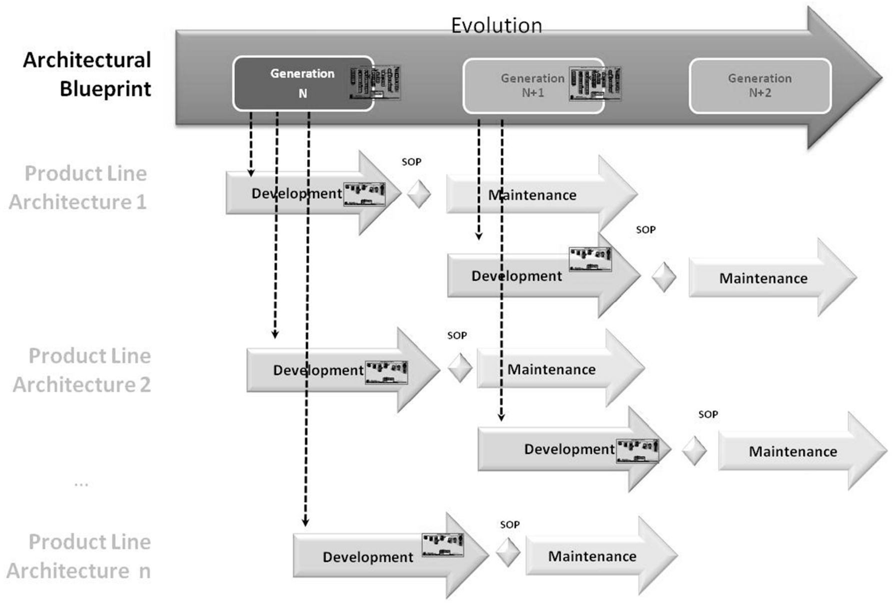


Figure 12 Architecture decisions in the life-cycle of E/E systems

Change requests in the *development phase* regarding the next E/E architecture generation are more fundamental, affecting the general system structure (definition of systems), initial definition of components including partitioning of functions and communality i.e. the re-use of existing components vs. development of product-line specific variants. The number of vehicles and time-span for consideration of expected functional changes is naturally larger than that affected by maintenance activities. Depending on product line the number of units affected lies in the order of 100 000 vehicles per year, throughout a time span of six years of production and additional 15 years of support after end of production.

Change requests in the maintenance phase include introduction of new services to an existing system and replacement of existing components by components developed in parallel product lines. The impact of changes is lower, and the number of units affected decreases with each year of production, which makes fundamental changes harder to justify economically.

Change requests in general can be subdivided by the cause of the change request: structural changes to reduce costs or system complexity and functional changes to implement a new customer function increasing vehicle value. Structural changes in turn can be categorized by the part of the architecture affected, either as a change to the partitioning of functions, components or physical structure, possibly accompanied by elimination of systems or components or conversely introducing a variant in an architectural element to exploit differences in requirements for distinct configurations.

Secondly, change requests can be grouped by the scope of the change, i.e. the number of vehicles and phases affected and the variability of requirements to be considered. In this respect, either the change is being applied to a product line or represents a global architecture guideline for all product lines, thus affecting the architectural blueprint.

The conditions for evaluation of architecture alternatives differ, depending on the kind of architectural change. The decision on structural changes depends on the business-case for the change, i.e. savings in production need to compensate investments in development and integration, meeting a fixed rate of return. This means that suboptimal architecture may pertain, simply because the change towards a more efficient architecture does not meet profitability goals.

The business case for functional upgrades is easier to construct, as the increased value of the vehicle justifies additional costs both in development and production. Depending on whether the additional feature should be designed as standard or optional equipment, additional revenues or product value arguments compensate increased costs. In fact preference can be given not to the least expensive alternative, but the one with the highest communality or architectural conciseness.

Typical architectural means, either in order to add new functionality or to optimize an existing design for a certain type of costs are:

- Integration of components
- Integration of systems
- Migration of components between systems (Component Re-Partitioning)
- Migration of functions between components (Function Re-Partitioning)
- Introduction of infrastructure variants
- Introduction of component variants

The presented categorization is not fully disjoint and the mentioned use-cases are correlated: for instance integration of two or more physical components into one naturally implies a change in the system infrastructure and potentially additional component variants. Likewise the integration of components is a special case of re-partitioning of functionality, namely partitioning to one or more remaining components. Finally the integration of communication buses is a special case of re-partitioning of components between systems. Nevertheless a categorization of structural changes is useful to identify the quality properties of the E/E system which will be affected by the change.

## 3.3 Evaluation Process

The evaluation of decisions and design alternatives regarding the E/E architecture is guided by a defined change-request process similar to the one described by (Gustavsson, et al., 2008) and depicted in Figure 13. The alternatives, progress of evaluation, relevant information and results are typically documented in an issue tracking system.

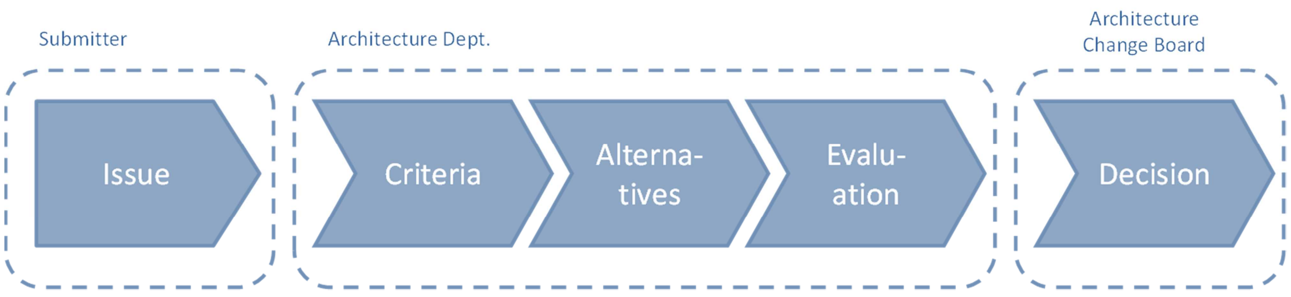


Figure 13 Architecture change and decision process (Gustavsson, et al., 2008) annotated with roles

Suggestions for an architecture change come from various sources within the organization, typically the centers of competence responsible for individual functions, marketing, product owners but also from the architecture department itself based on global observations of existing inefficiencies and possible synergies.

With the proposal of a change to existing E/E architectures, either in operation or under development, an architecture ticket is created. The creation of a ticket involves specifying a reason for the change, expected costs, benefits and affected components, as well as complexity and risk. Reasons for architectural changes can be one of: cost-down, functional change, optimization or reduction of system complexity or bug-fix. Regarding the assessment of existing architecture, cost down and reduction of complexity are most important. The content of one or more of these fields is the subject of the ticket, such as requests to evaluate the economic potential of an architecture change. In that case the fields are being filled out by the architecture engineer after assignment.

After creation, the architecture ticket is assigned to an engineer within the E/E architecture department. The task of the engineer is to collect relevant information from various sources, including departments affected by that decision, potential suppliers and experts, compile gathered data for decision in the E/E architecture board, and make sure all criteria relevant to the decision have been considered.

The evaluation of a proposed change is separated into technical and economic aspects, evaluated independently. The decision whether to implement the proposed change is made by a change board based on the economic and technical recommendations.

## 3.4 Evaluation Criteria

Architectural decisions need to be assessed according to a set of objective criteria reflecting all economically relevant aspects of E/E systems. This is to prevent the aforementioned explosion of complexity for the sake of lowering production costs and to avoid long-term consequences and lock-in situations, due to continuous decrease in architectural consistency in small steps.

The definition of such criteria is an ongoing task, as of the time of writing, three such lists were known in the company in addition to a number of projects to assess architecture in the architecture departments. The reason for constant changes in the catalogue of criteria is certainly the process of discovery but equally current trends such as $CO_2$ emissions or construction kit strategies. In addition problems arising in current vehicle development projects result in new criteria being added to the list, to prevent such problems from re-occurring. The most up-to-date and referred to set of criteria by the time of writing is presented in Figure 14.

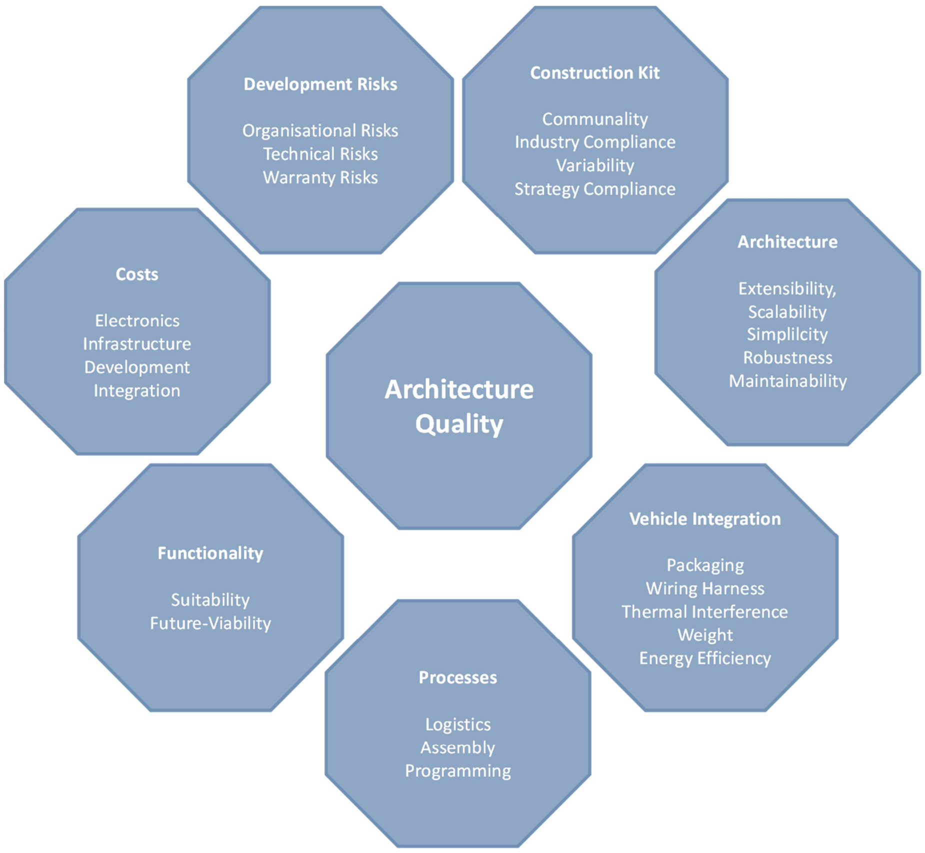


Figure 14 Evaluation criteria for E/E architecture decisions

It should be noted, that this list contains criteria regarding the change as a project and criteria being subject of the change, i.e. the aspects of the architecture's business value to be increased. Some criteria, such as impact on warranty costs, obviously apply to both, the change project itself and the outcome of the project, in that case any subsequent change. Since such subtle differences are not documented in the model itself, the appraisal is typically biased towards one or another interpretation.

The model defines seven categories and roughly 30 criteria. The categories are briefly discussed below. A more detailed discussion including their relevance for the architecture decision is given in section 3.5

**Functionality** – Functionality covers the extent to which the functional requirements are met or will be met by the architecture under investigation and the expected ability to implement future changes. This category is subdivided into *suitability* assessing current requirements and *future viability* regarding expected evolution of the system.

The first criterion can be assessed on an ok / not ok scale. Obviously a *nok* in that respect eliminates the candidate architecture and makes further assessment obsolete. A typical example is the assessment of architecture cases involving elimination of resources in order to reduce productions costs. Likewise lack of bandwidth or computational resources prevents certain functional allocation of additional functions.

The ability to realize future requirements is obviously hard to determine without making assumptions on such requirements. Without further specification on the kind of changes, the evaluation is reduced to the assessment of excess resources in the system. For instance the load on communication buses is not to exceed a certain threshold at start of production as changes throughout the life-time of a product line are known to consume preserved bandwidth.

**Costs** – the category 'Costs' covers the immediately measurable financial consequences of the change request. This includes vehicle-production costs of electronics and infrastructure, development and integration costs and to some extent costs of assembly.

The term *costs* may create the impression that the remaining evaluation criteria do not have economic impact. Obviously this isn't the case. The rationale behind this grouping is that, contrary to the remaining criteria, costs of change requests can be directly estimated by domain experts – either potential suppliers or engineers responsible for development and integration – and are directly represented in the change request's business case.

Costs as a category cover both costs related to the change project as well as the business value of architecture, i.e. the costs in production. The result of the cost analysis is a net present value of the change and constitutes the economic recommendation of the architecture evaluation.

**Development Risks** – Development risks comprise possible reasons for project failure. Development risks are subdivided into organizational and technical risks. Examples of organizational risks are unclear distribution of responsibility with the organization or lack of know-how and or alternatives among potential suppliers (vendor lock-in). Organizational risks are appraised qualitatively. Technical risks on the other hand are not further specified and none of the documented change requests refers to technical risk as a factor in the decision process. Probably, since the term is redundant to the more specific electromagnetic compatibility and thermal compatibility, but also as immature technologies are not considered for in-depth assessment. In some cases warranty risks are counted towards development risks and assessed quantitatively.

**Architectural Consistency**[4] – In the original evaluation model this category is denoted as "Architecture",

[4] Consistency in this context means coherent with guiding principles and exhibiting correspondence among related aspects which is admittedly underspecified but still better than subsuming the related criteria as

which is misleading, as all of the remaining evaluation criteria equally apply to architecture. The category spans several criteria unrelated to other categories. Concrete quality attributes include scalability and extensibility, robustness, complexity and maintainability.

The appraisal is qualitative. In concrete cases architecture related criteria measure the simplicity and consistency of architecture, such as separation of concerns, distribution of functionality etc. If existent the appraisal is performed by the architect responsible for the change, based on a qualitative -- through ++ scale.

**Construction Kit Compliance** – Construction kit compliance relates to compatibility and reuse of components and solutions. Defined criteria are forward- and backward compatibility – the ability to adapt the change in parallel product line architectures and compliance with industry or de-facto industry standards. In addition variability refers to the number of variants in the product line architecture, either as ECU or infrastructure variants.

The appraisal is qualitative and based on expert assessment. Compatibility with parallel product line architectures is determined informally through inspection of physical and logical interfaces of a component.

**Vehicle Integration** – vehicle integration criteria cover issues related to installation within a vehicle and interference with the rest of the vehicle. Installation within the vehicle applies to the packaging of components and the wiring harness in the available installation spaces, interference relates to thermal and electromagnetic compatibility.

Packaging criteria are evaluated by domain experts on an ok / not ok basis. Likewise thermal and electromagnetic interference are assessed in that form. Strictly speaking thermal and electromagnetic compatibility are risks, which render an alternative infeasible if exceeding a certain threshold. This threshold however isn’t defined and the appraisal is based on an ok/nok basis by domain experts.

**Processes** – the category reflects the impact on production and production related activities at the vehicle level. Criteria include assembly, logistics, and programming. In cases in which these criteria have been applied, the number of components, variants and partitioning of components to systems was evaluated and rules of thumb employed to determine the economic impact of derived metrics. The outcomes of the appraisal such as additional assembly time, impact on vehicle programming time as been included in the business case for the architecture change.

---

“architecture”.

## 3.5 Analysis of Documented Architecture Changes

The subsequent analysis is based on the evaluation of roughly 30 architecture change requests with documented assessment of their consequences on quality properties. In each case the criteria used for evaluation were very loosely based on the criteria listed in section 3.4 however the criteria have been applied selectively and no architecture change has been assessed for all listed criteria.

One third of the documented change requests were requests for functional changes, i.e. implementation of new requirements. The remaining cases were structural changes aiming at reduction of costs or system complexity.

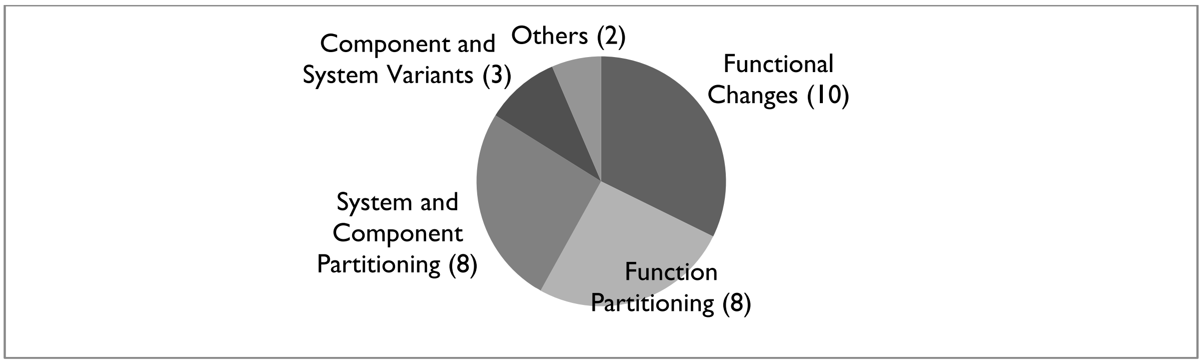


Figure 15 Distribution of analyzed architecture changes

Out of the structural changes, change requests affecting the number of components i.e. integration or segregation of components as well as requests for re-structuring systems i.e. assignment of components to communication buses constitute the largest part. Change requests to introduce new variants in infrastructure or components make up roughly ten percent. Other changes include removal of redundant sensors or changes to the physical wiring (transparently to the logical communication infrastructure).

An analysis of documented architecture changes revealed that the aforementioned evaluation criteria are neither being applied consequently nor in a unified way:

- In most cases just a few criteria are being applied. This indicates either inherent priorities of criteria leading to early rejection of alternatives, if high priority criteria are not satisfied, or a lack of standardized methodology of assessment. Another possible reason is that the omitted criteria may not be sensitive to the metrics affected by the respective change request.
- Secondly the terminology to discuss the costs and benefits of architecture alternatives is not uniform. Particularly soft criteria such as future viability, scalability, construction kit compliance are used in different contexts with various meanings and appraised differently.
- Finally in a number of cases metrics such as requirements of computational resources are being collected without an interpretation of the results. In the case of required resources, the metric could be attributed either to suitability (i.e. fulfillment of requirements) or to future viability, measuring resources remaining for future functional upgrades.

Figure 16 presents the distribution of criteria considered when evaluating decisions on change requests. Costs for instance have been considered in almost 90% of structural changes but only in 50% of functional upgrades. This means, the business case for a functionally transparent change to an existing architecture has been calculated in 90% of all change requests. If a new function was introduced, which typically increased vehicle value, only in 50% of all cases the costs of development and production have been evaluated prior to deciding whether to implement that change.

Process issues on the other hand have only been considered in 20% of the structural changes and in none of the documented functional upgrades.

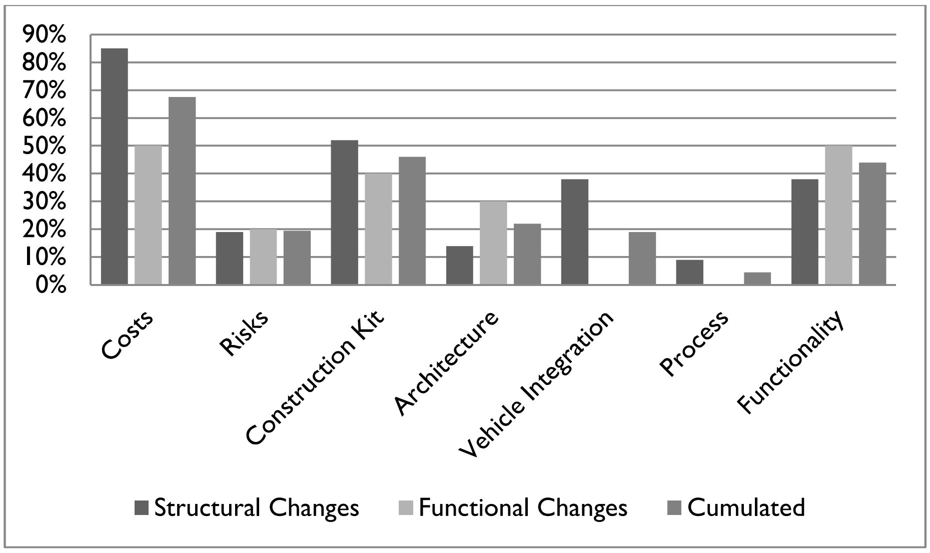


Figure 16 Distribution of evaluated criteria in documented architecture changes

Differences between structural and functional changes are not significant. Merely cost considerations and aspects of vehicle integration have been applied slightly more frequently in structural change requests.

The data presented in Figure 16 allows only general conclusions and need to be interpreted with caution. The presented categories of evaluation criteria are quite inhomogeneous. Costs, represented as a business case of an architecture change for instance contains both development effort and production costs and allow limited conclusions about the business value of the architecture itself. For instance development effort is dominated by the functional size of the newly developed function rather than by architectural structure.

The second most commonly documented category turns out to be construction kit compliance. Though undisputedly important, the category is also the most scattered comprising at least five distinct, partially overlapping criteria. This issue is discussed in more detail in section 3.5.5

Other criteria, while being evaluated less frequently, if present (and documented) outweigh other more predominant considerations particularly cost efficiency. This applies to risk and, unsurprisingly, the fulfillment of required functionality. Likewise violation of construction kit strategy or guidelines leads to exclusion of otherwise beneficial architecture alternatives in almost all cases. Vehicle integration and ‘architecture strategy' issues, although evaluated in a number of cases have not led to rejection or selection of any change request. Process efficiency is typically being evaluated as assembly costs and reflected in the business case.

The statistics presented count the occurrence of at least one criterion of each category in the evaluation, which is not very informative particularly with conglomerate categories such as "architecture strategy and consistency". In fact some categories are represented by another criterion in each case. The following sections discuss the categories in more detail providing statistics for individual criteria within categories.

### 3.5.1 Functionality

Functionality covers two seemingly unrelated criteria, namely fulfillment of requirements and future viability, i.e. the ability to fulfill future requirements.  Both criteria are elicited through comparison of required and available resources within the system. Resource consumption has been elicited for about 40% of the structural and two thirds of the functional changes.

As depicted in Figure 17 the fulfillment of known requirements by the candidate architecture is being evaluated in slightly less than half of all architecture change requests. With structural changes such as migration of components or integration of communication buses available resources are appraised in only about 30% of all decisions. Resources available for future requirements are considered even less frequently. With new functions to be implemented, available resources are being appraised slightly more often. Likewise the preservation of resources for future use has been appraised in almost half of the documented decisions. In all other cases, the architects assumed the existence of sufficient resources on communication buses and control units.

The appraisal of resources required has at least two implications: the requirement and availability of elastic and inelastic resources. The requirements for inelastic resources determine the feasibility of architecture alternatives, meaning the ability of an existing architecture to provide an additional function. The predominant example of inelastic resources is bandwidth on main communication buses, which renders architecture changes such as integration of communication buses, re-partitioning of components and functional evolution impossible, if raised beyond capacity.

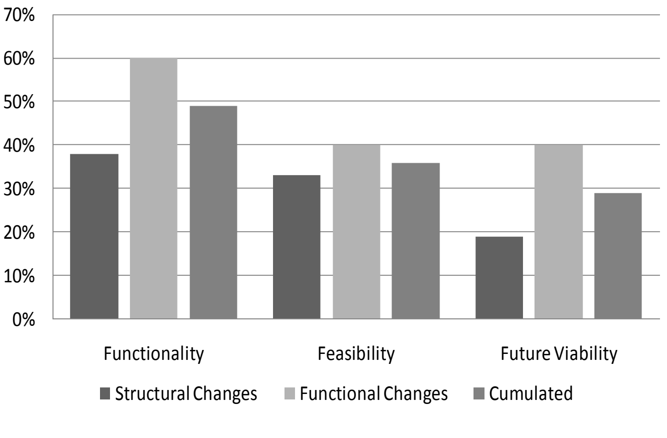


Figure 17 Frequency of appraisal of functionality issues in architecture changes

Requirement for elastic resources, such as processor capacity, memory and storage is less critical. Resources being added to the system result merely in increased production costs hence negatively affecting the business case for an architecture change. Requirements for inelastic resources however pose a knock-out criterion for a suggested change if not present.

The feasibility of an architecture alternative is in theory a fundamental requirement for the consideration of alternative, meaning that alternatives not fulfilling the functional requirements do not need to be evaluated at all. However the availability of required communication or computation resources is not always clear to the initiator of a change at the time a change request is submitted.  Thus the ability of a proposed change to provide the specified functionality is in practice a criterion for evaluation – for the sake clarity we denote that property as feasibility (in line with the feasibility criterion in optimization problems).

The complementary aspect of resource consumption is future viability, expressing the ability of the E/E system to incorporate future requirements, possibly unknown at the time of evaluation. The future viability of architecture is appraised through assessment of excess resources, i.e. unused resources within the systems. The resources considered for that criterion are bandwidth – as a typical example of an inelastic resource. Less commonly processor utilization but also connector pins on central ECUs are considered for future viability.

In only one case out of the roughly 30 analyzed architecture change requests the ability to implement future changes has been evaluated by assessing the distribution or functional logic among ECUs. In that case a centralization of functionality in one ECU has been considered beneficial. The reason for such an underrepresentation of functional aspects can be attributed to the high uncertainty of future modifications. Increased effort to modify or extend a given architecture, resulting from a concrete architecture decision, results in hypothetical increase of costs for future changes, however it does but does not prevent a future change altogether.

### 3.5.2 Costs

Costs are the single most important criterion for evaluation of structural changes. Since lowering production costs is the primary goal of structural changes, architectural measures not yielding a sufficient return are not being implemented. Typically costs of electronics outweigh the costs of infrastructure and assembly (detailed examples are provided in chapter 6).

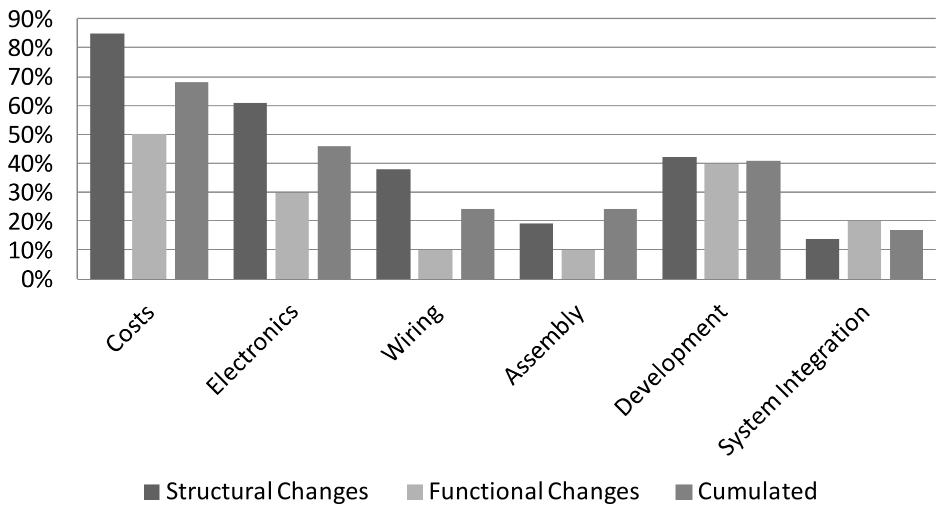


Figure 18 Frequency of appraisal of economic issues in architecture changes

Figure 18 presents the frequency of appraisal various criteria in the evaluation of architecture decisions. Some sort of costs has been considered in almost all documented decisions on structural changes and half of the functional decisions. Out of specific costs, electronics and development have been appraised in more than 40% of all cases, with costs of electronic components being less dominant for all changes leading to increased functionality. In general the frequency of appraisal represents the share of the individual cost types in the total business case in which costs of electronics and development dominate the other cost categories.

Strictly speaking, development and system integration costs are not measuring architectural quality, as the costs of a concrete change are dominated by its functional size and modifiability of the current architecture, which is a result of previous changes. The result of the current architecture decision in turn affects the costs of potential, future changes. The profitability of a change is determined by the up-front investment in both development and integration and thus is an important factor for the decision on a change. This indicates a methodical gap, that ability to implement future changes is rarely appraised, despite the fact that costs of changes which obviously depend on the architecture's modifiability determine the decision on any current change.

The means to assess the profitability of architectural changes is the architectural business case. Investments in development, integration and tooling are evaluated against savings in the costs of electronics and infrastructure, as well as costs of assembly. Out of the analyzed structural change requests for 78% a business case of some sort has been calculated. In the remaining change requests, costs have not been evaluated explicitly, as the architecture change was necessitated by a priori decision or the number of possible solution alternatives was limited otherwise. Unlike structural changes, only half of the functional changes have been assessed for profitability. The low rate of documented business cases can be partially explained by the fact that investments and production costs are compensated by increased vehicle value, thus architecture evaluation in these cases concentrates on technical feasibility.

### 3.5.3 Risks

Risks express the probability and consequences of missing project goals. In the set of criteria for architectural decisions project goals are not precisely defined, but according to architects responsible for their evaluation, serious delays to the development timeline, i.e. inability to implement a function until start of production, unexpected one-time costs due to additional verification cycles or development of fall-back solutions as well as increased number of errors detected after start of production are considered risks in that respect. Project goals can thus be defined as time, cost and quality of the vehicle's E/E system in general or selected customer perceived functions.

Besides the capitalized value of a change resulting from a business case, risk is an equally important criterion for the evaluation of architecture changes. Even though project risk has been assessed and documented only in two of the analyzed change requests, representing less than 10% of the investigated changes, each architecture change exhibiting increased risk has been rejected despite positive business case. Including the possible effect on warranty, some sort of risk assessment has been conducted only in 20% of the documented cases.

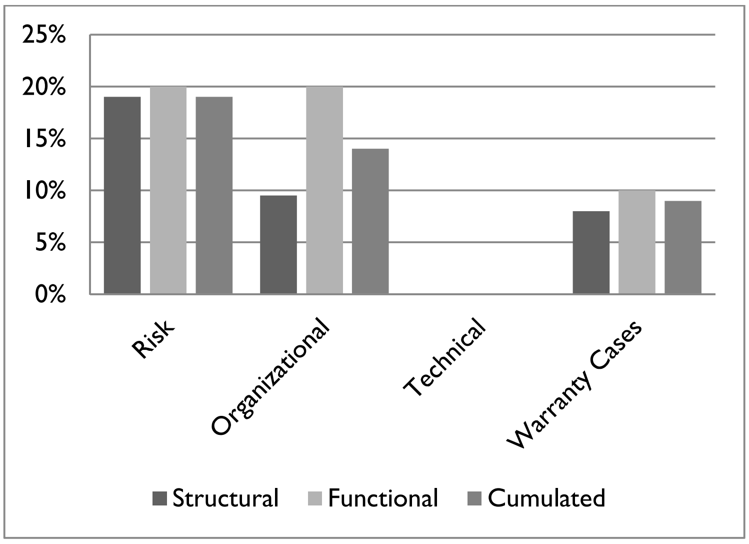


Figure 19 Frequency of appraisal of risks in architecture changes

In the documented architecture changes, risks have been appraised purely qualitatively. Lack of expertise by the supplier or experience with the function to be developed within the department in charge of it, as well the risk of vendor lock-in are documented reasons for increased proposed architecture measure.

Besides the risk of project failure, the increased warranty costs are being considered as a factor when deciding on an architecture change. The expected number of errors has been estimated based on known error rates and a 'surcharge' based on the complexity of the component. Its value has been appraised based on assumed constant costs per error discovered after start of production.

Even though *technical* risks are listed as a potential criterion, none has been considered in the documented change requests. Specific technical risks such as electromagnetic compatibility and thermal compatibility are evaluated separately as vehicle integration issues or not at all. Other technical issues potentially posing a project risk are not documented. Technical risks such as immature technology are not mentioned, probably as such technologies have not been seriously considered in as an alternative.

### 3.5.4 Architectural Consistency

As mentioned previously ‘architecture consistency’ is a heterogeneous category containing a number of distinct criteria, most of which not being appraised in the process of architecture evaluation at all. Simply stated architecture consistency intuitively refers to the coherence with overall guiding principles, correspondence of related aspects and complexity of the system. System complexity is a soft criterion applied predominantly to decide on change requests with similar economic characteristics. Reduction of system complexity however is the motivation for a number of architecture change requests particularly regarding reduction of subsystems.

As depicted in Figure 20 consistency is being used in at least two contexts: less frequently, in about 10% of the analyzed cases as the number and heterogeneity of systems, more frequently, in roughly a quarter of the considered changes as the number of functional dependencies between components. The first metric is used in connection with measures aiming at complexity reduction i.e. integration of systems or re-partitioning of components. However it is only sensitive to architecture changes targeting at the system structure, which constitutes only 25% percent of all changes presented. The latter is used to measure the impact on development related properties by the respective change. This metric is sensitive to function partitioning and component integration changes, which together constitute more than 50% of all changes.

The reason for appraisal of system complexity is the expected (future) specification and development effort and the risk of errors being introduced with subsequent changes. The examined architecture changes do not contain any monetary quantification of system complexity, thus a trade-off between lowered complexity and at the cost of increased production costs is not documented. Increases of system complexity due to short-term cost savings measures are typically prevented through architecture guidelines defined in the architecture strategy or construction kit.

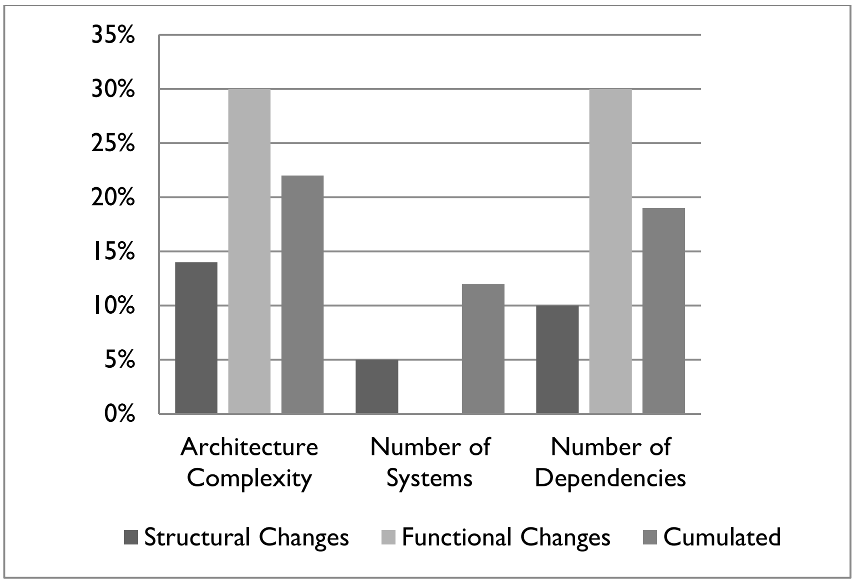


Figure 20 Frequency of appraisal of architecture complexity in architecture changes

The reduction of system complexity is a common motivation for structural changes. Most changes aim at reducing the number of systems or partition of components in order to reduce system complexity. The other aspect of system complexity in connection with architecture evaluation is functional interconnection as an (inherent) metric for the understandability of a solution. In this respect rather than the number of systems, components or variants, the extent of functional dependencies and distribution of functional logic is being assessed.

Other criteria attributed to architecture, but not documented to be actually assessed, are: modifiability, extensibility, scalability and robustness.

### 3.5.5 Construction Kit Compliance

Construction kit compliance is an often recurring subject in the evaluation of architecture changes; however it is also the most polymorphic. Construction kit considerations occur in about 50% of the documented change requests and if accounted for in sum, are the second most important category next to costs. If considered separately, construction kit compliance splits into at least five different criteria: variability of infrastructure and components, communality, conformance to construction kit strategy and industry compliance.

**Component and Wiring Variants** – The variability of requirements within a single product line is considerable. The derivatives of a product line split up into five engine variants, four body variants as well as right-hand and left-hand drive. This alone sums up to 40 possible vehicle models with more than 30 options of additional equipment to choose from (BMW Group, 2011).

Variability of components and infrastructure at the product line level is an architectural mean to adapt the vehicle E/E system to the customer specific requirement set in order to reduce the average cost and weight of individual configurations. Variability of components and infrastructure is a way to realize scalability with respect to costs or weight.

The negative economic consequences of infrastructure variants are being appraised quantitatively, by assuming a fixed surcharge to unit costs. Still the virtual costs of handling are tolerated, primarily as the surcharge lies an order of magnitude below the cost of a single unit. Secondly, as the concept of a vehicle specific wiring harness is well established, and the additional costs of handling accepted. An impact of component variants on system integration costs and risks is not being appraised at all, even though a direct relationship can be safely assumed.

**Communality (Reuse)** – Communality of the product line describes the degree of reuse between various product lines. The question, whether to adopt and re-use an existing component from a distinct product line is basically a question, whether the potential savings in production outweigh the additional costs due to the development and integration.

This calculation clearly depends on the variability of requirements of individual product lines, as savings can only be realized if the individual requirements allow a significant reduction of hardware requirements. If this is the case, the reduction of hardware requirements to a reduced set of requirements has a significant impact on production costs.

On the other hand, product line specific infrastructure, i.e. communication technology does not result in sufficient cost reductions to justify the additional development and integration costs. In one documented case the savings in the order of several Euro per vehicle have not balanced the additional costs of development and system integration on the order of millions of Euro throughout the lifetime of the product line.

**Industry Compliance** – In line with communality at the product line level capturing the synergies between individual product lines, industry compliance captures the potential synergies due to possible off-the-shelf components thus avoiding costly individual development. The benefits of off-the shelf electrical components are not documented, however the development expenses in such cases easily lie in the range on the order of hundreds of thousands of Euro.

**Construction Kit Compliance** – Compliance with construction kit strategy is hard to measure, if the construction kit strategy is not explicitly defined. Still architecture alternatives in change requests are known to be rejected if they violate the guidelines of architecture development.

In general the construction kit strategy defines partitioning of components to systems as well as the partitioning of functions onto components. Thus the construction kit strategy is less restrictive than communality requirement, as it allows product line specific variants within certain guidelines.

As depicted in Figure 21 the most commonly appraised criteria is variability of components and industry compliance closely followed by communality, i.e. possibility of re-use of components. This can be explained by the economic relevance of these criteria for development and maintenance costs.

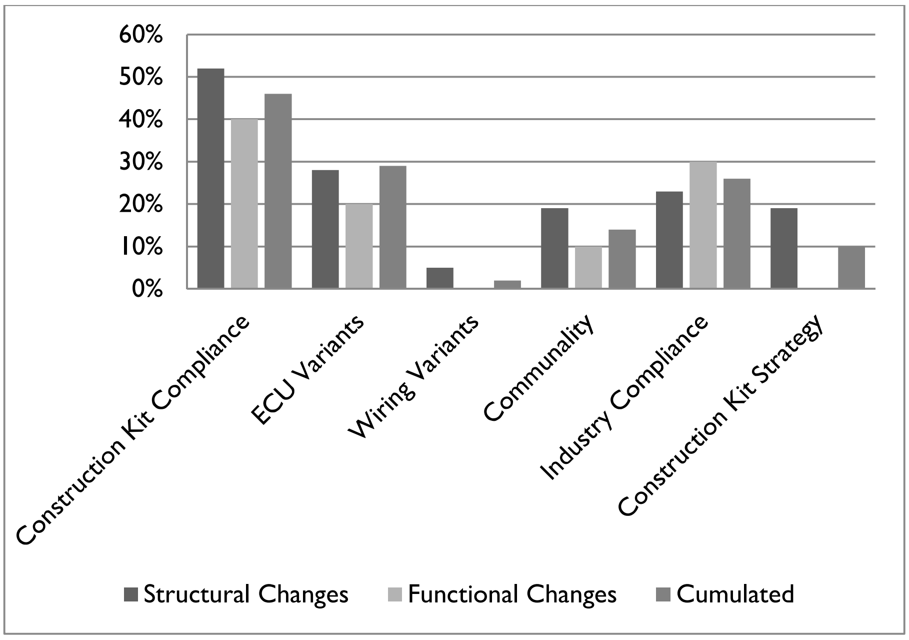


Figure 21 Frequency of appraisal of construction kit issues in architecture changes

As with the remaining construction kit related criteria, the economic benefit of compliance is not documented. A constructive definition of economic relevance would need to relate to risks and potential savings of future modifications prevented or made possible through strategic guidelines.

### 3.5.6 Vehicle Integration

Vehicle integration issues have been appraised in connection with structural changes only. As depicted in Figure 22 the relatively high number of one third of all architecture changes is basically due to distinct criteria being applied with each change. Electromagnetic compatibility has been assessed only in change requests regarding the physical infrastructure, not at the logical system connection level. The reason is that issues with electromagnetic compatibility can typically be solved by additional (functionally transparent) hardware at additional production costs. Thermal compatibility and packaging both have not been evaluated at all.

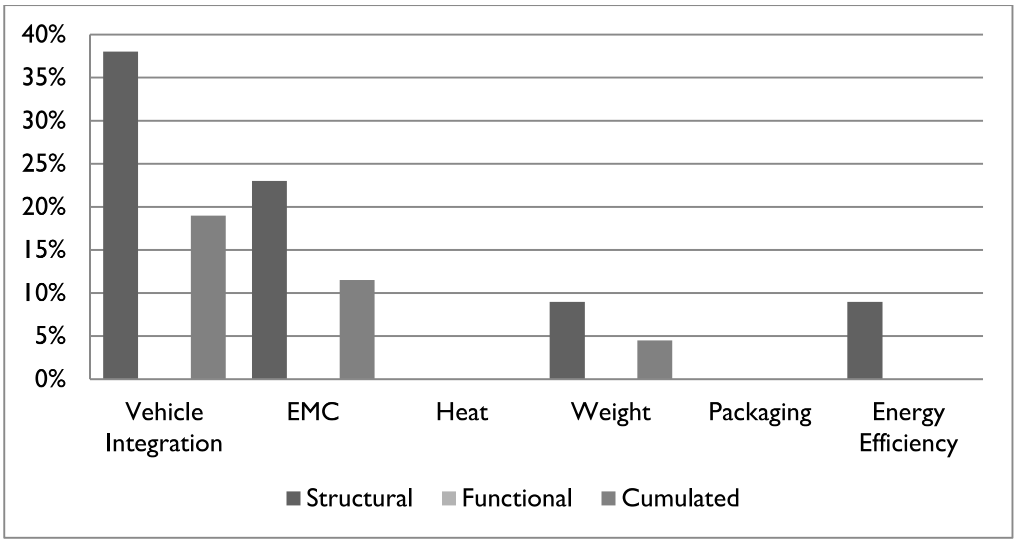


Figure 22 Frequency of appraisal of vehicle integration issues in architecture changes

The impact of architectural decisions on vehicle weight has been evaluated in connection with component integration. In all cases, the reduced component weight on the order of 100g is negligible if compared for instance with the total weight of a wiring harness of 25kg.

Energy efficiency has been appraised by contribution of the implemented functionality to energy savings, not by the ability of the architecture to provide the same service with lower energy consumption. The question of function re-partitioning in order to allow partial suspension of the E/E system has not been considered or documented so far.

All in all, the issues of vehicle integration are not being applied quite often. One reason may be that changes resulting from individual architectural measures are negligible at a global perspective, such as the 100g savings on component weight compared with the total weight of infrastructure. These issues should rather be targeted at a conceptual level to make significant difference in measurable goals.

### 3.5.7 Processes

Efficiency of processes has been appraised either as costs of assembly or of vehicle programming. In the documented architecture changes assembly costs, if elicited, were included in the business case of the change request and accounted for with costs on the order of 1€ per component installed. The efficiency of vehicle programming has been assessed in one case on an Ok/Nok scale considering the existence of a high speed communication connection to the component in question.

## 3.6 Summary

We have analyzed the current practice in architecture evaluation at a major premium brand OEM to gain an understanding of the most important requirements of architecture development and the relevance of individual criteria. Our findings show that a defined process for architecture change management as well as a set of evaluation criteria exists and are being applied; however there is still room for improvement with the methodology and appraisal of individual criteria listed below:

- The analysis of **production and development costs** is done consequently based on expert estimates. This works well for concrete architecture changes predominantly component integration vs. segregation. However, early estimates on infrastructure costs and systematic analyses of causes of inefficiency are not possible this way.
- **Development and integration costs** strictly do not assess the business value of architecture, but the complexity of the function being implemented or extent of structural change. Criteria measuring the ability to implement an arbitrary function (a.k.a. maintainability) are defined in abstract terms such as 'complexity' or 'future viability', which however are not economically defined.
- **Construction kit and architecture conciseness** considerations are commonplace, but vaguely defined. The evaluation category spans five independent criteria appraised subjectively. In addition economic benefits of construction kit compliance are not documented which makes trade-offs of construction kit criteria against cost savings arbitrary.
- Metrics such as **variability and scalability** do not express a design goal, but rather means to achieve efficiency in production. In fact variability is closely related to scalability and production costs efficiency, still variability is considered undesirable, while scalability a desirable property.
- **Vehicle integration criteria** are rarely appraised at all. This can be explained with lack of methodology and tool support at the architectural level but more importantly with the low impact of isolated measures on vehicle-level performance. These goals need to be addressed by conceptual means at the architectural level.

The main conclusion from the presented analyses is that vehicle E/E architecture is being optimized towards unit costs while the most critical risks and side-effects arising out of this optimization are limited by construction kit guidelines. A trade-off between violated construction kit compliance and unit costs however has not been documented. We will try to illustrate such trade-offs in the subsequent chapters.

The fundamental question regarding the presented model is, whether the defining set of criteria is *sound*, *complete* and *constructive*. An ancillary question concerns the benefit of the proposed subdivision into categories.

Regarding *soundness* we would like evaluation criteria to express an immediate, economically relevant, desirable property of the system (from the viewpoint of any stakeholder) – i.e. reflect the business value of architecture. Cost efficiency for instance clearly satisfies that requirement; variability on the other hand doesn’t as it positively affects cost-efficiency but negatively impacts development effort and risks. When defining a quality model bottom up, the valuation of this property should be made a criterion for its appraisal and criteria unable to be defined this way should be critically questioned for their purpose and justification.

Regarding *completeness* a proof that a model comprehensively covers system quality cannot be positively stated, still the opposite can be shown by counterexample. An indication for the completeness of the current model, at least from a practitioner’s perspective, is that the list is the result of practical architecture work and covers the requirements defined so far. But it is obvious that new criteria will arise or define quality from a distinct perspective. The requirement for completeness aims at the level of abstraction rather than the actual content of the quality model. We need to define quality at a level which is abstract enough not be invalidated by new requirements or stakeholders, but concrete enough to allow operational definition of criteria. A life-cycle and activity oriented analysis is one approach which we will pursue in subsequent chapters.

Regarding *constructiveness* we would like the criteria to be defined in terms of metrics to measure their achievement as well as in terms of architectural structure to guide the architectural design. Both requirements are not fully met by the presented model, besides the number of components (or functional density) as a structural metric related to costs. Effort of maintenance on the other hand is mentioned only indirectly through ‘complexity’ which in turn lacks a proper definition.

Regarding the *classification into categories*, we must conclude that any classification system is arbitrary. A benefit of classification is that common assumptions, economic key figures or methodology can be applied to criteria bound by a common category, however a model consisting of an unstructured list of criteria would be just as correct, though less convenient in application. The subsequent discussion proposes a categorization of criteria by a) the economic scaling factor of the respective property (or lack thereof) and b) by the variability of requirements to be met by the architecture instance.

Despite criticism, all of the architecture changes analyzed herein provide viable insight in the impact of structural properties of architecture on the value of architecture. The expert estimates mentioned therein will be used to valuate architecture qualities in subsequent chapters.

# 4 Architecture and Life-cycle Model for E/E Systems

In the previous chapter we have analysed the architecture decision and change process for automotive electrics-electronics systems. Current models for quality evaluation of automotive E/E systems are based on experiences made during the design of previous generations of the system and are often applied through a qualitative assessment of the individual criteria. We have concluded that this approach bears major uncertainty regarding the comprehensiveness and proper weighting of criteria.

In the design process, quality goals cannot be controlled directly, as design decisions primarily relate to the structure of the automotive E/E system under development. In order to analyze the impact of the E/E system architecture on quality goals, we need define an architecture model to formally describe structural decisions and relate these structural properties to quality goals. Our discussion of architectural properties is based on (Wild, et al., 2006) and (Broy, et al., 2009). In the following sections we discuss the levels of description and elements of this model and highlight the points relevant for architecture quality evaluation.

## 4.1 Architecture Model for E/E Systems

The architecture model used in this work is subdivided into three layers: a functional layer, a logical layer and a physical layer as depicted in Figure 23. For the discussion of certain properties such as electro-magnetic-compatibility, packaging or weight and cost efficiency of infrastructure a topology layer can be included in the model.

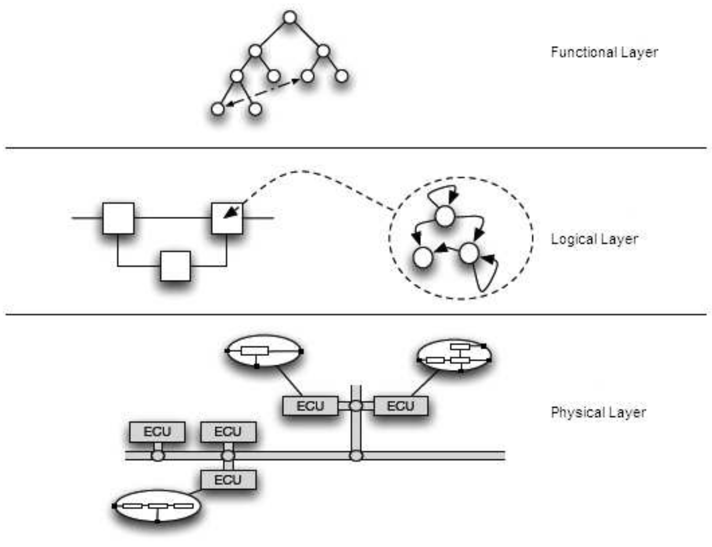


Figure 23 Architecture model for automotive E/E systems (Wild, et al., 2006)

### 4.1.1 Functional Architecture

The functional architecture $A_F$ defines a set of (externally perceivable) functions *F*:

The functional architecture describes the system from a black-box perspective. It consists of a hierarchy of functions containing the description of functionality that the system provides and the means of interaction with users and the environment. The functional architecture is the level of description to specify requirements, both functional and non-functional.

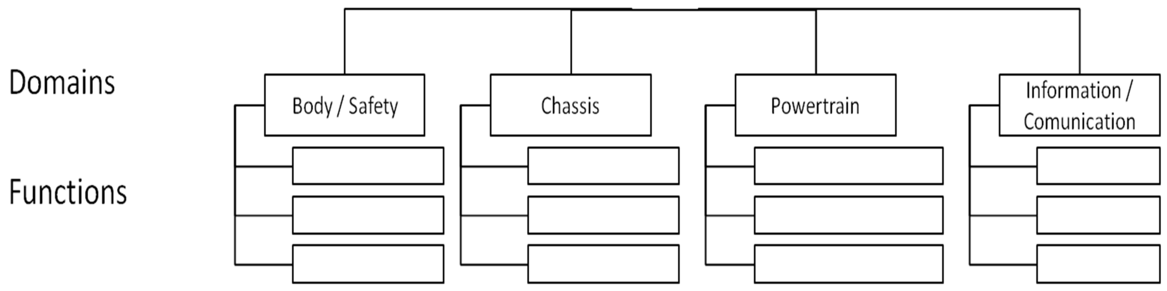


Figure 24 Functional architecture model

The structure of the functional architecture may be hierarchic. The criteria by which hierarchy levels and branches of this hierarchy are defined are not prescribed by the model. The functional architecture could as well be defined as a 'flat' list of functions to be provided by the vehicle.

In practice a grouping by semantic vicinity is applied, which by and large represents common responsibility and characteristics of functions. The top level branches represent functional domains such as chassis, power-train, body / safety and information / communication, as defined in (Frischkorn, 2005). The subdivision is reflected in the organizational subdivision of the product development division at the OEM (chassis, power train, communication and body) being responsible for the realization of the vehicle's E/E system, or vice-versa. It should be noted that novel, highly interconnected functions such as driver assistance systems spanning information and communication as well as chassis functions do not fit well into this division of responsibility and lead to additional functional interconnection and communication between these two domains.

Besides specifying the externally observable behavior of the E/E system, a function is the unit of separation and selection of functionality from the user's perspective. This means, at any given time the user utilizes zero or more functions whereas the remaining functions remain inactive. The usage patterns and their correlation of use of functions have impact on quality properties such as reliability or energy efficiency.

Finally a function serves as unit of configuration. With a subset of the functional architecture representing optional services which are being equipped with the vehicle at a surcharge, the combination of optional services defines the possible vehicle architecture instances to be built. Optional functions can be either independent of each other, meaning the function can be equipped regardless of the remaining configuration, exclude or necessitate each other.

From a structural analysis point of view, the identification of a function as a unit of specification, functional evolution, configuration, and utilization is sufficient. In order to analyze system behavior at runtime, further information can be specified at the functional level: reliability requirements, load and required response times for instance can be defined and propagated as feasibility criteria to the logical and physical architecture in order to assure sufficient resource capability.

## 4.1.2 Logical Architecture

The logical architecture $A_L$ describes the system from a glass-box perspective. It reflects the decomposition of a system into a number of logical components $C_L$, communicating via a set of logical signals $S_L$ to provide the functionality, defined by the functional architecture.

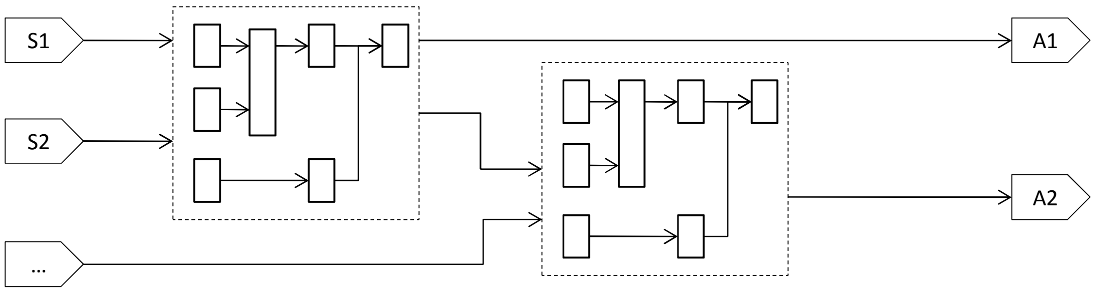


Figure 25 Logical architecture model

The logical architecture can be defined as:

$$A_L = \{c \mid c \in C_L, s \mid s \in S_L \}$$

The realization of a function defined as part of the functional architecture level is typically represented in the logical architecture as a chain of logical components, starting with one or more sources of logical signals, involving an arbitrary number of logical components providing defined application logic and ends with a number of signal sinks. Such a chain of components is referred to as an *event chain*. The relationship between the functional architecture and the logical architecture is a realization function R, yielding the set of components and signals providing the function:

$$R \coloneqq F \ \rightarrow \{c \mid c \in C_L, s \mid s \in S_L \}$$

The difference between the functional and the logical architecture is the definition of system internals. The internal structure of the system is defined in terms of logical decomposition into logical components, and definition of behavior of those components.

What's more important for the purpose of structural analysis is that logical components are the basic unit of reuse of functionality. Since a logical component can participate in an arbitrary number of services, even despite being executed only once, it is the logical architecture that reflects the actual computational complexity and load on the physical system to be built. Properties of the hardware architecture are not considered at the logical level, neither are the requirements regarding availability or timing. However a logical function can be attributed with its expected resource requirements.

In order to structure the logical model, logical components can be nested and a higher level component can encapsulate the composition of lower level components. This however is merely a question of modeling philosophy, if nested components are permitted and communication is allowed only between components defined at the same level, shared application logic, i.e. coupling between functions is expressed as communication links between higher level components. In the other case, if all logical components are represented at the same level, semantic correlation or shared application logic is reflected by multiple realization relationships between a functions and logical components, i.e. the participation of a logical component in more than one function.

(Wild, et al., 2006) define the specification of component behavior as the conceptual difference between the functional and logical architecture. For the purpose of structural analysis however the identification of a logical components as a unit of functionality ultimately provided by a physical component, is more important than their internal complexity, since this affects metrics defined at the vehicle level, such as resource efficiency, energy efficiency, reliability etc.

### 4.1.3 Physical Architecture

The physical architecture describes the actual components and infrastructure of the system to be built, which will provide the features defined in the functional architecture.

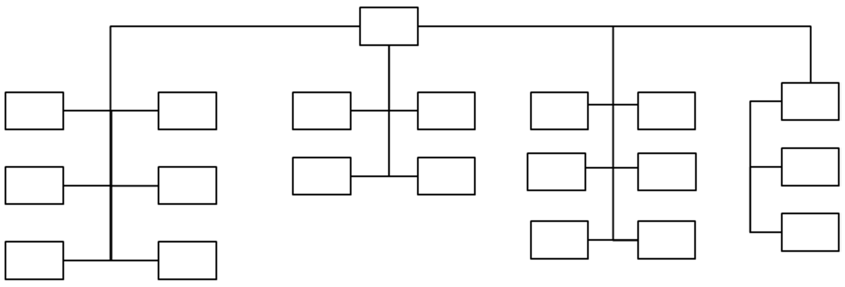


Figure 26 Physical architecture model

Concretely, the physical architecture defines a number of electronic components, software components and infrastructure:

Electronic components can be either a sensor an actor or an ECU, defined by their ability to produce, consume or transform logical signals. In practice the distinction between sensors and actuators on the one hand and ECUs on the other hand is somewhat blurred. Intelligent sensors can produce higher level signals, while ECUs have sensors and actuators installed thus act as one of these depending on the function under evaluation.

The infrastructure defines a number of communication relationships represented as systems connecting two or more electronic components, typically via a common communication buses or with direct connections between two components being a special case of a system:

$$I \coloneqq \{sys \mid sys \subseteq C_E\}$$

Additionally a system can be attributed with technical parameters specifying its bandwidth, latency, reliability etc. Logical components describing the semantics of individual services are represented as software components deployed on the physical architecture. The means of interaction with the user and the environment defined as sources and sinks within the logical architecture are realized as sensors and actuators.

Each software component is deployed on exactly one electronic component, described by a deployment function:

$$Deploy \coloneqq C_S \rightarrow C_E$$

The assignment of elements of the logical architecture to elements of the physical architecture is defined by the allocation (or partitioning) of functions. The allocation function yields the hardware component for a given logical component, whereas the Implementation relation yields a software component for a given logical component.

$$Alloc \coloneqq C_L \rightarrow C_E$$

$$Impl \coloneqq C_L \rightarrow C_S$$

That definition implies that allocation is merely a concatenation of implementation and deployment thus for subsequent analyses software components and logical components will not be distinguished, unless the distinction is relevant for quality analysis.

The computational resources required by the allocated logical components are provided by the ECUs and include processor time, memory and storage. Likewise the required communication bandwidth for the signals exchanged by logical components is provided by the infrastructure and dependent on its technical implementation.

Besides the computational capacity, other physical properties of the hardware can be specified at the technical architecture level, most predominantly availability or failure rates of components and infrastructure. Likewise energy consumption in different modes of operation can be inscribed in the physical architecture.

### 4.1.4 Topology

As stated previously, many of the physical properties of the vehicle's E/E system cannot be calculated by the logical information of functions and component interconnection alone. Particularly infrastructure weight, packaging and cost metrics require additional specification of the vehicle's geometry. This information can be specified in an additional the topology layer in the architecture model.

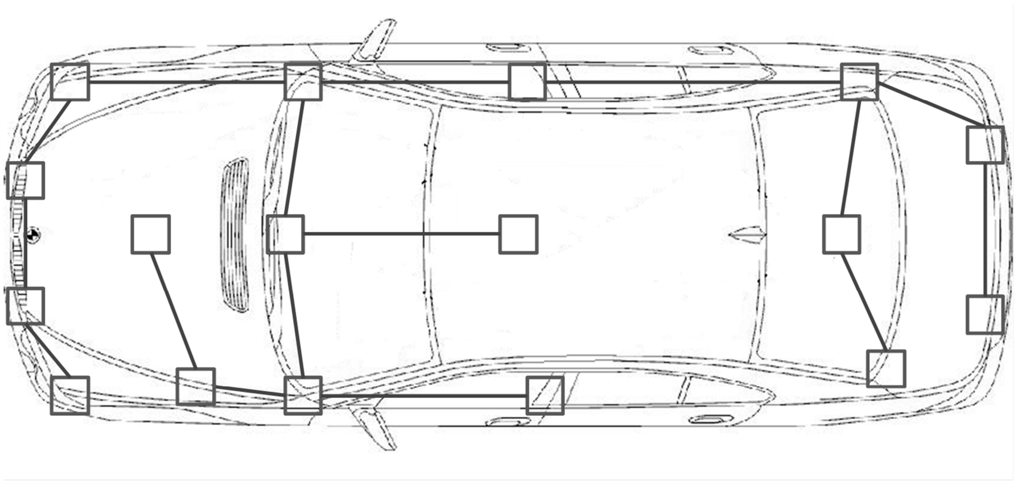


Figure 27 Vehicle topology model

The topology of a vehicle is defined by a number of topological locations $T_L$ in which electronic components, sensors and actuators can be installed and a number of topological routing paths $T_R$ along which infrastructure can be routed.

Whereas the predestined installation locations of sensors and actuators are determined by their function (e.g. tire pressure sensor, lights, steering wheel controls), ECUs can be placed freely within the vehicle, provided the physical boundaries of the respective installation spaces are not violated. Likewise the routing of physical wires is subject to constraints, defined by physical properties of the vehicle’s installation, such as exposure to water, heat, vibration and possible crash.

The relationship between the technical architecture and the topology is defined by the placement function, returning the topology location for an electronic component or the routing paths for a system:

## 4.2 Life-cycle Model for E/E Systems

In order to comprehensively assess the impact of automotive E/E architectures on quality we will analyse the activities and artefacts of individual phases in the life-cycle, as well as their associated costs, risks and opportunities. This analysis should yield a list of quality goals for in-vehicle E/E systems as well as organizational and environmental constraints – i.e. the external factors affecting life-cycle costs. Based on this model, we should be able to comprehensively evaluate the consequences of architectural decisions and trade-offs.

We distinguish three levels of architecture definition and instantiation: the vehicle architecture, the product line architecture and the architectural blueprint as depicted in Figure 28.

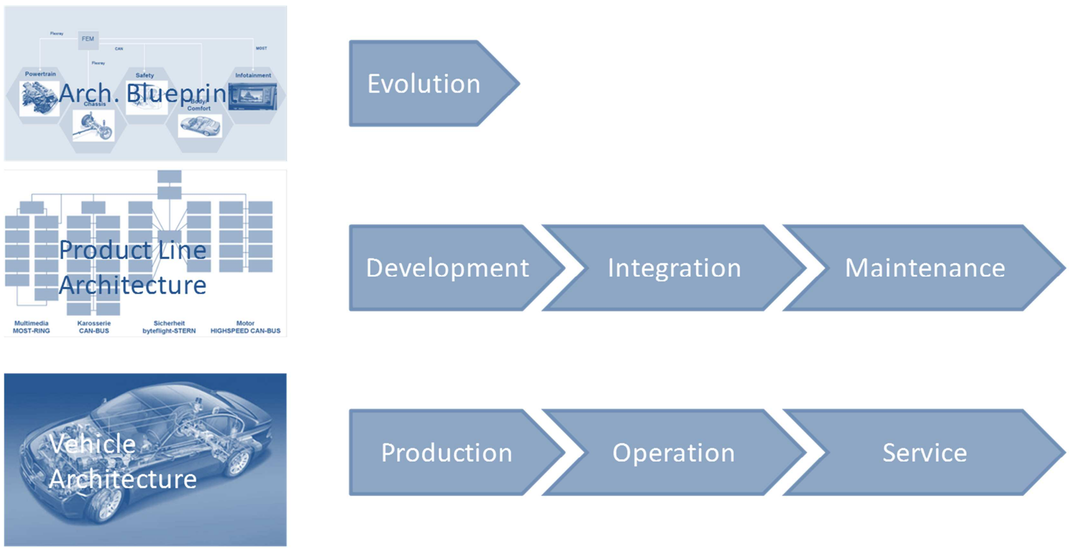


Figure 28 Levels of E/E system architecture definition and instantiation

The lowest level of abstraction is a concrete vehicle's architecture. The vehicle architecture is defined by the set of individual features (functions), the vehicle's motorization, geometry and so forth. The architecture model at this point is a 1:1 description of the components and wiring installed. With the specific set of components and features defined, vehicle-related metrics such as weight, cost, energy consumption and vehicle programming time can be calculated directly.

Whereas the vehicle architecture results from of a concrete set of features and a concrete production unit, the subject of development and integration is the product line. The representation of the system with its variability across all potential vehicles is the product line architecture. The product line architecture is typically the first design artifact in early stages of development and thus a primary subject of evaluation. Its evaluation needs to consider potential differences in vehicle configuration and individual take-rates, as well as development costs of software and hardware components and system integration and maintenance effort.

Just like the vehicle system architecture is an instance of the product line architecture, the product line architecture is instantiated from the architectural blueprint. Whereas product line architecture defines a range of possible vehicle configurations, the architectural blueprint defines common guidelines and designs for a set of product line architectures, each representing a class of models.

Product lines are being developed in parallel with some delay between their individual development phases. The number of parallel product line architectures being developed depends on the manufacturer's model portfolio and product line strategy. Each product line under development is subject to trade-offs between reuse of existing designs and re-development of components. Each re-design of components or architecture change is hence a contribution to the architectural blueprint available for subsequent or parallel product lines. The evolution of the architectural blueprint is hence a constant divergence and consolidation of alternative designs in order to adapt to evolution of requirements. The extent to which components can be re-used between distinct product lines depends on the variability allowed by the architectural blueprint.

### 4.2.1 Vehicle architecture

The life-cycle of a vehicle covers production, operation and service. Whereas activities in the development and maintenance phases of product line architectures apply to a handful of product lines and derivatives, mass production increases the number of units by several orders of magnitude. Thus marginal savings in production costs or reductions of probability of failure scale up to significant differences in absolute numbers. This explains why architectural analysis in the automotive industry so far concentrates on unit costs.

Unlike in product line architecture development, no design decisions are made at this level. Properties defined at the vehicle level reflect the quality of product line architecture they are derived from. The actual characteristics of quality properties are affected by the individual configuration of the vehicle in terms of motorization and optional equipment.

While the architectural properties affecting cost structures are fixed, external factors are still subject to uncertainty – errors detected after delivery, failures due to wear-off and upgrades initiated by the customer, all of which need to be expressed statistically.

#### 4.2.1.1 Production

In production, a vehicle is assembled according to a customer specific feature configuration. Vehicle production itself is a linear process consisting of assembly of parts and testing of the assembled vehicle.

The part costs and duration of assembly and configuration are directly accountable as production costs. At the end of the assembly line, an automated end-of-line test is conducted on the E/E system. If the produced vehicle passes the functional test, the vehicle proceeds with delivery and otherwise the vehicle needs to be diagnosed and corrected, thus incurring additional costs. The probability of errors in assembly requiring additional correction activities is likewise a measurable property of quality in production.

#### 4.2.1.2 Operation

Operation covers the usage of the vehicle by the customer. Properties related to operation include resource consumption providing the specified functionality, the probability of failure to do so and the level of service attained in the case of partial failure. We measure the costs of operation as the resources consumed to provide a function, or as metrics directly related to resource consumption such as the weight of the E/E vehicle architecture and its energy efficiency.

Given the set of functions offered by the vehicle, resource consumption can be described in more detail knowing the individual usage patterns. For a less detailed analysis, the operation can be subdivided into phases of driving, resting and parking with individual sets of functions being active at that time, all of which imply the operation of a part of the vehicle’s E/E system.

To measure the provision of service we use the definition of availability and reliability to describe the ability to provide degraded service in case of partial failure. Besides design errors, failures can occur due to hardware wear-off.

#### 4.2.1.3 Service

Service at the vehicle level refers to the effort required to analyze and correct errors in the vehicle’s E/E system after delivery. In line with the remaining vehicle-oriented, the economic relevance of service activities is affected by the number of vehicles, by the fraction of vehicles subject to service activities, and the individual effort per service operation. The means at the vehicle level shift from proactive means such as specification and validation, towards reactive measures to ease diagnosis and replacement of components and vehicle programming.

Errors in the operation phase are detected by malfunctions of the vehicle’s E/E system, i.e. failure to provide a function as expected. The measures to correct the error typically involve an update of the software installed in the control units or an exchange of the component deemed to be defective. The fraction of unnecessary component replacements can likewise be defined as a metric for quality in service.

A vehicle is produced and delivered with an individual configuration of features defined by customer by the time of production. The possibility to extend the feature configuration after delivery is beneficial both from the OEM’s perspective, because of increased take-rates of equipment and additional revenues, as well as from the customer perspective through increased value. A further motivation for functional upgrades of vehicles after delivery is the lifecycle mismatch. The life-cycle of a vehicle significantly exceeds the innovation cycle of electronics and automotive applications, thus new features are available on the market long before the introduction of a new product line or even a new model.

An upgrade-process is similar to maintenance-process. In the simplest case the change requires only software or configuration data change, in case the service to be installed requires additional hardware, sensors or actuators, the respective components need to be installed as well. In that case the infrastructure needs to be modified to integrate the additional hardware with the rest of the E/E system.

### 4.2.2 Product line architecture

The product line architecture is the level of development activities, as opposed to production, operation or service. Activities related to the specification, development, verification and validation and maintenance of the E/E system for a defined range of models with common set of requirements, geometry and physical characteristics can be described at the product line architecture level. The product line architecture embodies a major part of architecture design decisions affecting the cost structure and quality characteristics of individual vehicles but is in turn constrained by the architectural blueprint.

Contrary to the vehicle level, costs at the product line level do not scale with the number of vehicles produced. Secondly statements on concrete vehicles such as costs, weight and other efficiency metrics can only be made with respect to a certain configuration. Thus the accuracy of assessment of quality properties of derived vehicle architectures is limited to the extent assumptions can be made about typical feature configurations.

The development process of product line architectures can be described by the V-Model depicted in Figure 29.

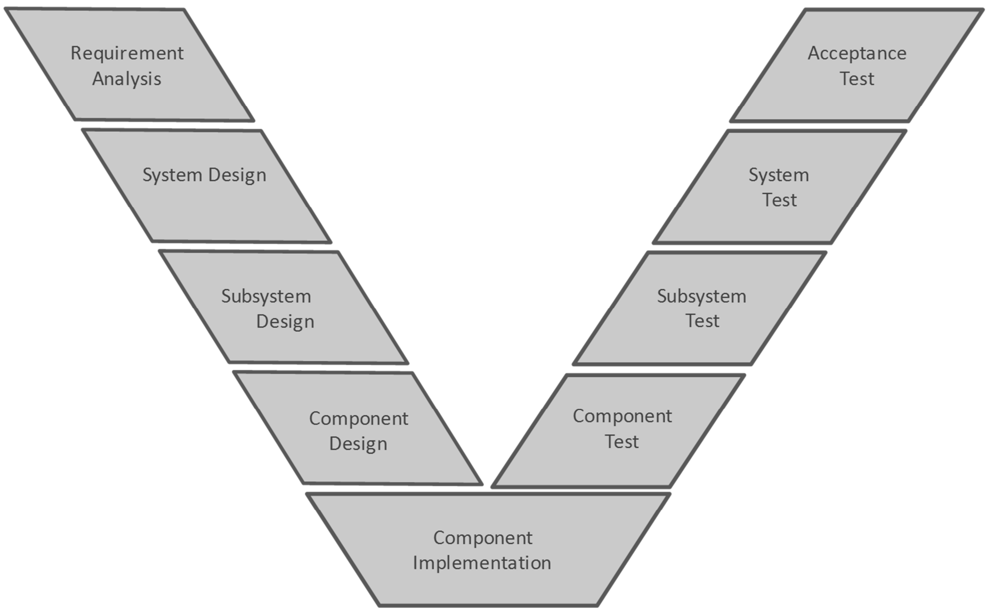


Figure 29 Classical V-model in system development

By this process model, the specification of lower level components relies on previous definition of their interfaces based on a functional decomposition of higher level elements. Analogously, the verification of functionality of higher level elements depends on previous implementation and test of their constituent parts.

The classical model suggests that a partitioning of the system into smaller units and definition of interfaces between them is derived purely on functional requirements defined up front. In practice the structure of the E/E architecture is not being developed from scratch, but rather inherited from previous ones and the architectural blueprint, and can only be adapted within limited bounds. The interfaces between subsystems and components and their behaviour on the other hand are dictated by the functions deployed therein.

Particularly the subdivision of the E/E architecture into domains and subsystems has proven considerably more stable that the set of functions implemented by its components.

Due to reuse of existing patterns of physical architecture, functional and structural design of E/E architecture in the development process is separated, as depicted in Figure 30. The actual development process for automotive E/E systems deviates from the classical V-Model in two ways: it is iterative and it parallelizes functional and physical design.

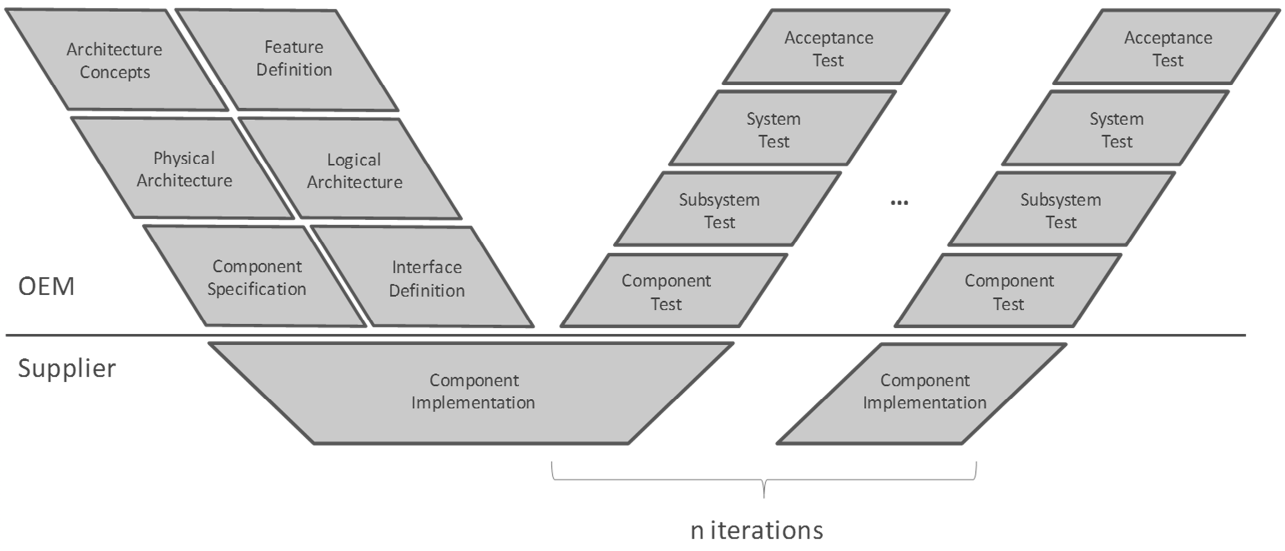


Figure 30 Product line architecture development process model

### 4.2.2.1 Architectural Design

The initial phase of architecture definition involves the definition of the function set to be provided by the product line architecture and the derivation of basic concepts for the product line under development. The main design decisions made here are the re-use of existing components, re-implementation of components in order to achieve cost-down effects, or to re-allocate existing functions onto components. These basic concepts are the very same decisions whose evaluation has been discussed in chapter 3.

What is to be referred to as architecture is defined in these initial stages of the development process. The resulting artefact of architectural design is a model as described in section 4.1 containing the feature set to be provided by the E/E system under development, the set of systems and components providing them and the allocation of function chains onto systems and components.

The structural and functional design of architecture converge in a separate partitioning step in which the functional units defined as part of the logical architecture are assigned to the physical structure of subsystems and components. The combination of physical decomposition and functional description yields the component specifications to be implemented by a supplier.

#### 4.2.2.2 Specification and Development

The specification of software architecture and development of software modules lies in the domain of suppliers. Any architectural structure below the level of components is hence not in the scope of this work.

Development is based on the component specifications delivered by the OEM. The economic goals related to component development can be intuitively identified as time and cost of development. However the actual extent of individual development by the supplier is not necessarily transparent to the OEM, since not all development costs are directly accounted but partially added to the component's unit costs. The complementary economic aspects in development are development time and correctness of the components, both of which are economically relevant and directly measurable at the OEM.

#### 4.2.2.3 Verification and Validation

While the true development costs of individual components can be hidden in the supplier's unit cost calculation, the verification and validation activities as part of the system integration process affect the OEM's core competency and involve the most internal resources in the development process.

The verification and validation phase covers the right branch of the V-Model. In this phase the specified components are incrementally implemented and tested up to a full level of functionality. The discussion of the verification and validation process is based on (Rumpf-Steppat, et al., 2005) on integration processes at BMW and (Olejniczak, 2007) on integration at Audi.

A single iteration in the integration process is called "integration level" and involves the development of hard- and software component prototypes with a defined level of functionality and integration tests at the component, subsystem, system and vehicle level. Subsequent integration levels come gradually closer to a level of functionality and quality required for production. For each main integration level a number of integration cycles is executed, depending on the number of errors detected in verification and validation.

A scheme of the integration levels during the development, verification and validation phases is presented in Figure 31.

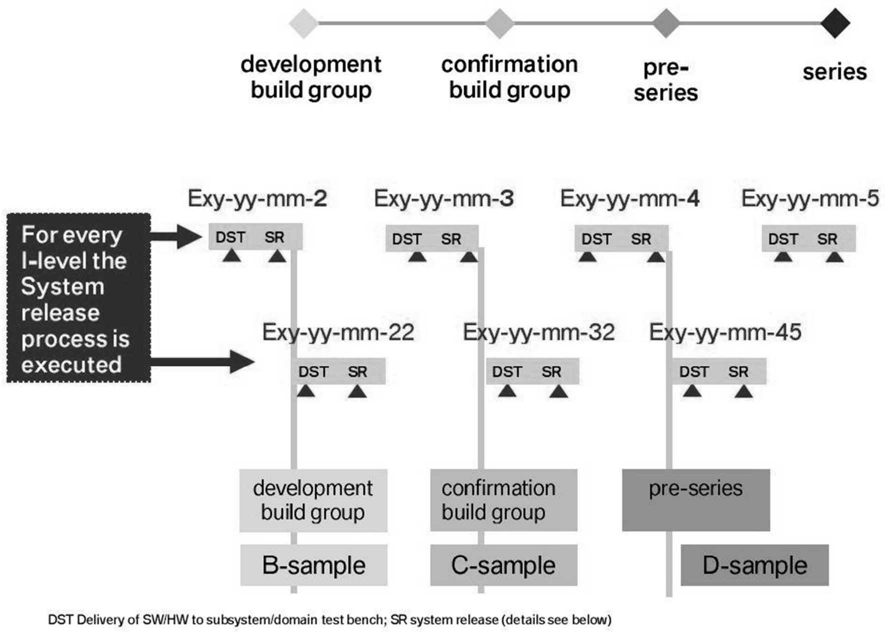


Figure 31 Integration cycles in the development phase (Rumpf-Steppat, et al., 2005)

The quality of the E/E system with respect to verification and validation activities can be measured by the number of integration levels required in the development process, by the number of required test and prototype setups and the number of errors detected. The verification and validation process is limited both in time as in budget, hence all of the aforementioned metrics can be lowered by management decisions without increasing the quality. Therefore as a complementary metric the number of errors detected after start of production must be defined to capture the quality of the E/E system with respect to development, verification and validation.

#### 4.2.2.3.1 Integration Levels

Each integration setup is tied to a defined level of functionality. The set of features to be implemented at a certain integration level depends on their criticality, mutual dependencies and interactions with the environment. Low-level, highly depended on functions and functions with a higher implementation risk are tested in earlier integration setups.

As depicted in Figure 31 the integration levels throughout the development and integration phase are structured according to the maturity of the system and completeness of functionality. Consecutive integration tests are performed on the *concept confirmation prototype*, the *development build group*, *confirmation build group*, *pre-series components* and *production grade components*. For each I-level the complete system release process is executed.

**Concept Confirmation Prototype**

Integration starts as soon as the initial concepts for infrastructure, power supply and geometry are defined.

The infrastructure and system functions are to be implemented by the first integration level, the concept confirmation prototype. The concept confirmation prototype is assembled some 3 years before the start of production (SOP), therefore components employed in this step build up on predecessor versions of the hardware to be installed in the vehicle which introduces an uncertainty regarding the load on the communication and power infrastructure generated by the actual components and the behaviour of system functions.

The purpose of the concept confirmation test is to verify the capacity of the communication infrastructure and power supply for the intended configuration of hardware components. In subsequent integration levels executed on the development build group, confirmation build group and pre-series, basic and advanced user functions are implemented and integrated gradually.

**Development Build Group**

The components installed in the development build group build up on the actual hardware to be installed in the product line under development, but differ in terms of packaging and implemented functionality. Functions to be implemented as part of the development build group potentially bear a higher risk on the development process and start of production, hence the early implementation and integration. The functions to be implemented up to this level include features available in the concept confirmation prototype and basic user functions.

Examples are functions with a high degree of interaction with the environment, i.e. chassis and engine control and functions requiring a system level setup for integration, i.e. functions involving communication across domain boundaries. Likewise differentiating features are to be implemented in the development build group.

**Confirmation Build Group**

In the confirmation build group advanced user-perceivable functions are implemented and integrated. Advanced user functions include functions requiring little interaction with the physics of the vehicle, and little potential of market differentiation. Examples are cell phone integration and data services. By that integration level, basic user functions exhibit a maturity level to pass positive and negative tests as well as stress tests. The related prototype component is supposed to be identical with the intended production grade component, as far as hardware, packaging and interfaces are concerned.

**Pre-Series and Series Prototype**

In the pre-series integration level no new functionality is added to the vehicle. Implementation concentrates on the off-board infrastructure and functionality for service and support. Between the pre-series and the series level, no new functionality is implemented. The test cases executed in the system release test are to verify the production and delivery process with production grade components.

#### 4.2.2.3.2 System Release Process

Within each individual integration level, the architecture under development is tested in a bottom up fashion following the standard V-Model. The complete cycle of component, subsystem, system and acceptance test is called the system release process.

Components are tested in isolation upon delivery and subsequent tests of distributed functions are executed in domain test setups, i.e. in domain-specific subsystems. Finally interactions with mechanical and physical components and full control loops are tested in a system level prototype. Depending on the nature of functions to be tested, the system prototype is realized as a static or dynamic labcar. Finally the acceptance test of features and tuning of parameters related to driving dynamics is conducted in a vehicle intensive test.

**Component Tests**

Component tests are conducted by the suppliers as part of their development process. At the OEM, individual components pass an entry test by the organisational entities (centres of competence or CoCs) responsible for their specification and development.

According to (Olejniczak, 2007), three kinds of test cases can be distinguished at this level: performance tests, functionality tests and communication. Performance tests ensure the functioning of components in varying conditions, such as heat, cold, vibration and electrical disturbances. Tests for functionality and communication verify that the component behaves according to specification, both when interacting with the environment through actors and sensors, and with the rest of the architecture through exchanged messages.

In addition to functional tests, general operation characteristics of the component can be measured in isolation, such as reset-free operation, sleep mode and wake up, quiescent current as well as system functions for flashing and diagnosis.

**Subsystem Level Test**

While the component tests at the CoCs focus on customer functions and system functions such as diagnosis and coding, the subsystem integration tests concentrate on the functionality of the network and interfaces. The subsystems of automotive E/E architecture are part of the structural definitions dictated by the architectural blueprint. Typically the system is subdivided into application domains such as power train, chassis, safety and infotainment (Reichart, 2005).

The subsystem integration test concentrates on functions provided cooperatively by distributed components, but more importantly on coexistence of functions in an interconnected environment.

Subject to subsystem tests are: network performance particularly while power up and power down, temperature tests, voltage tests, application of coding data to the complete network, verification of vehicle programming and diagnostics of each component in the network, and energy management.

Aiming at verification of functions provided cooperatively by more than one component, tests of event chains are executed during subsystem integration. Since most components participate in the provision of more than one function, event chains are inherently overlapping. Consequently subsystem integration tests cannot realistically represent all simultaneous functions in real operation. A probabilistic approach to detect interferences due to coexistence of event chains is the system level test.

**System Level Test**

Within the system integration process the labcar is the first place to assemble the complete E/E system of a vehicle including hardware components, software and the original wiring harness. The main focus of labcar tests lies on system functions and communication and networking aspects of the E/E architecture and on the subsystem interfaces. The system level test can be subdivided into two distinct setups: a static and a dynamic labcar. (Rumpf-Steppat, et al., 2005).

Whereas most of the non-functional characteristics, such as power up and power down network behaviour, bus loads and power consumption can be measured in the static setup, functional tests related to driving experience and customer perceivable functions are conducted in a dynamic setup. The purpose of the labcar setup is to increase the realism of test cases. The control loops include actual mechanical and physical connections as well as robustness of the control algorithms to erroneous data and mechanical factors, such as vibration and water resistance etc. can be only tested realistically in a prototype.

During the labcar test phase a standardized test program is run through to perform all system tests: check of plant and diagnostic processes, energy management, current and voltage tests, communication logic and timing, error management and behaviour in fault situations etc. The foundations for these tests are defined test procedures executed in every I-level. Errors found during these tests are likewise communicated to the development departments. The maturity level of the respective function is determined based on the errors discovered here.

**Vehicle Level Test**

The final level of integration testing is the vehicle intensive test˙ In the vehicle intensive test, test drives involving realistic, customer-typical vehicle operations are conducted, in order to simulate an as complete as possible set of situations and verify vehicle behaviour.

The purpose of the vehicle intensive test is summarized in (Reif, 2006): The development and integration process is largely based on abstractions and simulation which might insufficiently reflect the actual behaviour of the vehicle’s control loops or the environment. At the same time acceptance criteria of the overall system ground on subjective perception of the driving experience. Test drives with a fully assembled vehicle generate realistic feedback about the vehicle’s behaviour and allow a fine tuning of parameters to make up for inaccurate abstractions in the development phase. Diagnosis in vehicle intensive test prototypes is restricted to measuring and logging, likewise reproducibility of errors occurred is low.

By the end of the complete test cycle a quality level of the architecture under development is determined. This numerical value reflects the confidence in the integration stage, with respect to the performed depth of testing.

#### 4.2.2.4 Maintenance

The product development process ends with the start of production (SoP). The subsequent maintenance and support phase of a product line continues during the production phase until the end of production. Maintenance and support activities aim at the improvement and modification of a product line in production. The maintenance and support phase covers modifications to the product line architecture which can be categorized according to the cause of the change and the architectural elements affected by the change.

The primary reason for changes to the product line is removal of errors discovered after start of production. Error corrections take effect in vehicles produced from there on. Costs of error removal in already delivered vehicles are accounted for at the vehicle level, as its costs depend on the number of vehicles affected.

A further kind of maintenance activities is re-design of components to cope with the life-cycle mismatch between vehicles and electronics and to exploit decrease in prices for electronics during the production phase. These lead to cost-down modifications and re-implementation of individual components. Finally introduction of new features to existing models and introduction of derivatives with distinct feature sets require changes to a defined architecture after start of production.

The following sections discuss the specific issues related to maintenance scenarios describing error removal, transparent exchange of components in the architecture and functional upgrade. Quality of architecture is reflected in the effort to perform these modifications and the risk of unwanted side-effects.

**Error Correction**

Despite verification and testing in early phases of system development design and implementation errors undetected until production and delivery occur regularly in practice. Warranty costs accounted directly to the OEM easily become astronomical, particularly if errors lead to a call-back of parts of the product line. In addition image loss and resulting loss of profit easily exceed the pure costs of error correction.

The detection and removal of design and implementation errors discovered after start of production is a most cost critical task during the maintenance and support phase, since the error is being reproduced in vehicles in production. The frequency of necessary error corrections cannot be predicted exactly. For a quantitative evaluation of expected maintenance incidents a probability must be assumed. The risk of errors depends on the results of earlier phases, particularly the correctness and completeness of specification, implementation and the effectiveness of testing, outlined in previous sections.

A scenario of error correction involves an identification of the root cause of the malfunction, a correction of the respective component or artefact and integration of the corrected component into the vehicle. Effort related to error correction can hence be subdivided into effort related to analysis, and modification of specification and implementation artefacts and integration.

Corrections to components in the product line architecture in production are delivered cumulatively in regular releases, analogously to integration levels in the development phase. The system release process is executed regularly throughout the maintenance phase however its extent is reduced by compartmentalization of architecture and the severity of changes to the architecture.

**Structural Modification**

Besides the removal of errors, product line architectures undergo modifications during the production phase, to leverage on increased capacity and decreased component prices and to cope with the life-cycle mismatch between vehicles and components. As opposed to error correction, costs of potential transparent modifications can be predicted more reliably, as the frequency of such changes can be planned throughout the maintenance and support phase.

The subject of a transparent modification to a product line is a component – an electronic control unit including the software components deployed therein. As the modification is functionally transparent, the evaluation scenarios of cost-down measures need to consider only development and integration efforts and risk of side-effects caused by implementation errors or deficiencies in testing of the successor version of the component. Effort related to specification and design is not considered in this scenario.

**Functional Upgrade**

Changes to an existing architecture such as the introduction of new features likewise occur in regular intervals throughout the maintenance phase. A functional upgrade or model thus can be described as a change to the feature set of the product line and incurs costs of specification and development and integration dependent on the kind of function to be introduced. For functional upgrade activities, the metrics defined for the development and verification and validation phase can be applied.

An exemplary scenario for a functional upgrade can be represented as an instance of the development process: a function added to the current feature set and the resulting modifications to the architecture. The design of the new feature results in a change to the logical architecture, either through modification of existing functions or addition of functions and events to the logical architecture.

Finally, the resulting mapping of the modified function network yields the resources required to provide the additional feature. These result in either in introduction of new electronic components or the deployment of additional software components onto the existing infrastructure, along with increased load on the communication network and ECUs.

### 4.2.3 Architectural Blueprint

The development of an in-vehicle E/E system for a concrete product line builds up on existing architectural patterns and components as well as a common set of functions shared by all product lines whose evolution can be traced across generations of E/E architectures. In general, the ability of architecture to adapt to varying requirements, both over time, as well as across distinct product lines, can be attributed to quality of the architectural blueprint.

The architectural blueprint represents the rules and patterns for architecture development. These rules affect the compatibility of parallel product line architectures and the extent of individual development required for the instantiation of a new product line and introduction of new functions. The definition of the architectural blueprint is thus both a constraint on product line architectures as well as an architectural mean to globally trade-off development costs vs. production costs.

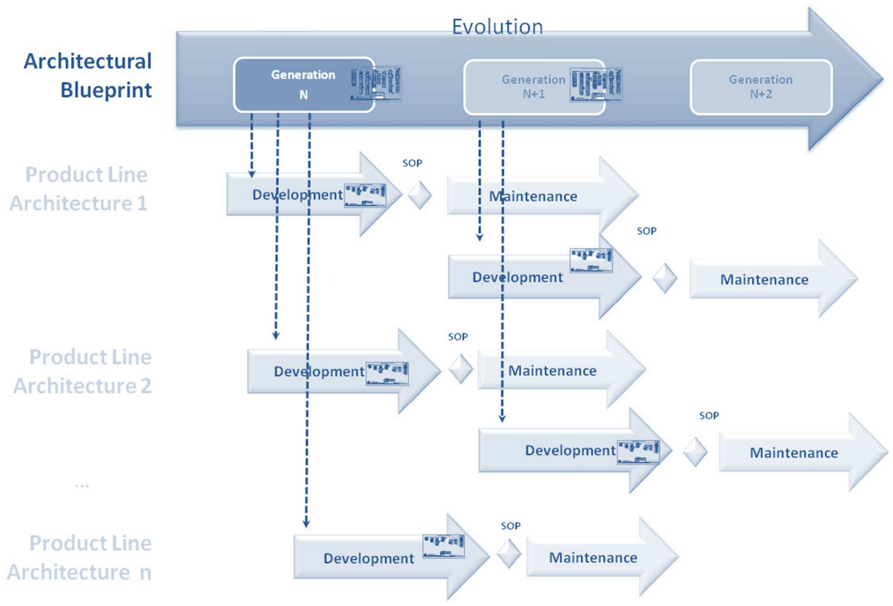


Figure 32 Instantiation and evolution of product line architectures

### 4.2.3.1 Instantiation

With respect to product line instantiation, the main architectural design decision is the extent of re-use and product-line specific development. This decision depends on the product-line specific feature set to be provided, and whether a (reduced) functionality and product-line specific optimization yield sufficient efficiency gains to justify the investment in development and integration.

Without further specification of the variability of requirements across individual product lines, an assessment of the suitability of the architectural blueprint remains very vague. In general the quality of the architectural blueprint is reflected in the ability to leverage product line specific without further changes to common parts of architecture.

Each instantiation of a new product line is also an opportunity to extend the feature set trough innovations and to adapt these innovations to existing, parallel product lines. The quality of the architectural blueprint is reflected in the effort to adapt an innovation to parallel product lines and depends on the compatibility between individual product line architectures.

#### 4.2.3.2 Evolution

According to (Axelsson, 2009) a re-design of the architectural blueprint is conducted once every 5 to 10 years resulting in the definition of a new architecture generation. The evolution of the architectural blueprint serves the purpose of changing structure to reflect the evolution of functional and organizational requirements, i.e. to increase capacity or organizational and functional homogeneity of structural elements (systems and components) and to reduce dependencies between structural elements.

The evolution of requirements over the next 5 to 10 years and beyond is hard to predict. The quality of the architectural blueprint is reflected in the effort and risks associated with such a change. The impact on concrete, derived product line architectures in terms of risk and costs of structural changes can be lowered by reducing the coupling between services and components and components and systems.

## 4.3 Summary

In this chapter we have introduced the structural model used to describe architectural changes and the life-cycle model defining the activities by which to judge the quality of the system under development. We have identified the distinct levels of abstraction defining architectural decisions – the vehicle architecture, the product line architecture and the architectural blueprint. This distinction is crucial, not only because of the different scaling factors - per vehicle, per product line or global – but more importantly, because of the inherent variability of requirements to be satisfied by the specific architecture definition. Defining a product line the architecture has to account for the optional functions to be individually configured per vehicle, at the architectural blueprint, variability of product-line specific requirements needs to be considered.

The insight from the initial discussion of the various levels and life-cycle phases is that prediction of system properties becomes the more uncertain the higher the level of abstraction. Eventually statements on evolution and instantiation of the architectural blueprint are vague by nature, unless accompanied by concrete scenarios, defined at the functional and logical levels of the architecture model.

Having defined both the architectural means and the quality goals, we will investigate the economic dimensions of the life-cycle phases drawn here and the relationship between architectural structure and quality with respect to these activities in the subsequent chapter.

# 5 Quality Model for E/E Systems

## 5.1 Cost Structures in Automotive Electronics

As quality goals are often subject to trade-off discussions, we give a brief overview of the economic importance of distinct activities and life-cycle phases before presenting the interrelationship between architectural structure and quality goals. The purpose is to indicate the potential of optimization by architectural means and the possible return on investment in architectural flexibility and extensibility.

### 5.1.1 Composition of Vehicle Retail Price

In a 2001 Mercer study (Mercer 2001) the total costs of software and electronics has been estimated to 16 per cent of a vehicle's total value in 1990, 25 per cent by 2001 and predicted to be at 40 per cent in 2010. In 2010 the figure of 40% is referred to as 'most representative for current industry' (Blanco, 2010) but its accuracy cannot be confirmed by these sources. Other publications state that the level of 40% will not be reached before 2015 (A2A, 2010), stating that currently electronic components make up 10-15% of the total production cost of a 2007-model compact car such as the Toyota Corolla, for 20-30% of the cost of luxury models like the Lexus-brand, and for around 50% in the case of hybrid electric vehicles (HEVs) such as the Toyota Prius.

The fraction of 40% refers to total costs of production. These costs can be deduced from the suggested retail price. According to (Vyas, et al., 2000) costs of production account for 50% of the retail price, the remaining shares of direct and indirect costs reported by this study are listed in Table 4. Compared to production, research and development as well as engineering which, in the case of E/E architecture, includes development as well as system integration, amounts to merely 6.5% of the vehicle retail price.

| Cost Category | Cost Contributor | Relative to Cost of Manufacturing | Share of MSRP (%) |
|---|---|---|---|
| Vehicle Manufacturing | **Cost of Manufacture** | **1.00** | **50.0** |
| Production Overhead | **Warranty** | **0.10** | **5.0** |
| | **R&D/Engineering** | **0.13** | **6.5** |
| | Amortization | 0.11 | 5.5 |
| Corporate Overhead | Corporate Overhead, Retirement and Health | 0.14 | 7.0 |
| Selling | Distribution, Marketing, Dealer , Support, and Dealer Discount | 0.47 | 23.5 |
| Sum of Costs | | 1.95 | 97.5 |

| | | | |
|---|---|---|---|
| Profit | Profit | 0.05 | 2.5 |
| Contribution to MSRP[5] | | 2.00 | 100.0 |

Table 4 Costs of production, manufacturing and engineering relative to retail price (Vyas, et al., 2000)

An exemplary calculation for a BMW 1 series is given in Table 5. The cost of electronics is assumed to be 30% of the total production costs as indicated above for premium brand models. Prices are taken from the BMW website as of November 2010.

| **Cost** | **BMW 120i basic** | **BMW 120i w. optional equipment** |
|---|---|---|
| Retail Price | 26.550 € | 34.250 € |
| Production Costs | 13.275 € | 17.125 € |
| Production Costs E/E | 3.982 € | 5.137 € |
| Warranty | 1327 € | 1712 € |
| Warranty E/E at 30% | 398 € | 513 € |
| Warranty E/E at 60% | 796€ | 1026 € |
| R&D and Engineering | 1725 € | 2226 € |
| R&D and Engineering E/E at 30% | 517 € | 667 € |
| R&D and Engineering E/E at 60% | 1034 € | 1334 € |

Table 5 Example calculation of production costs of electronic components

If applied to the 1-series BMW, costs of production of E/E components amount to roughly 4000€ to 5000€. (Magna, 2005) reports costs in the range of $2,500 to $4,800 for the years 2005 and 2015 respectively. The lower figures presented therein may be attributed to mass-market models taken into account as opposed to premium vehicles.

Out of the total E/E costs the infrastructure (wiring harness), if considered separately, is the most expensive part of the E/E system and the second heaviest part of the vehicle. A wiring harness consists of roughly 2.5km of wires, weighs up to 80lbs and incurs costs in the range of 190$ to 570$ for copper alone (as of 2006) depending on product line, size and equipment of the vehicle (Rhines, 2007).

The costs of infrastructure i.e. the wiring harness of a vehicle are highly variable and dependent on the configuration and type of vehicle. (Leoni, 2010) reports a price range from 100€ for economy, mass market models to 1500€ for fully equipped luxury class, premium brand models (as of 2010). Assuming costs of infrastructure on the order of 400€ to 600€ for the exemplary 1 series BMW, as a compact class, premium brand model, infrastructure represents roughly 10% of the E/E production costs.

Whereas the production costs of E/E components can be derived from their fraction in total production costs, warranty and R&D costs for the E/E system are not necessarily just the corresponding 30% share of the expenses at the vehicle level. In fact some studies indicate that the E/E system is more test intensive and error prone than the mechanical parts. Studies by the ADAC report the electronic system to be responsible for 50% to 60% of vehicle breakdowns (Dudenhöffer, 2004), as depicted in

[5] Manufacturer's Suggested Retail Price

Table 6. These numbers however are dominated by battery outages, and thus cannot be generalized.

| | 1998 | 1999 | 2000 | 2001 | 2002 |
|---|---|---|---|---|---|
| Vehicles in 1000 | 9.068 | 9.372 | 9.641 | 10.189 | 10.536 |
| Total failures | 215.961 | 230.514 | 236.962 | 242.362 | 243.167 |
| Total failures per 1000 | 23,81 | 24,59 | 24,58 | 23,79 | 23,08 |
| E/E failures | 109.054 | 120.082 | 128.712 | 135.426 | 138.690 |
| E/E failures in % | 50,50 | 52,09 | 54,32 | 55,88 | 57,03 |

Table 6 General vehicle failures and failures related to the E/E system (Dudenhoeffer, 2004)

In a bottom up calculation, we assume the general rate of errors including mechanical, hardware and software to be on the order of 1 incident per vehicle in the initial phase of production spanning 6 months and some 100.000 units produced. Calculating with of 300€ per incident the total amount lies at 30 millions of Euro. Assuming an equal distribution across mechanical, hardware and software errors, the amount of 10 million Euro representing the fraction of software errors is the target for optimization by means of architecture.

The share of research and development costs amounts to 6.5% of the vehicle value. Even though the actual amount depends on the number of units produced, the percentage is widely constant across the industry differing by 2% between premium and mass manufacturers. (Hansen, 2010) reports 3% to 4% for mass manufacturers and 5% to 6% for premium manufacturers for 2010 and predicts an increase to 6% and 8% respectively for the year 2015 due to the electrification of vehicle engines and increase in infotainment functions. In a conservative estimate the fraction of research and development of electronics and software can be assumed at 30% share of the total research and engineering costs, and may be well above that figure, given that 90% of innovation is generated by electronics and software (Negele, 2006).

Based on the numbers in Table 5 and assuming an E/E share of 30% R&D costs per vehicle are between 500€ and 700€. With 2 million units produced throughout the product line life-cycle, research, development and integration of the E/E system totals at 1 billion to 1.4 billion Euros. The composition of the engineering budget will be treated subsequently.

### 5.1.2 Composition of Engineering Expenses

Calculating based on the figures outlined in section 5.1.1 research and development budget for the E/E system of a single product line is 1 billion to 1.4 billion Euro. We assume the amount spent on pure research, i.e. activities unrelated to a concrete product line development to be negligible compared to specification, development and integration activities and try to validate this figure through a top down and bottom up estimate.

In a top-down calculation, considering a man-year valued at 100.000 Euro, the total budget for design and development of a product line of 1 billion yields 10.000 man-years of labor both at the OEM and the supplier. This does not seem excessive, considering that the time-span from the initial definition until start of production is about 5 years and a maintenance phase including model updates etc. of additional

7 years. Even if the initial concept phases and maintenance is less labor intensive than specification, and serial development and integration are assumed to consume 50 per cent of the overall budget, the total man-power available for the 36 months of serial development and integration is 7.000 engineers both at suppliers and the OEM. Parts and material such as prototypes, test rigs etc. contributing to the total budget have not been considered yet. All in all the figure of 1 to 1.4 billion does not seem implausible.

In a bottom-up calculation, we consider the architectural business cases discussed in chapter 3 and costs for validation and verification reported in interviews: Cost calculations distinguish internal and external development costs, as well as integration costs. The typical costs for electronic component as well as software development and integration lie in the range of 100.000€ and 10.000.000€ per architecture change as listed in Table 7 with a mean value around 2 millions of Euro.

The indicated ranges represent the cost span for different analyzed solutions of a functionally equivalent change.

| **Architecture Change** | **Ext. Dev. in 1.000 €** | **Int. Dev. in 1.000 €** | **Validation in 1.000 €** | **Total in 1.000 €** |
|---|---|---|---|---|
| Integration of two ECUs of low complexity | 600 | 600 | - | 1.200 |
| Integration of two ECUs of medium complexity | 200 – 1200 | 300 – 800 | - | 500 – 2.000 |
| Re-design of a cluster of three ECUs for a new product line | 1000 – 8400 | 200 – 2200 | 650 | 1.850 – 11.200 |
| Re-design of two ECUs of medium/high complexity to a new product line | 900 – 2000 | 700 – 2000 | 100 - 300 | 1.700 – 4.300 |
| Re-partitioning of ECUs without change to communication technology (per ECU) | 300 | 75 | 200 | 500 |
| Re-partitioning of ECUs with change to communication technology (per ECU) | 1000 | 1000 | 200 | 2.200 |
| Implementation of a new function on an existing component | 2.200 | - | - | 2.200 |

Table 7 Development and Integration costs for selected architecture changes

External development costs typically represent the highest part of development, however the relation between internal and external costs varies with the fraction of in-house development for each component. Depending on the kind of change and the number of component modified, total costs of development and integration can reach up to 10 Million Euro, the least expensive changes total on the order of 100.000 euro. Finally the (re-)implementation of a single ECU totals on the order of 1 million up to 5 million euro, depending on complexity.

Internal development costs which account for specification and project management, but also for some extent of (in-house) development do not exceed 2 Million Euro, if in-house development is in fact required, otherwise the costs amount on the order of 100.000 Euro, or 10%-20% of total costs, for the aforementioned specification and coordination tasks.

The integration costs reported in the documented architecture cases are relatively small compared to the remaining costs of development. However in these calculations only the costs directly accountable to an architectural change (predominantly the introduction of a new component) have been considered. The large part of fixed, common costs of integration and validation including sub-system test, vehicle tests was not accounted for at the level of a single architecture change. In the documented architecture changes up to 3 ECUs were involved, either by modification or re-design.

Given a total of 40 ECUs, the costs of re-developing the complete E/E system amounts to merely 100 millions of Euro, which is way below the estimated 1 billion. Nevertheless, the order of magnitude of the estimate can be confirmed through interviews with units responsible for controlling. Concrete numbers are considered sensitive and are not disclosed, however the distribution of engineering expenses in the body and safety and infotainment domain, including driver assistance are depicted in Figure 33.

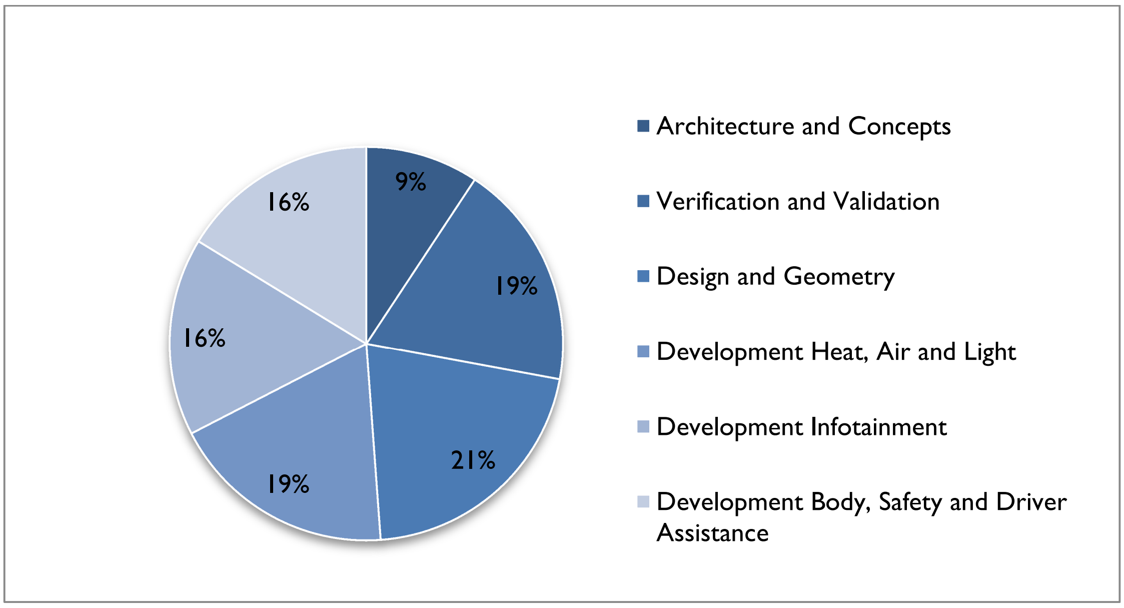


Figure 33 Distribution of engineering expenses in the infotainment and body/safety domain

Total engineering expenses are several times the costs estimated initially in this chapter, based on documented architecture cases. While verification and validation activities account for roughly 20% of the R&D budget in that domain, functional development including component level tests represents roughly 50% of the accounted engineering budget. Considering only the development phase of the lead derivative (i.e. the first model of the product line), the distribution is slightly different, with conceptual design and verification and validation consuming roughly 40% of the engineering budget.

The gap actual of development costs compared to estimates based on documented architecture changes can be attributed to two factors. The development of a product line is stretched throughout several years with individual derivatives being released on average once every year. The re-occurring adaptation of the E/E architecture towards specific derivatives accounts to some extent for the high level of development costs not accounted for in the initial estimates. Secondly, releases of individual derivatives are used to perform cost down measures in order to reduce unit costs without functional changes to the overall system. In total roughly half of the development effort is spent on functional design, whereas the other half is consumed on cost-down and maintenance measures.

All in all the order of magnitude of 1-2 billion Euro development budget for E/E systems can be confirmed. The main part of the effort however cannot be attributed to development, but rather to maintenance and adaptation for cost-down measures or derived vehicle models.

## 5.2 Vehicle Related Qualities

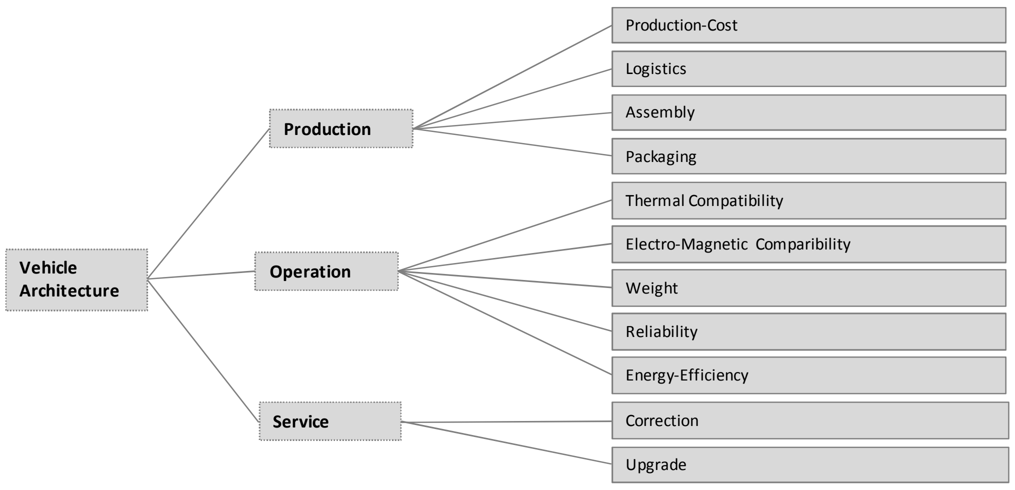


Figure 34 Quality goals at the vehicle architecture Level

The vehicle architecture is an instance of the product line architecture, configured with the individual set of features according to the customer's order. As opposed to the product line architecture, the vehicle architecture exhibits no variability with respect to the functional, logical or physical architecture.

The life-cycle of vehicle E/E architecture is identical from the life-cycle of the vehicle, hence production is associated with vehicle manufacturing, operation describes the usage by the customer and service covers the correction and modification of a concrete vehicle instance. As opposed to the product line architecture, cost of development and the correctness of design and implementation are not subject at the vehicle level.

The quality goal tree of a vehicle architecture instance is depicted in Figure 34. The individual quality goals will be discussed subsequently.

### 5.2.1 Production

Production relates to cost efficiency expressed as the total physical resources required constructing the vehicle E/E system. With respect to costs of production, relevant resources are computational ones such as CPU, memory and storage or infrastructural, i.e. communication infrastructure and power supply. Besides costs in production, the effort and risks involved in assembly and logistics are considered as quality goals in the production process.

#### 5.2.1.1 Costs Efficiency (Electronic Components)

Production costs make up the largest part of the total costs of a vehicle. Concrete figures on component and infrastructure costs are regarded sensitive and are not being published by OEMs. The estimates in section 5 indicate total costs of 4000€ to 5000€ for the exemplary vehicle model. With 10% of that being costs of infrastructure and another 10% accounting for the battery, generator and power distribution, the costs of sensors, actuators and ECUs lie in the order of 3000€ to 4000€.

Overall costs are helpful to point out the economic relevance, but do not necessarily aid in architecture evaluation, as the absolute costs are dictated by provided functionality and justified by vehicle value. For the purpose of architecture analysis the interesting question is, how architectural means aid in providing the same functionality at lower costs or conversely increased cost efficiency.

Taking into consideration just the electronic components of the vehicle E/E system (as opposed to infrastructure), we can derive a cost model from the sensors, actuators and computational resources required by the logical components defined in the functional architecture.

We define $C_{Electronics}$ as the total cost of all electronic components in the E/E system:

$$C_{Electronics} = \sum_{ECU} C_E$$

the costs of an ECU be defined by its sensors, actuators and computational resources, plus a constant term representing housing, power supply etc.:

$$C_{Ecu} = C_{Const} + C_{Res,Ecu} + C_{SenAct,Ecu}$$

With

| | |
|---|---|
| $C_{Const}$ | Constant part of component costs i.e. housing, power supply, connectors etc. The value is assumed independent of the capacity of the ECU. |
| $C_{Res,\ Ecu}$ | Computational resources required by the ECU defined by the number and resource requirements of logical functions partitioned on the respective ECU as software components. |

|  |  |
|---|---|
| $C_{SenAct, Ecu}$ | Sensors and actuators connected directly to the ECU defined by the signal sources and sinks partitioned onto the respective ECU. |

The vehicle configuration consisting of a set of functions *F* is individually defined per vehicle, Its realization as a set of logical components *F* is part of the architecture definition R: $F \rightarrow C_L$. $C_L$ defines the resource requirements to be satisfied by the physical architecture.

Given a distribution of functions in $C_L$ across *n* ECUs described by the inverse allocation function and known costs per logical function for each ECU, the total cost of electronic components can be calculated as:

$$C_{Electronics} = nC_{Const} + \sum_{C_E} C_{Sen\,Act} + \sum_{c\,\epsilon\,C_E} cf_c * |S_c|$$

With

|  |  |
|---|---|
| $C_{Sen, Act}$ | The costs of sensors or actuator either as an individual component or as part of an ECU |
| $S_c$ | The set of software components deployed on the ECU c |
| $cf_c$ | The costs per allocated function specific to an ECU |

The sensors and actuators are defined by the functional requirements and can be assumed independent of the architectural structure. Redundancies in the sensor and actuators are a potential target for optimization but depend on the functional requirements and the concrete configuration of the vehicle.

The remaining costs of electronics depend on the number of ECUs ($nC_{Const}$) thus integration of two or more components yields at least savings on the order of the constant fraction. The constant costs per ECU represent a small part of ECU costs and with increasing computational complexity and functional density of the overall system that fraction will further decrease.

The dominating part of the cost structure is the fraction of computational resources of individual ECUs ($cf_c * |S_c|$). Assuming distinct costs per function provided on each ECU, the optimization strategy resulting from that relationship is to deploy functions on components exhibiting the lowest cost per function. Further assuming a relationship between the functional density and cost per function, the impact of centralization (measured in terms of functional density) on the cost per function can one of the following:

- The costs per function increase with the number of functions deployed on the ECU. The penalty of centralization at some point outweighs the savings of the constant ECU costs.
- Constant costs per function are assumed. The cost per function is equal on each ECU, in which case the partitioning does not make any difference. Total costs are dominated by the constant fraction and the number of ECUs in which case centralization always pays off.

- The costs per additional function decrease with increasing functional density. The savings through integration are increased through savings because of computational resource costs.

(Willutzki, et al., 2006) suggest an increasing cost per function represented by a 'complexity factor'. According to their work the costs of one ECU replacing two or more components is the sum of the separate components times the complexity factor. Regarding physical resources alone, this hypothesis cannot be verified based on empirical evidence. As the subsequent analysis indicates, savings decrease with increasing functional density, but still an integrated component is significantly more cost-efficient than two or more separate components. The highest savings can be realized by integrating several medium complexity ECUs. The complexity factor mentioned by (Willutzki, et al., 2006) mainly reflects increased development costs, which are not considered at the vehicle level.

Costs of electronic components are primarily sensitive to component integration reflected by increased functional density of ECUs. Likewise integration of systems and partitioning of functional components reflected in gateway load has a limited impact on ECU costs. Finally elimination of redundant resources and periphery

**Integration of Components**

The integration of several components into one ECU is beneficial, provided the cost of additional resources on a component due to integration are proportional to the additional number of functions. Within a technologically defined range, such linearity can be assumed.

The actual savings due to integration depend highly on the actual components being integrated. Particularly the functions to be deployed, the extent of sensors, actuators, supplier's portfolio and market conditions reduce the accuracy of estimates based on abstract cost models. Table 8 lists a number of examples of component integration to indicate the range of potential cost savings. The ECUs are classified according to their complexity and functional density into the following capacity classes: Class A: 32Bit processor > 400 MHz, Class B: 32Bit processor > 100 MHz and < 400 MHz Class C: 32 Bit processor < 100 MHz Class D: 16 Bit processor < 100 MHz, Flash and RAM vary and were not used for classification.

| Separate ECUs | Integrated ECU(s) | Separate Costs | Integration Savings |
|---|---|---|---|
| Class B, Class C | Class B | 40€ | 8€ |
| Class B, Class D | Class B | 120€ | 3€ |
| Class B, Class C, Class B | Class B, Class C | 200€ | 13 € |
| Class C, Class D | Class C | 40€ | 8€ |
| Class D, Class D, Class D, Class D | Class B, Class C | 132€ | 25€ |

Table 8 Savings through integration of ECUs

**Integration and homogenization of systems**

Besides application logic a part of the system's resources are used for infrastructure services, particularly gateway functionality. The existence and extent of resources spent on gateways is a result of system structuring and inter-system communication. By integrating several communication buses the resources required for gateway functionality as well as one communication bus interface in the former gateway can be eliminated. This architecture change comes at the price of reducing bandwidth available for future functionality, secondly the physical installation of the infrastructure, possibly negatively affecting costs of wiring.

**Resource Allowance and Redundancy**

Given the set of services to be provided and the resource requirements of their realizing functional architecture, differences between resources required by the logical architecture and resources provided by the physical architecture represent (intended or unintended) redundancies and thus are a target of optimization.

Unintended redundancy in terms of excess computational resources is rare in the automotive domain, as savings through reduction of hardware resources i.e. less advanced processors and less memory are typically identified in the development process. Besides unintended redundancy and unused synergies, redundancy and allowance is a mean to increase reliability or extensibility to future developments. As these are conscious architecture decisions to improve other quality goals, redundancies as a cost factor will be discussed in the respective chapters on extensibility and reliability.

**Optimization to a Reduced Set of Requirements**

Given the variability in equipment and requirements economic gains can be realized by tailoring the hardware resources of components to a more specific (smaller) set of requirements. In the powertrain and chassis domain this strategy is commonplace as different engine type variants have direct impact on the resource intensity of the related E/E functionality. The strategy to optimize the individual configurations comes at the cost of increased variability of components and development costs. The adaptation of resources of the defined vehicle architecture to the level of requirements to is defined as scalability and will be discussed at the product line level.

#### 5.2.1.2 Cost Efficiency (Infrastructure)

Vehicle infrastructure includes the distinct kinds of communication systems and the power infrastructure within the vehicle. The total costs of infrastructure can be calculated based on the number of systems, the number of connections within a system and the physical kind of connections, i.e. the system specific resource types.

For the analysis of infrastructure costs, the power infrastructure can be considered a separate system and assessed alongside with communication buses. The cost of a single system is determined by the number of connections between components attached and their physical installation within the vehicle, affecting their length. The model for infrastructure costs

$$C_{Infra} = \sum_{Sys} \sum_{Conn} (c_{Const,Sys} + c_{Len,Sys} * len_{Conn})$$

With

| | |
|---|---|
| $c_{Const,\,Sys}$ | the constant costs per the connection i.e. costs of transceivers, connectors etc. |
| $c_{Len,\,Sys}$ | the length dependent costs of a connection, i.e. cost of wiring per meter |
| $len_{Conn}$ | the length of a connection |

The costs of a single system are thus the sum of individual connections, while the costs of a connection consist of a constant part and a connection-length dependent part. The individual costs per system are determined by the wiring type. Abstracting from the concrete length of connections and substituting the number of connections within a system with the number of components *n*, the costs per system reduce to:

$$C_{Infra,Sys} = n * c_{Const,Sys} + n * len_{Avg,Sys} * c_{Len,Sys}$$

Unlike costs of hardware resources, costs of infrastructural resources do not exhibit equal scalability with respect to provided resources. The bandwidth of a communication bus once installed can only be reduced or increased through replacement of the complete infrastructure or additional communication buses. This means that the infrastructure of the E/E system can hardly be tailored to the individual requirements configuration as a mean of optimization. Rather the infrastructure needs to account for the whole bandwidth of possible feature configurations for the vehicle and potentially subsequent extensions throughout the production-cycle.

Without optimization to a smaller set of requirements, the remaining architectural means to reduce unit costs of infrastructure are, similarly to costs of electronics:

- centralization by means of component integration, in order to reduce the number of connections

- centralization through system integration, in order to reduce the physical redundancy of wiring within the vehicle as well as the average connection length
- alignment with the geometrical placement of components within the vehicle to reduce costs incurred by the length of individual connections
- substitution of resources to reduce the system-type specific costs of connection and wiring

**Integration of Components**

Integration of components reduces the number of physical connections both in the communication and the power infrastructure. However the integration results in an increase in the lengths of the remaining connections and thus may increase total costs. Documented architecture changes summarized in Table 9 indicate that an integration of components typically leads to a reduction of wiring costs on the order of 1€ to 10€ through elimination of connections, however the savings depend on the topological placement of the involved components. In extreme cases the integration may lead to increase in infrastructure costs.

| Architectural Change | Difference in Costs of Infrastructure | Comment |
|---|---|---|
| Integration of 2 ECUs | -1,2€ | Integration of an actuator into an ECU, savings through elimination of individual connections |
| Integration of 4 ECUs | -12,7€ | Integration of several interconnected ECUs and elimination of individual connections |
| Integration of 2 ECUs | 3,00€ | Integration of two ECUs located in opposite positions of the vehicle leads to substantial extension of remaining connections |

Table 9 Impact of centralization on the costs of wiring (documented architecture cases)

Obviously the effects of component integration cannot be predicted without additional knowledge of the interconnection of the integrated components and the topological placement of the components involved, as both the eliminated parts of infrastructure as well as increased connection length.

**Integration of Systems**

Integration of several independent systems into one can reduce the geometrical redundancy of infrastructure. The downside of integration of communication buses is that available bandwidth is reduced. Typically an integration is beneficial if low speed communication buses are integrated into a higher bandwidth system, to compensate the elimination an independent communication resource. However the integration is limited by physical constraints particularly electromagnetic compatibility limiting the number of participants connected to a communication bus and incurring additional costs for physical components if the limit is exceeded.

Alignment of systems with physical distribution of components

Besides elimination of physical redundancies, the partitioning of components into communication systems can be based on physical installation. With a suitable partitioning of components into systems the length of connections can be significantly reduced, even if their number remains unchanged.

With the current development and integration processes this option is rather hypothetical, as communication buses are structured by domains and driven by application specific, non-functional requirements rather than the physical placement of components. In addition placement of unrelated components onto a common system would increase inter-domain communication requiring additional resources for gateway-activity and increase in distribution of functions. It is common sense that despite potential savings in the wiring harness, systems design follows functional clustering and application domains rather than physical distribution.

**Substitution of resources**

The heterogeneity of available interconnection technology, theoretically allows expensive communication resources to be substituted with less expensive ones or vice versa (Navet, et al., 2005). The possible ways of component interconnections are communication buses with distinct price/quality characteristics and discrete connections. (Kebemou, 2008) lists the following price and quality parameters of common interconnection technologies:

| | CAN | TTCAN | LIN | Byteflight | MOST | TTP/C | FlexRay |
|---|---|---|---|---|---|---|---|
| Bandwidth | 33,3k-1M | 33,3k-1M | 20k | 10M | 25M | 25M | 10M |
| Load in % | 50 | 50 | | 60 | 15 | 80 | 10 |
| Latency | Variable | Determ | Determ | Determ. | Variable | Determ. | Determ. |
| Conn Cost | 1-4€ | 3-4€ | 0,5€-2€ | 1-3€ | >4€ | 3-4€ | >4€[6] |

Table 10 Price and quality of service of communication bus technologies (Kebemou, 2008)

In as an alternative to the aforementioned bus technologies, direct connections can be considered for interconnection at an arbitrary price/quality ratio. These however do not exhibit any scalability with respect to the number of participants and are primarily used for connecting primitive sensors and actuators. Assuming that there are no further restrictions on the choice of communication technology, the least expensive one will be chosen. Typically the less expensive technology provides less bandwidth or less reliable communication. In practice, substitution of resources is constrained by requirements defined at the application layer. Reliability of the transmission medium and timing constraints are only two examples.

[6] lower with on-chip integration

#### 5.2.1.3 Logistics

In order to lower costs of production, OEMs and suppliers transfer the production of electronic components and the infrastructure to low labor cost countries, lowering production costs at the expense of increased costs of logistics. Within the bill of materials, logistics and handling and contribute to roughly 10% of the total part costs (for infrastructure components). The cost of logistics under such conditions is compensated by the savings in cost of labor which make up roughly 20% on the bill-of-materials; or conversely the transfer of production is only justified, if the reduced costs of labor outweigh increased transport costs.

The impact of architecture on logistics is not the incurred additional costs, but rather the time between ordering and delivery. If components can be stockpiled, as it is the case with electronic components extended times of transport can be compensated. If parts are custom-tailored to a concrete vehicle order the production in low labor cost countries increases the time between ordering and delivery. The time from order to delivery is considered customer perceived value and can be monetarized. However the dependency between variability and cost and constraints of logistics is not a linear one but rather depends on class of variability, i.e. whether the infrastructure is individual for each vehicle or remains unchanged throughout the product line.

The architectural mean to lower the dependency of time to delivery and risk of delays, as well as bound capital is to reduce the variability of vehicle architecture both in terms of infrastructure and component variants. In order to improve quality goals measurably in that respect, variability needs to be reduced by an order of scale, read: from vehicle-specific wiring harness to a derivative-specific one.

### 5.2.1.4 Assembly

Assembly of the E/E system consists of installation of components and connection of infrastructure within the vehicle, tests in-line and end-of-line tests as well as potential corrective measures. The total assembly time of a BMW amounts to nearly 8h. The fraction of E/E systems thereof is 1h.

Regarding the cost model for assembly time, each component accounts for a certain amount or time for installation and connection. The infrastructure, i.e. the wiring harness is to be considered separately involving the time to insert the wiring harness into the vehicle.

$$T_{Assembly} = nT_{Ecu} + T_{Infra} + T_{Rework}$$

With

| | |
|---|---|
| *n* | The number of components |
| $T_{Ecu}$ | The time for component assembly, i.e. installation, connection and to the infrastructure |
| $T_{Infra}$ | The time for infrastructure installation, i.e. the placement of the wiring harness in the vehicle |
| $T_{Rework}$ | The time for additional diagnosis and |

Whereas the costs of infrastructure installation cannot be substantially reduced, as the wiring spans the complete vehicle, and needs to be manually placed in the respective installation paths, the number of components directly affects assembly time.

Besides the number of components and connectors to assemble, the risk of errors in assembly of vehicles is to be considered. The risk of misconfiguration or errors in assembly is related to the number of connections and variability of vehicle architecture. Architectural means to lower assembly costs are thus component integration and reduction of variability.

**Component Integration**

Assembly time is sensitive to the total number of components and thus to centralization of architecture. Savings due to reduction of components are moderate in comparison to the cost savings in of electronic components. A mentioned previously a reduction of the number of components by one results in assembly-time savings on the order of 1 minute or 1€ in assembly costs. As centralization positively affects both assembly and part costs it contributes to a positive business case of component integration but does not constitute a trade-off point for system quality.

**Reduction of Variability**

Regarding the risk of assembly errors and misconfiguration of the vehicle E/E system are typically detected either in the in-line test or end-of line test of the vehicle E/E system. The consequence is a rework-cycle of the vehicle meaning extended assembly times. The impact of variability on the probability of potential errors cannot be stated, as other than architectural measures are taken to avoid these.

#### 5.2.1.5 Packaging

The vehicle E/E system is physically installed in the vehicle hence installation space can be regarded as a physical resource consumed in production and required by both electronic components and the infrastructure. Available installation space is scarce and the E/E system is not the highest priority in the geometrical design process a new model, hence it is the vehicle geometry which dictates the constraints of E/E architecture.

Whereas electronic components require installation space of given volume, infrastructure requires routing paths of certain diameter. Both requirements - for installation space and routing paths may prevent architectural choices regarding the integration and interconnection of components.

Ease of packaging is hence in practice not an optimization criterion, but rather a feasibility criterion applied to architecture alternatives. Conversely optimization beyond the constraints of available package space cannot be directly translated into economic gains. However a preservation of existing installation space has a hypothetical value with respect to extension of architecture with components in the maintenance phase.

It is not constructive to postulate a functional relationship between the desired property of package efficiency and measurable structure such as (de-)centralization. The quality goal regarding package space can rather be expressed as a constraint that for each component the associated installation space should exceed the components required space. In the same manner, the geometry defines an upper bound on the diameter of the wires routed along a certain path.

The geometrical concept of a vehicle typically defines a number of installation spaces rather than a few large ones, thus as a rule vehicle geometry favors decentralization of ECUs. For the same reason vehicle geometry favors integration of communication buses and alignment with physical distribution as described in the discussion on cost efficiency of infrastructure.

Regarding the conventional connections, substitution of direct connections with communication buses is beneficial from a packaging perspective. Whereas conventional connections exhibit a direct, linear relationship between the number of connected components and redundancy of infrastructure, since each additional component adds physical connections, and high probability of physical redundancy. Opposed to that, the relationship between components connected to a communication bus and resulting physical redundancy is almost constant.

### 5.2.2 Operation

Operation comprises properties related to the usage of the vehicle by the customer. Quality in use can be expressed in two ways, on the one hand as the level of service attained, including correctness of service provision and preservation of service in case of partial failure, on the other hand the costs of service provision in terms of resources consumed at runtime. Resources in this respect are foremost energy and (indirectly) weight.

#### 5.2.2.1 Thermal Compatibility

Thermal compatibility describes the risk of failures due to overheating of components. Strictly, the quality goal expresses the probability of service failure as a function of the functional density of components. In practice the risk is assessed based on a threshold in required processing capacity. Thus the graded scale is in practice as a feasibility criterion applied to architectural designs, similarly to packaging feasibility. The architectural mean to reduce the risk of thermal interference is to lower the functional density of components. Conversely thermal compatibility defines an upper bound on the functional density and thus on centralization of components in the design of components.

#### 5.2.2.2 Electro-Magnetic Compatibility

Similarly to thermal compatibility, electro-magnetic compatibility describes the risk of service failure due to electro-magnetic interference of components or infrastructure. The risk can be economically expressed in terms of failure rates, but the costs incurred by an electro-magnetic failure are considered prohibitive in architectural trade-offs. In line with thermal compatibility, the risk due to electromagnetic interference is in practice applied as a feasibility criterion on architectural design. This means architecture options are assessed on an Ok/Nok scale based on an informal evaluation of the level of integration and required computational resources. Thus the criterion in practice defines an upper bound on the functional density of components.

Besides decentralization of electronic components, electro-magnetic compatibility must be ensured at the wiring level as well. This likewise defines physical and topological constraints on design and placement of infrastructure:

*Physical constraints* affect the number of components physically connected to a communication bus, i.e. the component density per system. In practice components can be added beyond a certain threshold at additional costs for physical components such as repeater components.

*Topological constraints* prevent parts of the infrastructure to be co-located and require separate topological paths of the wiring harness. In practice this results in a separate wiring topology for the audio- infrastructure and the main wiring harness. This may results in sub-optimal wiring and increased costs of infrastructure as the connection length increases.

Whereas physical constraints cause additional costs and thus reduce the potential savings through system integration, topological constraints are to be considered in the design of topological placement of infrastructure such as thermal compatibility and packaging.

#### 5.2.2.3 Weight

Weight of the E/E system comprises the weight of components as well as the infrastructure. The economic relevance of E/E system weight can be estimated by its impact on fuel consumption. The actual figure is certainly dependent on the motorization and the absolute vehicle weight, but as a rule-of-thumb a reduction of 100kg in vehicle weight results in fuel savings of 0,6l / 100km and a reduction in $CO_2$ emissions of 14g / km.

Given the average weight of the wiring harness 80lbs the potential for savings lies in the order of 0,24l / 100km of fuel and 5,6g / km of $CO_2$. Regarding weight of electronic components, considered in the range of 0.2 kg per ECU, the span for optimization by reducing the total number of components is limited.

This figure gives only a hint of the relevance of weight as a quality goal. Since weight directly affects nominal fuel consumption and $CO_2$ emissions, reductions have significantly higher marketing value than the actual savings in operation by the customer. Taking into consideration legal restrictions on fleet emissions, weight reduction bears even more economic potential. Table 11 illustrates the tax penalties and benefits for vehicles in France (planned as of 2010).

| $CO_2$ Emissions [g/km] | Bonus/Malus [€] |
|---|---|
| **<60** | -5000 |
| **61-90** | -1000 |
| **91-110** | -500 |
| **111-120** | -100 |
| **121-150** | 0 |
| **151-155** | 200 |
| **156-190** | 750 |
| **191-240** | 1600 |
| **240** | 2600 |

Table 11 Incentives and penalties for $CO_2$ emissions

Considering this kind of incentive, each gram of $CO_2$ in the given range is well worth 42€. The value is different for distinct European countries; however the presented figures indicate a similar order of magnitude as the savings in components costs discussed in previous chapters.

| Wiring Harness Part | Weight kg | Weight % |
|---|---|---|
| Cables | 25,90 | 64,27 |
| Tape | 3,10 | 7,69 |

| | | |
|---|---|---|
| Connectors | 2,90 | 7,20 |
| Formers | 2,30 | 5,71 |
| Fuse box | 1,60 | 3,97 |
| Others | 4,50 | 11,17 |
| Total | 40,30 | 100 |

Table 12 Distribution of weight within vehicle infrastructure

The total weight $W_{Total}$ of a vehicle E/E system includes the electronic components as well as the weight of infrastructure. In analogy to the cost model for electronic components and infrastructure we postulate the following relationship:

$$W_{Total} = \sum_{Ecu} W_{Ecu} + \sum_{Sys} \sum_{Conn} W_{Conn}$$

The total weight is the sum of individual component and system weights, whereas the weight of each system is defined by the minimal spanning tree covering the respective communication system. We assume, that a close-to-optimal, redundancy-free wiring within individual systems is possible, thus the total extent of the wiring harness can be expressed as the overlay of minimal spanning trees for each system.

Assuming further component weight $W_{Ecu}$ being largely independent of architectural decisions (particularly the functional density of an ECU) and the weight of connections independent of the kind of communication system, we can rewrite the sum of the physical lengths of individual systems with the help of a redundancy factor:

$$W = n * W_{Ecu} + m * \rho * mintree_{Arch}$$

With

| | |
|---|---|
| $n$ | The number of ECUs |
| $m$ | The number of systems |
| $\rho$ | The degree of physical redundancy in the wiring harness |
| $mintree_{Arch}$ | The minimal spanning tree for the complete vehicle architecture. |

Given that relationship, the architectural means to reduce total weight is to reduce the number of components by means of integration, reduce the number of systems systems and to reduce physical redundancies in infrastructure. These means and their respective metrics are in lie with the discussion on infrastructure cost efficiency.

**Integration of Components**

The weight of components is directly affected by their number, since increase in computational resources dose not result in an equal increase in weight. For concrete architectural decisions the benefits of component integration can be directly measured as the difference in weight. For an exemplary analytical analysis a typical value per component can be assumed, documented architecture changes assume between 100g and 300g savings per component.

Besides the reduced weight through elimination of at least one component, is the elimination of infrastructure interconnecting the integrated components positively contributes to the business case. On the other hand the length of connections to sensors and actuators which are typically bound to their physical installation space may be extended through component integration and eventually outweigh the savings.

**Integration of Communication Systems**

In line with the discussion of production costs, weight of infrastructure on the other hand is much more apt to optimization, given the total length of wiring or 2,5km and total weight of 80 pounds for the whole infrastructure. Unlike production costs which are equally determined by the constant costs of a connection as its length, weight is directly affected by connection length, thus the potential for weight savings is larger than that of cost savings.

**Alignment with physical distribution**

Besides the integration of systems, alignment of systems with geometrical placement in order to reduce infrastructure redundancy is, in theory, a possible strategy to reduce infrastructure weight. In practice, the constraints discussed in section 5.2.1.2 on infrastructure costs apply here as well, making the partitioning of components onto systems a limited design decision.

### 5.2.2.4 Energy Efficiency

Whereas weight is directly related to fuel consumption, energy efficiency in operation affects fuel efficiency through consumption of electrical energy. As a rule of thumb 100W reduction in energy consumption of the E/E system results in 0.1 l savings on 100km (EE-VERT Consortium, 2009). This report calculates potential savings based on an average load of the E/E system of 360W. A reduction of that by two thirds would result in savings of 0.24l per 100km and 2,7g $CO_2$ per km. As outlined in section 5.2.2.3 on Weight, the potential savings in fuel consumption provide only a hint of the economic benefits of optimization and the marketing effect of low nominal consumption and compliance with legal standards have significantly higher economic value.

In line with the argument on costs, the goal of architecture optimization is not the overall energy consumption, which is driven by the customer-perceivable functions, but to increase energy efficiency, i.e. to identify more efficient patterns of providing the same level of service. Given a set of function being active at time t $F_t$ and the realization of services by a set of functional components $C_L$ defined by the realization function and the allocation of logical functions onto electronic components we can calculate the amount of electrical power consumed at time t as

$$P_t = \sum_{c \in C_L} P_c$$

The total energy consumed is then

$$E = \int_0^T P_C dt$$

We further define a simple model for the individual energy consumption defined by an energy level $E_0$ consumed in idle mode and energy consumption per executed software component $S_c$.

$$P_C = P_0 + |S_c| P_S$$

With

| | |
|---|---|
| $P_0$ | The energy consumed by the ECU irrespectively of functional load, i.e. consumption in idle mode. |
| $P_S$ | The energy required per function being executed on on the electronic component |
| $S_c$ | The set of software functions being executed on C. |

The optimization target here is the set of active components $C_L$ given a set of functions being active and the individual utilization of components, i.e. the set of functions on each ECU $S_c$. The architectural requirements on service realization and function partitioning to lower overall energy consumption depend on the concrete energy requirements of the involved ECUs and the interdependencies of individual services. In general centralization of functions in order to reduce the number of active components and alignment according to usage patterns are beneficial.

**Component Integration**

Disregarding correlations between the invocations of individual services, the architectural means to optimize energy consumption is to increase the functional density of electronic components by means of integration. With the simple model for energy consumption presented previously, the centralization of services on a few components is beneficial, as the constant part per ECU can be avoided. Similarly to costs of electronic components, each integrated component yields savings on the order of the constant fraction. However the savings in energy consumption can be achieved only with power management at the component level, including a down-scaling of hardware and suspend periphery not used in the component state.

**Alignment with Service Usage**

In practice vehicle services are indeed correlated. Some functions such as driver assistance and driving dynamics are only activated if the vehicle is in motion or depending on the vehicle speed. Others such as window lifting, wipers and air conditioning are activated only on demand. Partitioning all functions realizing services with a high probability of simultaneous execution increases the probability to suspend the remaining E/E system. Thus energy consumption is in fact sensitive to the partitioning of functions onto components, favoring the reduction of inter-component dependencies requiring the activation of additional ECUs.

An analysis of potential savings through usage-oriented partitioning of functions is only possible taking into consideration the correlation at the functional level and the functional dependencies on sensors and actors defined as part of the physical architecture.

### 5.2.2.5 Reliability

Fundamental notions of reliable systems are presented in (Avizienis, et al., 2004). From a customer's perspective, reliability comes closest to the intuitive notion of quality. Its monetary value is the worth of the achieved level of service and confidence in the correctness of operation as perceived by the customer.

The individual value, as what the customer is willing to pay for a certain level of confidence is highly subjective. From the manufacturer's perspective costs of a failure can be expressed by defection costs, i.e. the monetized effect of failures on subsequent purchase decisions of the customer, or as a requirement to be met by the system under construction. In that case the additional costs required to achieve a required level of reliability such as redundancy or repartitioning can be evaluated.

Structural metrics relevant to the number of development errors i.e. correctness and resulting consequences will be discussed at the product line level along with development-oriented quality goals. At the vehicle level, the goals and investigated metrics to increase reliability aim at preserving the level of service in presence of errors.

(Willutzki, 2006) defines the failures per million units and year (*fpm/a*) and the mean time to failure of a service (*$MTTF_S$*) as the reliability measure to optimize for. Both metrics are interchangeable by the relationship:

$$\frac{fpm}{a} \cong \frac{T * 10^6}{MTTF_S}$$

With

| | |
|---|---|
| *T* | The operation time per year in hours; typical values are on the order of 10.000 |
| *$MTTF_S$* | The mean time to failure in hours |

*$MTTF_S$* is calculated as the integral of the reliability function of the service over time.

$$MTTF_S = \int_0^\infty R_S(t)dt$$

For a service provided by *n* components without further redundancy, *$R_S(t)$* is

$$R_S(t) = \prod_{i=0}^{n} R_i(t)$$

With *$R_i(t)$* being the reliability function for each individual component. For a redundant realization by n parallel components, the reliability function is

$$R_S(t) = 1 - \prod_{i=0}^{n} (1 - R_i(t))$$

With the presented relationships and simplifying assumptions (Willutzki, 2006) derives an fpm per year value depending on the number of components providing the given service *n*, the distance between the involved components *d* and the redundancy of communication *b*:

$$\frac{fpm_S}{a} = 10^6 T\lambda \left[ n + \frac{1}{\ln(b) + \gamma} ((d-1)\alpha^{GW} + d\alpha^{COM}) \right]$$

With

| | |
|---|---|
| $\lambda$ | The failure rate of electronic components, typically assumed values are on the order of $10^2$ to $10^3$ failures per $10^9$ hours. |
| *n* | The number of components participating in the provision of the functions |
| *b* | The number of links of the communication network to eliminate to prevent communication. |
| *d* | The distance, i.e. the number of communication links in the event chain |
| $\gamma$ | The Euler-Mascheroni-Constant (0.57721) |
| $\alpha^{GW}$ | Reliability factor of involved gateways, relative to $\lambda$, a value of 2 means twice as many expected failures in time in gateways than in electronic components. |
| $\alpha^{COM}$ | Reliability factor of communication links |

Obviously the distance, i.e. the number of gateways and the number of components linearly affect reliability of service provision, whereas additional redundancy has a positive, yet significantly smaller impact on reliability. An exemplary calculation for a number of functions is presented in

| Function | $n_{red}$ | $d_{avr}$ | b | $ppm_{eff}$ / a |
|---|---|---|---|---|
| EMF | 3 | 2 | 1.5 | 998 |
| EPS | 5 | 2 | 1.2 | 1447 |
| KAFAS | 5.5 | 2 | 1 | 1590 |
| AHL | 5 | 2 | 1 | 1490 |
| EHB | 6 | 2 | 2 | 1540 |
| RDC | 3 | 1.5 | 1 | 943 |
| Wiper | 3 | 2 | 1 | 1090 |

Table 13.

| Function | $n_{red}$ | $d_{avr}$ | b | $ppm_{eff}$ / a |
|---|---|---|---|---|
| EMF | 3 | 2 | 1.5 | 998 |
| EPS | 5 | 2 | 1.2 | 1447 |
| KAFAS | 5.5 | 2 | 1 | 1590 |
| AHL | 5 | 2 | 1 | 1490 |
| EHB | 6 | 2 | 2 | 1540 |
| RDC | 3 | 1.5 | 1 | 943 |
| Wiper | 3 | 2 | 1 | 1090 |

Table 13 Exemplary reliability calculation for selected services (Willutzki, 2006)

Depending on the number of components involved, the reliability of the presented services varies by up to 50%. The presented figures indicate, reliability in general benefits from a reduction of components and gateways. However the increased risk of development errors resulting from component integration has not been considered at this level.

### 5.2.3 Service

The service phase involves activities related to the concrete vehicle and typically performed by the service organization. In practice the operation and service phases take place simultaneously however due to the distinct approaches of defining the value of quality characteristics are considered separately. The distinction into life-cycle phases in that respect defines the perspective from which quality is assessed rather than a fixed time-span of accounting.

We distinguish the cost and accuracy of diagnosis, cost of error correction, the duration of programming and costs of upgrade as quality goals related to the service of concrete vehicle instances. We consider the feature configuration of the vehicle and quality of component development as fixed and concentrate on the effort of activities related to maintenance and correctness of the outcomes.

#### 5.2.3.1 Vehicle Programming

Vehicle programming describes the installation of software on the vehicle's E/E system. In practice component programming (flashing) is a requirement resulting from the number and complexity of services provided by the vehicle's E/E system and frequency of changes of software components deployed on the ECUs. In recent years the amount of software in the vehicle has grown exponentially. Figure 35 depicts the actual and expected increase in data size and programming times.

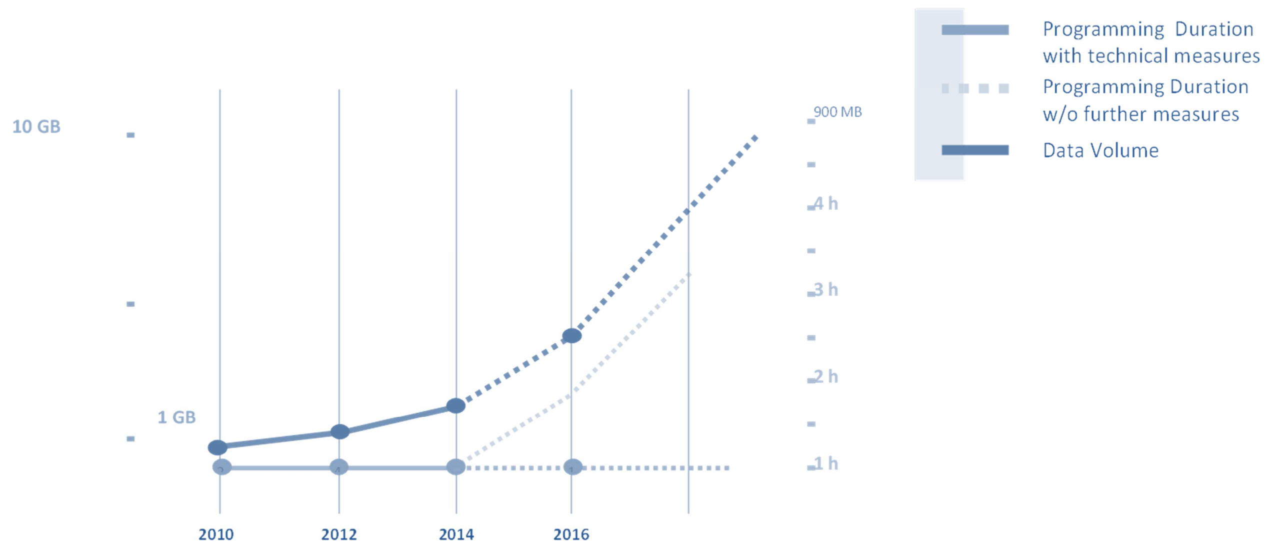


Figure 35 Software size and programming times of vehicle architecture

The economic potential can be assessed by the time required for programming and the frequency of service activities. Driven by the increase in functionality, the amount of software deployed in-vehicle increases exponentially. Table 14 depicts the contribution of distinct domains to the total software size, based on

(Frischkorn, 2007). The emergence of driver assistance functions with a high amount of software is not represented therein, but should not change the disparity between the infotainment domain and the remaining domains significantly.

| | Infotainment | Body | Chassis | Powertrain | Safety |
|---|---|---|---|---|---|
| Software and Data Size | 90% | 2,5% | 5% | 2% | 1,5% |

Table 14 Contribution of application domains to total size of vehicle software

Whereas components with low internal complexity and infrequent updates are not subject to vehicle programming, components hosting OEM specific application logic are required to be updated after delivery. The connection of components to main system buses partially results from the requirement of the component to be re-programmed after delivery. Thus vehicle programming is involved both in correction and upgrade activities. As a default, the software installed in the vehicle is being updated upon each visit in the workshop. The time required for programming is accounted for in the total service time, i.e. the time from reception of the vehicle up to handover of the vehicle back to the customer. Vehicle programming makes up 30% to 50% of total service time.

For an estimate of the economic potential of vehicle programming we assume a total programming time of 60 minutes for the complete vehicle. The duration of programming depends on the number and data intensity of individual functions deployed on a single ECU, as well as the bandwidth of the connection between the access point for programming and the respective ECU.

The time of vehicle programming can be calculated given the amount of data and the speed of the data access, i.e. the net bandwidth of system buses. In practice, vehicle programming can be parallelized system-wise and component wise. This means each system bus can be flashed simultaneously, while on each system bus, several components can be programmed in parallel. The number of channels on each system bus depends on the bandwidth and protocol, i.e on the type of system.

Taking into account the possibility of parallel programming, the total time can be derived from time of preparation, time for postprocessing and the longest individual system to be flashed:

$$T_{Flash} = T_{Prep} + max \cup_{Sys} \left(T_{Flash,Sys}\right) + T_{Post}$$

With

| | |
|---|---|
| $T_{Prep}$ | Time for preparation, i.e. flashing of the gateway or gateways. |
| $T_{Flash,Sys}$ | The individual duration to flash a single system, dependent on the system's bandwidth, number of components and amount and distribution of data to be |

| | |
|---|---|
| | flashed. |
| $T_{Post}$ | Time for postprocessing, mainly verification |

Both $T_{Prep}$ and $T_{Post}$ contribute to the critical path of vehicle programming as these activities cannot be parallelized. The total time is dominated by the programming time of the individual systems. If the amount of data is unequally distributed among the individual systems, the target of optimization in that equation is the most time consuming system to be programmed, as it constitutes the critical path of vehicle programming. Figure 36 depicts an exemplary schedule for vehicle programming. The infotainment system dominates the overall flashing time, thus optimizations on other systems should not lower the total programming time.

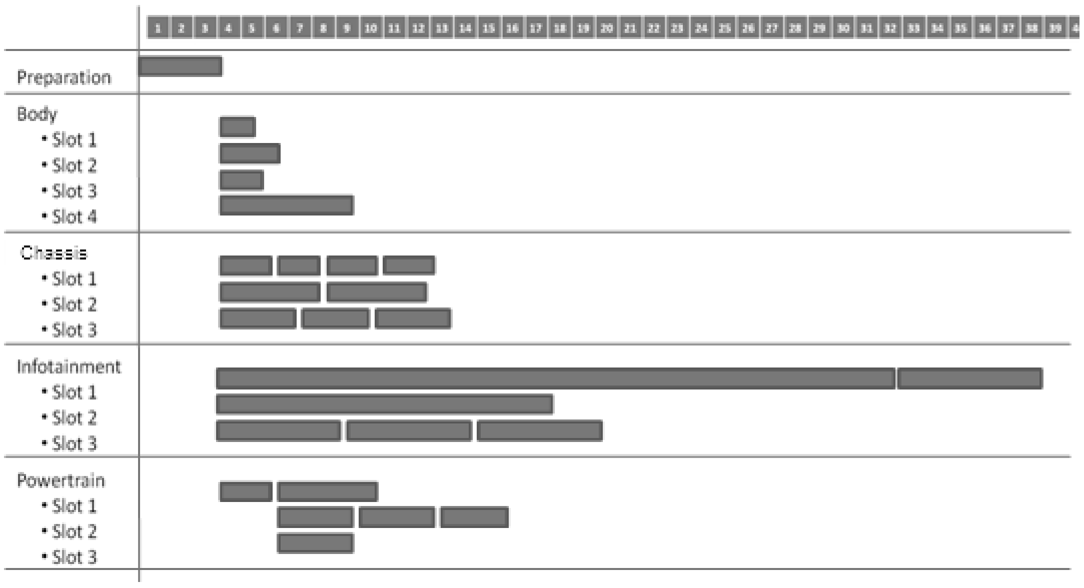


Figure 36 Exemplary schedule of vehicle programming

If the infotainment domain was not dominating in terms of programming time, in general un-balanced allocation of logic and data would extend the critical path of programming. In addition integration of components would reduce the ability parallelization potentially resulting in long vehicle programming. Once again this would impose an upper bound on the possible centralization of architecture.

#### 5.2.3.2 Diagnosis

The cost and risk of diagnosis activities involves two aspects, the effort of error diagnosis by service engineers and the reliability of diagnosis, i.e. the risk of not identifying the root cause for the malfunction. Whereas the first aspect can be expressed as an average time to detect an error, the latter describes the probability of replacing the wrong component thus incurring unnecessary costs. (Tindell, et al., 2003) report on a study stating that more than 50% of the supposedly failed electronic modules removed from vehicles and returned to the supplier, passed bench tests. The risk of replacing the wrong component if it manifests itself, in fact doubles the costs of error correction.

Intuitively, the effort used for diagnosis consists of a constant time, irrespective of the kind of error, for setup and reading of the error logs recorded within the vehicle and a fraction dependent on the complexity of the function. The complexity dependent part is related to the number of participating components and buses, as each needs to be checked for a possible malfunction.

$$T_{Diag}(f) = T_{Const} + T_{Anaysis} * |C|$$

With

| | |
|---|---|
| f | The malfunctioning function and subject of diagnosis |
| $T_{Const}$ | The constant time of diagnosis, for setup and reading internal error logs |
| $T_{Analysis}$ | The time to analyze errors |
| *C* | The set of components participating in the provision of f. |

Taking into account, that a majority of error causes is detected by inspecting the error logs recorded by the vehicle or because the symptoms of the error allow a direct identification, the equation should be rewritten as:

$$T_{Diag}(f) = \max(T_{Const}, T_{Analysis} * |C|)$$

And $T_{Const}$ adjusted accordingly to a typical duration of diagnosis activities. With that equation, only errors exceeding a certain complexity threshold result in diagnosis times beyond an assumed average value. Reported times used for the calculation of service costs are 30min.

Analogously the risk of incorrectly identified causes for a function failure is a random variable correlated with structural properties of the E/E system. Assuming a totally random replacement of components upon failure, the probability of picking the correct one is $\frac{1}{n}$ with n being the number of components installed. Assuming at least basic knowledge of the vehicle's E/E system, n can be assumed to be the number of components participating in provision of the service.

Assuming the latter, the probability of replacing the right component is

$$\rho = \begin{cases} \frac{a}{n} \; if \; n > a \\ 1 \; if \; n \leq a \end{cases}$$

With

| | |
|---|---|
| n | The number of components providing the service being diagnosed |
| a | A form factor representing additional knowledge of the error symptoms and potential |

Since *a* is dependent on external factors, the target of optimization is n, making diagnosability sensitive to concentration of services on a few dedicated ECUs, either through physical separations of components responsible for individual functions or through reduction of the total number of components altogether.

#### 5.2.3.3 Correction

As outlined in section 5.1 warranty costs and measures make up 5% of vehicle costs. For the exemplary 1series BMW this adds up to 1327 € or 1712 € for a fully equipped vehicle respectively. In a conservative calculation 30% of those costs, being is the share of E/E system in the overall vehicle costs, can be attributed to E/E failures. The cost potential per vehicle lies in the range of 398€ to 513€. Assuming 60% of all warranty cases to be caused by E/E failures, as indicated by Table 6, the potential per vehicle amounts to 796€ to 1026€ respectively.

The numbers presented in Table 6 cover only failures which led to a vehicle breakdown, most predominantly battery outage. The total number of service incidents including false error messages and minor faults can be assumed to be higher by an order of magnitude. Also the presented figures include all costs related to correction activities. This includes diagnosis, development effort of software, possible recalls and additional cost drivers such as guaranteed mobility services.

At the vehicle level, we do not consider measures to lower the overall number of errors, as design errors are subject to the development and integration process, while hardware errors are assumed to be independent of architectural means. The target of optimization at this level is thus the average costs of correction. We distinguish two kinds of errors: design errors and hardware failures. Whereas design errors are to be corrected by software update, errors due to component failure require replacement of the component in question.

The costs of correcting hardware errors through component replacement are given by the cost of the respective component and the cost of labor. Costs of flashing are included, since vehicle programming is conducted on each service case.

$$C_{Hardware} = C_{comp} + C_{labor} * T_{Replace} + C_{labor} * T_{Flash}$$

The costs of correcting software errors are merely the costs of vehicle programming. Since vehicle programming is conducted with each service case, the actual costs of error correction, assuming a fixed error rate, are transparent to the OEM.

$$C_{Software} = C_{labor} * T_{Flash}$$

The average costs of all error corrections is then

$$C_{Correct} = C_{labor} * T_{Flash} + \kappa(C_{comp} + C_{labor} * T_{Replace})$$

With

| | |
|---|---|
| $C_{Comp}$ | Part costs of a component, dependent on the functional density |
| $C_{labor}$ | Costs of labor, assumed on the order of 1€ per minute |
| $T_{Replace}$ | Time to replace a component, assumed constant for all E/E |
| $T_{Flash}$ | Time for vehicle programming as discussed in 5.2.3.1 |
| $\kappa$ | The fraction of errors to be corrected by component replacement |

The target of optimization is $\kappa$ which is mainly driven by hardware failures, but not completely. One exception are software errors within components which cannot be corrected through vehicle programming due to their connection to low-speed communication systems, thus incurring unnecessary costs. Numbers on cases in which components had to be replaced rather than re-programmed are not being reported and are assumed negligible.

A further target of optimization is the cost of a component to be replaced $C_{Comp}$. As discussed in section 5.2.1.1 part costs depend on the functional density which makes $C_{Comp}$ sensitive to component integration. However there is little evidence, that hardware failure rate increases with component complexity. Thus with increasing functional density of components, the average costs per warranty case should increase, but the actual cases of component replacement decrease.

In summary the average costs of error correction are hardly sensitive to architectural structure. Costs can be lowered by through vehicle programming, but the fraction of design-errors within components not connected to a high speed bus, is low.

### 5.2.3.4 Upgrade

Upgrade activities cover the functional changes to the vehicle's E/E architecture conducted after delivery. Given the number and variability of optional services, any vehicle equipped with less than all available options is a potential target for functional upgrade; however the actual probability of such activities is low and varies depending on the optional service in question.

As indicated in Table 5 the value of optional equipment of a compact class, premium model, represents up to 30% of the value a basic vehicle configuration. In the case of the exemplary 1 series BMW this amounts to 7700€ in 2010. Economic relevance of architecture extensibility can be appraised from the customer's or the manufacturer's perspective.

For the customer the price of a functional upgrade includes the price of additional equipment and the costs of labor for installation. The value of functional extensibility of the vehicle's E/E system can thus be measured as savings of installation costs. For the manufacturer the value of extensibility is not the expected savings on parts and labor with subsequent upgrades, since the customer is charged the additional costs of upgrades.

The value for the OEM results from the potential increase of aftermarket sales, due to lowered price or the mere possibility to perform such an upgrade. The actual effect of reduced upgrade costs on take rates in aftermarket is not well studied, given that part costs dominate the overall price for upgrades, a reduction in costs of labor may not have a significant effect on aftermarket sales. To measurably improve economic key figures, i.e. the share and revenues of the automotive aftermarket, optimization should target upgrades exhibiting prohibitive costs given an initial configuration.

The costs of extension are to be defined with respect to a change in the set of functions provided. The change of physical architecture results from the additional services to be added to a vehicle configuration. As simple cost model for a functional change in the service set of a vehicle architecture is

$$C_{Upgrade}(\Delta \mathrm{F}) = \sum_{\Delta C} C_{Upgrade,Comp} + \sum_{\Delta I} C_{Upgrade,I} + \sum_{\Delta S} C_{Upgrade,S}$$

Where ΔF denotes the set of optional functions to be installed and ΔC, ΔI and ΔS the electronic components, infrastructure and software components required by the ΔF and not provided by the current technical architecture $A_T$. The target of optimization is the cumulated costs of the tuple (ΔC, ΔI, ΔS), short $\Delta A_T$ which is to be minimized.

In order to determine the change in architecture given a change in the services to be provided, we need to investigate the dependency of individual services on the physical architecture. This can be done based on the realization relationships between the feature architecture and their event chain representation and finally the physical and software components, and selecting the elements which are not part of the default configuration.

In the simplest case, none of the additional components is required for instance if the function is controlled by the vehicle's configuration data; in that case $\Delta P$ is empty. In all but trivial cases, to assess the effort of functional upgrades the architecture change vector $\Delta P$ for each production configuration needs to be determined and attributed with part- and labor costs of installation. Regarding labor costs, the following observations can be constituted:

- Infrastructural resources – as defined in section 5.2.1.2 are the most complex to add, as infrastructure is connected to at least two ECUs and physically distributed across the vehicle.
- Components – as described in section 5.2.1.1 are easier to be modified, if components need to be exchanged, provided the infrastructure does not require changes. In case, components are added, the required architecture change vector naturally includes infrastructure changes
- Software components – are the easiest to add, since the addition requires merely the programming of the vehicle. The installation of additional software after delivery however requires either resource allowances or modifications to the remaining architecture i.e. increased production costs.

Whether one or more of the previously mentioned components need to be added or exchanged depends not only on the realization of the concrete function to be added, but also on the interdependencies between services. Resources shared by functions already installed and services to be added positively affect the effort of the functional upgrade. In general, the more resources required by optional services are contained in the basic configuration, the fewer components are contained in the change vectors for additional services making upgrades less expensive. This leads to resource preservation as a mean of improving extensibility.

**Resource Allowance and Variability**

Architectural means to reduce the cost of functional upgrades are resource allowance. The more physical resources are delivered with the basic vehicle configuration, the lower the effort for subsequent upgrades. This however directly contradicts the goals of cost efficiency, prohibiting any resource allowance beyond the level necessary to provide the specified set of functions.

The extent of resource allowance needs to be tailored with respect to distinct kinds of resources. Intuitively the economically justifiable level of allowance is dictated by the costs of extension of the respective kind of resource. Whereas inelastic and resources expensive to install such as communication infrastructure can be installed even with basic configurations, computational resources with lower costs of extension (i.e. components) can remain variable depending on vehicle configuration. Software based features are mostly apt to variability, as their installation requires merely vehicle programming which is being performed per default with a service case.

## 5.3 Product-Line Related Qualities

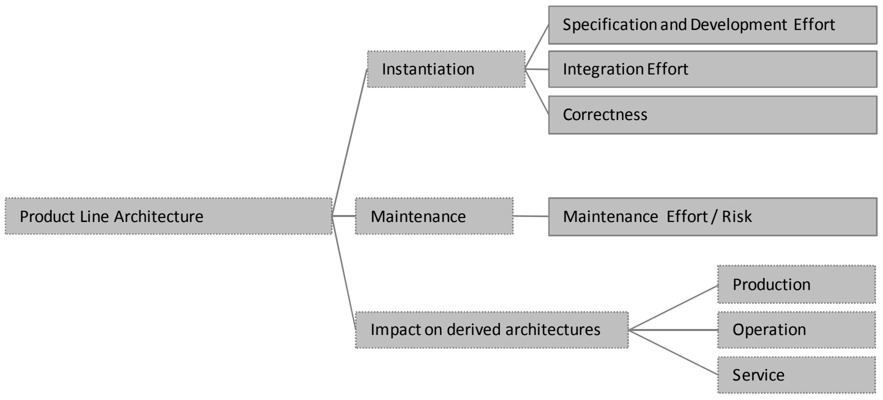

Product Line Architecture
Instantiation
Specification and Development Effort
Integration Effort
Correctness
Maintenance
Maintenance Effort / Risk
Impact on derived architectures
Production
Operation
Service

Figure 37 Quality goals of product line architecture

While the previous chapter investigated the impact of architecture on quality properties measurable at the vehicle-level, we now focus on the properties related to the product line architecture.

At the product line architecture level vehicle related qualities are likewise relevant since the design decisions for a vehicle are inherited from the product line, particularly the adaptability to a reduced set of requirements, i.e. scalability is dictated by product line decisions. Thus centralization and scalability in order to increase efficiency are targeted equally in the design phases. However, in addition to efficiency, development related qualities targeting time, budget and correctness of the system under development are subject of evaluation at the product line level.

Regarding development related quality properties we consider specification and development activities as one since the effort of both activities are aligned and the correctness of developed components results cannot traced to one or another separately. Verification and validation covering all test activities are treated as a single phase. Efficiency and effectiveness of testing activities reflect the major costs and responsibility at the OEM, functioning as a systems integrator rather than component developer. Finally maintenance activities are discussed in separation, as these cover separate, small development cycles constrained by initial architecture decisions.

## 5.3.1 Specification and Development

Specification and development is based on the architecture design artifacts developed in the concept phase, i.e. the *logical architecture* and the *allocation of functions* onto physical components. Given the set of logical functions allocated on each physical component, specification results in a specification document for each ECU used by individual suppliers as a basis for development.

It should be noted that depending on the kind of component the degree of individual development is highly variable. With off-the-shelf components, the specification defines merely a set of parameters or signals. For distinguishing, market-relevant functions the application logic is specified in detail, while for functions reflecting core competencies of the OEM only the hardware and system functions are developed by the supplier.

The quality metrics of specification and development can be measured by the time and budget required to create the respective artifacts and by the correctness of its outcomes. Documented architecture changes indicate a high variance in the estimated costs of changes, ranging from a few hundred thousand of to several millions of Euro. Even for a defined, functionally equivalent change, implementation alternatives differ up to an order of magnitude, depending on whether exiting hard- and software can be reused.

### 5.3.1.1 Specification and Development Effort

Architectures are not being developed from scratch. Rather changes are being introduces as a consequence of changed requirements or added functionality. Given a change in the functional architecture, alternatives of implementation require changes to software, hardware and infrastructure, resulting in varying development effort, risk but also vehicle properties.

Given the details of an architecture change, i.e. the number of soft- and hardware components as well as infrastructure changes, the costs of development can be roughly estimated. We can postulate a cost model for an architecture change of the kind:

$$C_{Change} = C(\Delta A) \ = C_C|\Delta C| + C_S|\Delta S| + \ C_I|\Delta I|$$

With

| | |
|---|---|
| $\Delta A$ | The changes to the existing architecture due to changes in the set of requirements. |
| $\Delta C$ | The set of hardware components which are subject to modification |
| $\Delta S$ | The set of software components which are subject to modification |

| | |
|---|---|
| $\Delta I$ | The set of infrastructure changes, i.e. the number of components allocated to a different system. |

The respective orders of magnitude of $C_C$, $C_S$ and $C_I$ can be approximated based on empirical values derived from documented changes listed in Table 7. In this case

$$C_C \cong 10^6$$

$$C_S \cong 10^5$$

$$C_I \cong \begin{cases} 10^5 \, for\ compatible\ communication\ protocols \\ 10^6 for\ incompatible\ communication\ procotocls \end{cases}$$

Changes to hardware, either because of introduction of alternative physical infrastructure or because of component re-design in order to increase resources, are charged on the order of millions of Euro.

As the foregoing discussion shows, external effects such as necessity of re-implementation and the functional size of additional features dominate the costs rather than architectural structure, i.e. the number of dependencies or functional density. Approximations based on structure of the changes are significantly less accurate than scenario specific cost estimates, since the information on synergies in existing functionality and new requirements are not contained in the architecture model.

More importantly, more fundamental structural changes than actually required by the change in requirements may be necessitated by economic considerations such as savings in production, or the lack of physical resources. Architecture-inherent factors such as the extent of communication between allocated functions, functional density or heterogeneity of functions allocated are then outweighed by external factors.

We focus on factors affecting the extent of functionality to be implemented through reuse and through redundancy of implementation. Reuse is reflected by communality of architecture, while redundancy of implementation is affected by the number of component variants. We should expect that while component variants increase the costs of development as opposed to a single, common component, re-use through adaptation of existing components to the set of requirements at hand decrease effort of development.

**Reuse**

A component designed and developed in a distinct product line can be re-used or adapted in the product line under development with potentially less effort than if it was designed from scratch. Table 15 illustrates the estimates on development costs of a complete re-implementation and partial re-use of components. Derivation from existing ECUs cots roughly 20% - 25% of the initial development, however in two documented cases, a derivation from an existing component is considerably more expensive than a re-design, due to incompatible requirements.

Cases in which components could be re-used at no additional development cost have not been documented. Such cases definitely do occur, if a re-use with no functional changes is possible and a re-implementation does not promise sufficient savings in production costs, alternatives involving re-implementation are not being evaluated and documented at all.

| | Re-Implementation | Derivation from Derivation from Product Line I | Derivation from Derivation from Product Line II |
|---|---|---|---|
| Architecture Case A | 500.000 € | 1.000.000 € | - |
| Architecture Case B | 5.000.000 € - 7.000.000 € | 1.500.000 € | 9.000.000 € |
| Architecture Case C | 4.000.000 € - 9.000.000 € | 900.000 € | - |

Table 15 Documented estimates for re-development and reuse

The decision, whether to reuse an existing component, depends on the compatibility of the individual requirements of product lines and the compatibility of the product line architectures, but likewise on possible unit-cost gains through a re-implementation.  Whereas compatibility of requirements is extrinsic to architecture development, the compatibility of the remaining architecture is a question of decisions made at the architectural blueprint level.

Factors at the architectural blueprint level can be subsumed as architectural differences: incompatibility of infrastructure and incompatibility of interfaces due to distinct allocation of functions. We will focus on such architectural differences will be considered in the subsequent chapter when discussing communality of architectures as a criterion for the architectural blueprint.

Without further specification of the difference in requirements and the resulting savings in production costs, general statements are not possible. But the knowledge regarding a) the variability of requirements and b) the compatibility of product lines can be captured in the architecture model. Thus architecture modeling per se does not prevent development costs, but it is an enables development in a way that makes both variability and compatibility manageable.

**Component Variants**

Component variants are introduced to increase architecture variability, in order to adapt the architecture a reduced set of requirements and lower resource preservation and thus production costs. Additional variants are a design alternative to separation of components providing optional features. The reasons and benefits of that strategy at the vehicle level will be discussed in chapter 6. For now we consider only development costs of additional component variants. Actual examples of development projects are hard to find, as component variants are introduced as an addition to existing implementations, thus total development costs do not allow a direct comparison with projects where one or more component variants are being developed simultaneously. Developer interviews listed in Table 17 indicate that a component variant can be developed at 20% of the initial development costs.

The total costs of all variants as opposed to a basic component and a number of optional components implementing additional functions are harder to find. One of the documented architecture changes discussed in chapter 3 involved number of optional functions was subject to re-partitioning. Three alternatives were considered, as depicted in Figure 37.

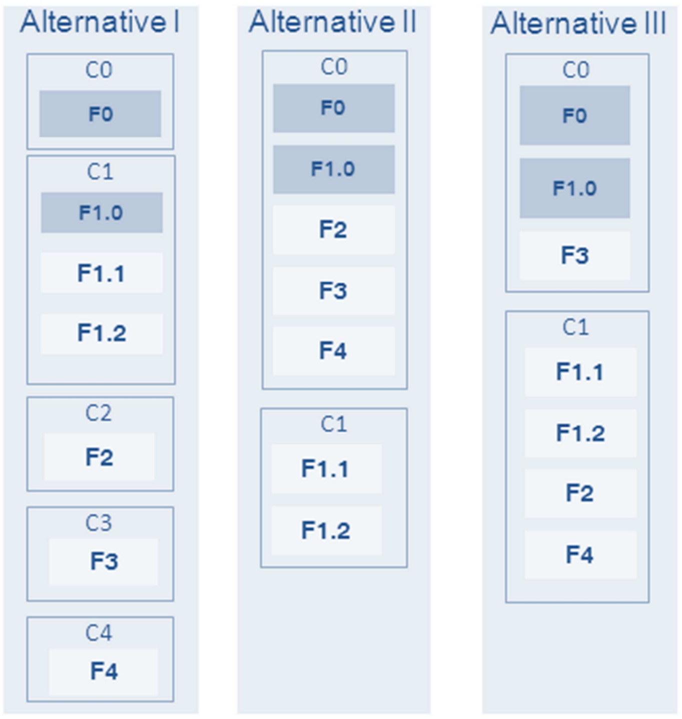


Figure 38 Architecture alternatives involving various component variants

| | Alternative I | Alternative II | Alternative III |
|---|---|---|---|
| **Costs of development** | 3.000.000 € | 18.000.000 € | 11.000.000 € |

Table 16 Costs of development of multiple variants of components[7]

**Alternative I** is the straight forward implementation with the basic functionality allocated on $C_0$ and $C_1$. Two functions allocated on $C_1$ belong to an optional feature, thus the resulting implementation requires four variants of $C_1$.The components $C_2$ through $C_4$ realize independent, optional functions, thus no additional variants are involved.

**Alternative II** proposes a reduced number of components, allocating all optional functions onto $C_0$ and

[7] BMW-Referenz: ARCPS-1002

$C_1$. With two and three optional functions on each ECU, the number of variants increases dramatically. Providing each optional function separately results in 8 variants of $C_0$ and three variants of $C_1$. The development costs of 18 million € (as opposed to the 3 millions in the previous case) are due to the large extent of functionality and hardware changes required by the reimplementation of functions on $C_0$ and $C_1$.

**Alternative III** likewise reduces the number of components to two; however most of the optional functions are allocated on $C_1$ theoretically yielding 15 different variants of $C_1$. Compared to that, $C_0$ comes in two variants. It should be noted, that not each optional function allocated on $C_1$ necessitates a separate component variant. The decision for an independent version of the ECU is a trade-off between expected development and handling costs, potential savings in production costs and expected take rates. With high take-rates and low expected savings, the variant may as well be realized through configuration.

Surprisingly alternative III defining the most component variants is less costly than alternative II. This supports the previous observation that factors not captured in the architecture, such as synergies between functions or the suppliers' portfolio dominate the business case more than architectural aspects.

### 5.3.1.2 Correctness

More than the effort spent on development and integration, the correct functioning of the system under development affects the economics of the architecture life-cycle. Incorrectly functioning components and errors in system behavior makes additional development and test cycles necessary. Secondly correctness ties the development related-aspects of the product line architecture level and the production-related vehicle-architecture level, as development errors are reproduced in mass production. The economic consequences of errors undetected during integration involve analysis and correction activities as well as the correction of already delivered vehicles, not to mention user-related perception of quality. It should be noted however, that the number of errors undetected during system integration, can as well be attributed to low testability of the architecture, thus be evaluated with respect to verification and validation activities.

Regarding development effort, insufficient correctness or maturity of a product line under development may result in additional integration cycles with each additional cycle being valued on the order of $10^5$ Euro. Regarding quality of produced vehicles, simple economic models assume fixed costs per error detected after delivery. In the architecture cases documented in Chapter 3, these costs typically amount on the order of 500€ per vehicle. More sophisticated error models assume fixed costs of correction on the order of 10.000€ to 100.000€, and costs per vehicle affected, whose economic impact depends on the severity from 100€ up to 1000€ for vehicle breakdowns. The perception of value by the customer and the economic aspect of quality for the manufacturer certainly align in this case.

Lack of correctness can be attributed to a number of sources: errors in specification, errors in implementation and failure to discover both kinds of errors in integration testing. Each of these sources of errors can be in turn attributed to deficiency of methods, processes or lack of resources, all of which are independent of architectural choices. For the discussion of architecture quality with respect to correctness we have to assume circumstances unrelated to the product itself, i.e. tools, methods and budget available to be equal for each architectural solution. With all remaining factors being equal, we will try to explain the differences in architecture resulting in differences of correctness.

Based on the foregoing discussion, we identify four aspects of architecture related to the measures taken to improve other – mostly vehicle related – metrics: **functional density**, as a consequence of component integration, **functional coupling** as a consequence of functional segregation and elimination of redundancies and **component variability** as a consequence of adaptation of architecture to a reduced set of requirements.

**Functional Density**

*Hypothesis: Components with higher functional density exhibit more errors in integration than a set of (simpler) components with same functionality.*

To validate that dependency we investigate the number of errors detected during testing and validation of separate ECUs and after these ECUs have been integrated into one component. Two cases of ECUs being integrated to reduce component costs have been documented Figure 39 depicts the integration of four separate ECUs in the body domain (1.1 through 1.4) into two ECUs denoted as 2.1 and 2.2 and a subsequent integration into a single component. The time span considered is roughly three to four years during serial development and integration for product line A and B, and two years of development for product line C.

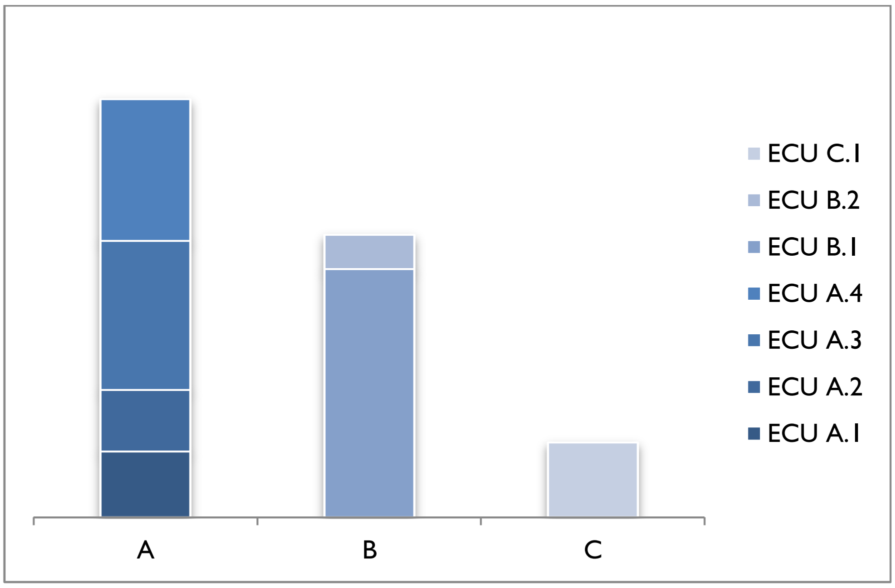


Figure 39 Bugs fixed before and after integration of components

The four original components developed in product line A, if considered separately exhibit a slightly higher number of problems as the two integrated ones developed in product line B, despite potential increase in functionality from one architecture generation to another. In product line C the number of errors detected and fixed drops significantly again. However the errors have been detected and fixed over a shorter time, of two years, as the respective product line is still under development. We can draw the conclusion that each component developed anew introduces a certain level of errors irrespectively of its functional complexity, typically these are errors related to the implementation of system functions and standard behavior – i.e. diagnosis, programming etc. This static set of component errors can be avoided through component integration.

A further observation is that the most problems have been detected in the leading product line denoted with PL A whereas the subsequent product lines re-using these components were considerably more mature (exhibited fewer errors) than the initial one. The re-implementation of functionality has set the number of development errors back to its initial value three vehicle generations before. Nevertheless, based on the presented incidental observations we can consider the hypothesis falsified.

**Functional Dependencies**

*Hypothesis: Components with a high number of external dependencies exhibit more errors in* *integration than functionally equivalent components with fewer dependencies.*

The modularity of components is determined by the degree of communication within a module (cohesion) and with external entities (coupling). A common assumption in literature is that high cohesion and low coupling characterizes a 'good' architecture, which results in low effort and risk of development and maintenance. It seems intuitively clear that communication and know-how exchange within an organizational unit responsible for specification and development of an ECU is more intensive and involves fewer risks, whereas external communication potentially causes errors through miscommunication and inconsistent changes in implementation.

We compare two components in the chassis domain. Both have comparable complexity (measured by the extent of computational resources installed) however ECU A exhibits by far more external dependencies than ECU B. The number of errors detected in integration is depicted in Figure 40.

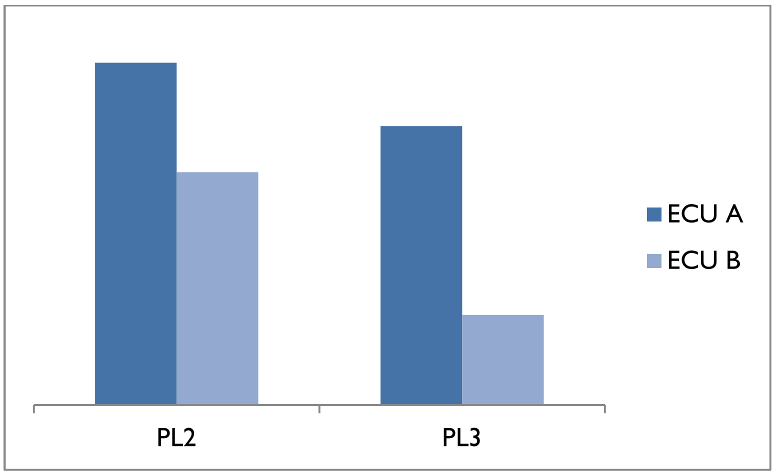


Figure 40 Errors detected in comparable ECUs with distinct number of external dependencies

Certainly the significance and representativeness of this example can be questioned and other factors may affect correctness of the respective components. However lacking component projects of similar complexity and differing number of dependencies it must suffice as anecdotal evidence, and the hypothesis must be validated or falsified through empirical research.

**Reuse and Component Variability**

*Hypothesis 1: Newly developed components exhibit more errors in integration than functionally comparable components which have been derived from existing ones.*

*Hypothesis 2: Components with a higher number of variants exhibit more errors in integration than functionally comparable components with no variants.*

To validate these hypotheses we compare the number of errors in generation changes vs. the number of errors of derived components. To reduce the impact of errors introduced because of the functional change in the respective component, we focus on a component with little functional changes from one generation to another – the sun-roof control. The implemented functionality is stable across product lines and its economic calculation is dominated by unit costs, hence every other generation is subject to a bidder competition. The number of errors in respective re-designs is depicted in Figure 41

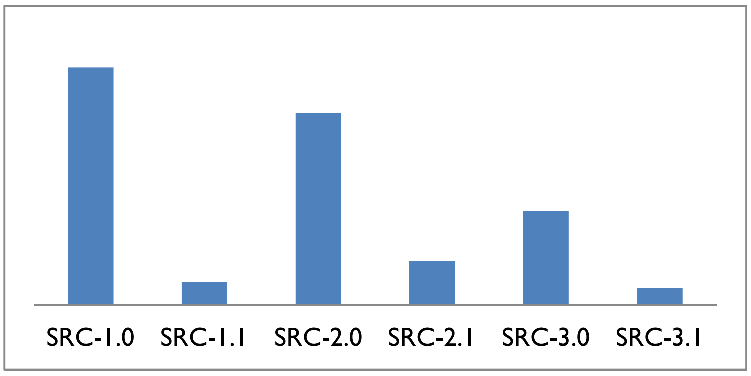


Figure 41 Errors fixed in development upon re-use and re-development

Each change of the supplier results in an increase in errors detected during development, while components being re-used exhibit error rates of 10%-20% of the initial development. This corresponds to the reported development effort for completely re-implemented and derived components.

Figure 39, discussed in the section on functional density, likewise exemplifies this relationship. The number of errors detected in product lines re-using these components is considerably lower than the number of problems detected in the product line architecture they were initially installed. In numbers, the increase of correctness or maturity is about 80%. Particularly errors within system functions such as flashing, diagnosis and wake-up behavior which do not change across distinct variants occur less frequently in derived components.

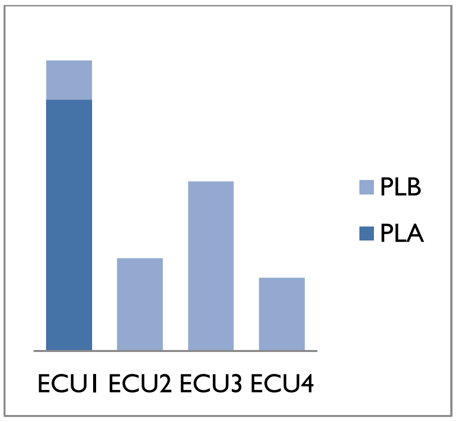


Figure 42 Comparison of errors detected in lead component and derivatives

Another example supporting this hypothesis is a family of ECUs responsible for engine control depicted in Figure 42. The verification of the lead component ECU 1 revealed significantly more errors than the derived ECUs: 2, 3 and 4. The absolute number however does not decrease with each additional variant, as could have been expected, but oscillates at a lower level.

### 5.3.2 Verification and Validation

Verification and validation covers the right branch of the V-Model and can be subdivided into component, subsystem, system and vehicle level test. Considered at component level, the costs of integration are estimated on the order of 100.000€ per newly component. These estimates however represent only the difference in costs of validation if an additional component or functionality is added. The total costs of system integration are significantly higher. As noted initially, 80% of the research and development budget cannot be attributed to the development of a dedicated component, but covers cross-concerns such as prototypes, tooling as well as integration and validation.

In line with the foregoing discussion, the effort for integration to a large extent depends on the extent of functional changes, i.e. newly developed hard- and software components. Fewer and less fundamental changes naturally mean fewer tests to execute and fewer potential errors. Nevertheless even an optimal architecture will not reduce integration effort to zero, as each functional change necessitates validation activities.

As mentioned in chapter 4 discussing the E/E system life-cycle, the verification and validation phase consists of a number integration cycles each spanning tests from the component level to the vehicle level. The cost-drivers in verification and validation are

a) the number of required integration cycles and

b) the effort required at the component ,subsystem, system and vehicle level

The effort required by each integration level depends can be roughly attributed to the architectural structure. Each component involves an initial component test, while each subsystem and system and vehicle test requires a number of test setups depending on the alternative system compositions.

The number of integration cycles in turn depends on the overall implementation planning for the main integration steps, but more importantly on the quality or maturity of the system. While the main integration steps represent the milestones for system functions, basic customer functions and advanced customer functions in the development process, the number intermediate integration steps depends on the number of errors detected in the main integration step which need to be tested again.

#### 5.3.2.1 Verification and Validation Effort

As described in Chapter 4, the integration phase is subdivided into several integration levels with each level spanning the complete integration cycle consisting of a component, sub-system, integration and vehicle test, as depicted in Figure 43.

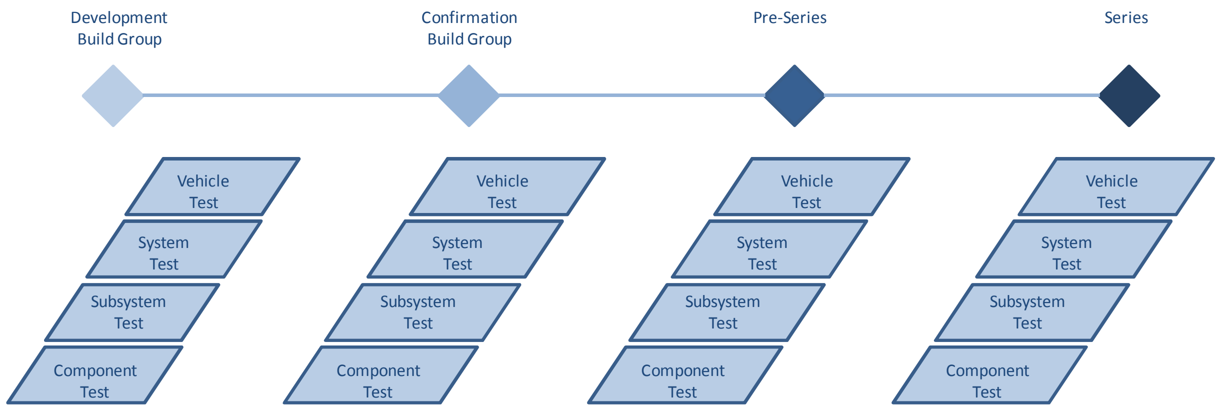


Figure 43 Milestones for verification and validation in product line development

The number of the main I-levels is fixed and dictated by risk consideration for implementation. The implementation planning of system functions, basic customer functions and advanced customer functions is based on these milestones. Thus the number of main I-levels is not a primary target for optimization. However with each main I-level the number of sub I-levels varies depending on the number of errors detected. Hence lack of quality drives the number of test cycles within each I-level, potentially multiplying test effort.

For the all verification and validation activities, the following cost structure can be postulated:

$$C_{Verification\ Validation} = \sum_{I-Level} \left(C_{Test,Comp} + C_{Test,Subs} + C_{Test,Sys} + C_{Test,Vehicle}\right)$$

With

| | |
|---|---|
| $C_{Test,Comp}$ | Component level test effort |
| $C_{Test,Sub}$ | Subsystem level test effort |
| $C_{Test,Sys}$ | System level test effort |
| $C_{Test,Vehicle}$ | Vehicle level test effort |

Cost structures in verification and validation differ for each level hence we need to analyze each level of tests in separation. In the subsequent sections we discuss the impact of central architectural metrics, such as number of components, number of variants and number of systems.

**Component tests**

The effort required to perform components tests is estimated based on the number of components and the functional density of each component. The number of components includes component variants which count as independent components thus each additional variants adds a constant part to overall costs. Costs of testing per component can be assumed proportional to its functional density, however typical business cases assume costs on the order of $10^5$ per component as an approximation throughout the complete development cycle.

With respect to architectural design, the question is how this effort (or inversely: test-coverage at fixed effort) behaves with respect to centralization and the realization of variability through modularity and component variability. With a set of mandatory functions and a single optional function, an integrated component needs to be tested in two versions, while the modular solution involves two independent components to be tested.

With increasing number of optional functions, the theoretical test effort of a modular solution grows linearly, while the effort for an integrated architecture and component variants, the effort grows exponentially. With three optional functions, the theoretical number of variants to be tested is eight, vs. three individual components in a modular setting. In practice however component variants are less test-intensive than individual component, at the expense of test-coverage, test-cases are being executed selectively on individual variants. In line with development effort, we assume effort of 20% per additional variant.

**Subsystem tests**

Analogously the effort of system tests can be estimated by the number of tests to be executed and the number of systems and system configurations in question. Typically each organizational domain i.e. power train, chassis, body and safety, and information and communication, is responsible for testing within its own sub-system, regardless of the physical system structure. This means each domain in tested in isolation, regardless of the number of physical systems and communication buses it contains.

Once again the variability of architecture affects the theoretical number of system configurations to be tested for full test coverage. Theoretically each combination of component variants needs to be tested in a separate system test setup, thus the number of configurations of each system to be tested is the product of the variants / components contained therein.

Modular systems with all optional components installed should behave like configurations with fewer components, as the same side-effects will occur with more participating components as with fewer participants. This assumption cannot be safely made if component variants are involved, as side-effects may be different for each component alternative. Thus for a complete test coverage all variants and combinations of component variants need to be tested in a separate setup.

We conclude that each component variant theoretically multiplies the number of tests at the subsystem level, however in practice the actual effort is traded off against test coverage.

**System and vehicle level tests**

Regarding integration and vehicle tests, the extent of tests is determined by the set of functions requiring validation at vehicle level. The number of functions to be tested depends on the set of functional requirements and is thus not directly affected by architectural decisions.

The extent of testing can be assumed constant per vehicle configuration, as a standardized test suite is executed. The effort affected by architectural structure is driven by the number of vehicle configuration alternatives, i.e. the number of prototypes to be tested, basically following the same patterns as on the system level. The number of possible configurations to be tested in order to achieve full test coverage is grows with the product of all alternative components. Conversely, at a fixed budget for vehicle testing, the test coverage is reduced with each additional, alternative configuration.

#### 5.3.2.2 Test Coverage

Besides the effort required for execution of test-cases and the number test setups, the effectiveness of testing, i.e. the rate of errors detected before delivery is a suitable metric for the quality architecture with respect to verification and validation. Both effectiveness and efficiency are not independent – since budgets for development and testing are widely fixed, a system requiring excessive resources for complete verification has a higher probability of not being tested completely. With decreasing test coverage the probability of errors remaining undetected until delivery will increase.

The economic aspects of an error undetected in the verification and validation phase scale with the number of units produced, in the worst case on the order of millions of vehicles per year. The costs of recalls can easily become astronomical. Calculating with costs on the order of 100 euro per unit and 100.000 vehicles involved, the costs of an individual error may reach tens of millions of euro. Still the actual probability of defects occurring systematically in the field is relatively low.

Despite the risk of errors remaining undiscovered, the number of errors undetected during verification and validation tends to remain low and even decrease without specific measures taken to reduce architectural complexity. Apparently the system tends to stabilize itself this can either be achieved by focusing tests on most common vehicle configurations and most widely used functions. Secondly, in addition to tests during the development phase, selective tests after vehicle production help to discover errors prior to delivery, however at higher cost.  Finally errors occurring in extremely rare vehicle configurations may .

Structural properties of architecture with potential impact on test effectiveness are expectedly the same as the properties negatively affecting testing effort: the most predominant are functional dependencies and system variability. Since data on errors undetected in verification and validation is not representative, these relationships are being validated through interviews documented in chapter 5.3.4

### 5.3.3 Maintenance

If costs of development are primarily driven by the extent of changes to the functional requirements, architecture evaluation needs to focus on modifiability, which is determined by previous decisions and trade-offs with respect to structure.

Changes to an existing architecture occur either as structural changes or as functional upgrades. Analogously to the discussion on development effort and risk, an empirical analysis of the relationship between architectural structure and ability to implement changes is confronted with the problem that the absolute costs of individual changes as well as problems detected in integration and validation are dominated by factors unrelated to architecture. Predominantly changes to the functional requirements or re-development of components.

Abstracting from concrete architectural changes, we need to investigate which architectural patterns make changes more costly or more risky than necessary, or prevent such changes altogether. As Table 7 documents, changes to hardware due to insufficient computational resources are significantly more cost-intensive than changes to software. Likewise interviews with engineers responsible for component development indicate that changes to interfaces between independent units of development are a source of problems in integration. We assume dependencies to increase risk, while scarce, inelastic resources prevent changes are likely to introduce prohibitive costs of architecture changes.

#### 5.3.3.1 Modification Risks

The architectural cause why changes become costly in terms of specification and development and bear the risk of errors introduced can be subsumed as dependencies between independent units of development. A dependency introduces a) an additional necessity for communication in case of changes, and b) the necessity to change units of development (components) initially unrelated to the source of the change. It should be noted however that these consequences are likely to occur only, if the change affects the interfaces realizing the dependency.

The degree of dependency between units of development can be roughly assessed by the number of signals exchanged between them, reflecting the requirement of one component for another to perform a logical operation or provide data. A large number of signals exchanged indicate a higher risk that a change will not remain transparent, but is not a necessary consequence.

The effect of additional dependencies on the total costs of a change is hard to prove statistically, as these are dominated by other factors as outlined in section 5.3.1 . The number of errors is a more reliable indicator of the risks introduced due to changes not communicated or insufficiently specified, but once again such correlations are blurred by factors unrelated to architecture. These factors are predominantly the size and complexity of components which are subject to re-development as well as the development methods and processes.

Documented interviews indicate, that it’s not the existence of dependencies which increases development risk, but rather the frequency of changes to interfaces defined by them. Communication and development required by changes to an existing grows linearly with the number of units of development depending on a given interface.

An analysis of the relationships between the kind of interface defining a dependency and the frequency of changes involves a definition of application semantics and the semantics of interfaces which is beyond the scope of this work. However the centralization of interfaces within gateway components tends to ease modification, due to the additional separation of units of development. Similarly to layered design in network systems, the separation of systems by dedicated gateway-components introduces an additional layer of separation.

#### 5.3.3.2 Possibility of Functional Changes

Excessive dependencies increase the extent of communication and development and introduce the risk of errors or increase costs of modification. The consequences however represent only a small fraction of the overall development effort and risk compared to functional changes. Particularly, if the increase in customer value provided through an intended functional change, or the efficiency gains through structural changes, typically justify both risk and effort.

Worse than additional costs of development and integration, at least from a business perspective, are architectural patterns preventing such changes from being implemented, i.e. patterns leading to prohibitive costs of a functional or structural change. In this discussion we concentrate on resources which can be added only at considerable unit costs or at the cost of re-designing the existing system.

The implementation of any additional function requires resources such as processing power, memory, and storage. Most of these resources can be added to a single component at the cost of re-development, thus all of these resources can be categorized as elastic. However a re-design of a component which is at the edge of capacity can be quite costly, as documented in chapter 3. The re-design of a component’s hardware due to limited capacity runs at ten times the development and integration costs if only the software is affected.

Communication resources and package space on the other hand are inelastic in the sense, that subsequent extension of communication infrastructure in order to provide more bandwidth or changes to the vehicle geometry involve prohibitive costs, at least in the maintenance phase of a product line. In one documented architecture change case, the integration of four ECUs with estimated savings of several millions of Euro has been rejected because the required changes to the vehicle geometry to provide required package space would have incurred greater costs.

The preservation of resources needs to be evaluated against the probability and potential gains of an architecture change, possible changes to the vehicle geometry are rarely anticipated in development given the uncertainty that such changes will occur. However the preservation of package space through the integration of two separate components may be a additional argument for that architectural mean in the evaluation of several alternatives. Likewise connector pins to add periphery components are considered a scarce resource and the availability or preservation of connector pins may be a relevant factor in the evaluation of architecture changes.

Other resources are well being considered at design time. Excess bandwidth is easier to provide for at design time, particularly since infrastructural resources do not scale linearly and do not exhibit a linear relationship between resources provided and costs. In concrete cases, given a CAN bus with an indeterministic media access protocol, a faction of 30% of available bandwidth is preserved at design time for the provision of future extensions.

We can conclude that maintainability and extensibility with respect to inelastic resources is a trade-off taking into account the possible extent and benefit of functional changes and the costs of excess resources provided for at design time. Lack of excess resources, computational, infrastructural and geometrical can be considered a risk and monetarized in terms of lost profit if such a functional change cannot be realized. Such a calculation however depends on the ability to state the probability and extent of required resources at design time, which once again requires a detailed modeling of both requirements and resources in the design phase of architecture.

### 5.3.4 Empirical Validation

Correlations regarding the development effort and correctness of components as well as test coverage and architectural structure are hard to proof, as both are primarily affected by external factors, predominantly functional size and complexity, methodology, processes and so forth. The generality of relationships documented with singular evidence can likewise be questioned. In order to validate the hypotheses mentioned above, a number of engineers responsible for component development and system integration were questioned for their experiences in development and verification and validation. The results are listed in Table 17 and Table 18 an interpretation of the results is given in the subsequent sections.

#### 5.3.4.1 Development Effort and Correctness

The interviews with component developers concentrated on the body and safety as well as the chassis domain. The powertrain and the infotainment domain were not considered due to significant differences of functionality provided by these domains. In the infotainment domain, both development effort and error numbers are dominated by user interface design, while the powertrain is limited to engine and gear train control but more importantly development cycles are organized along engine generations rather than product lines.

The interviews reveal that **additional component variants** are not seen critically by the responsible engineers either. Even though the development costs are reported to increase by 20% on average for each additional variant, it is widely accepted that variants are justified either by sufficient differentiation in vehicle equipment and functional requirements. Likewise none of the interviewed engineers reported a negative impact on quality of the component, rather the maturity of a component variant is considered higher than that of a complete re-implementation.

An increased number of component variants is typically a consequence of centralization of components, with previously optional components are integrated as a module into a mandatory component. **Centralization of components**, particularly in connection with a shared responsibility is reported to increase overall management overhead on the one hand, but on the other hand reduces the number of complexity-independent errors in system functions by reducing the total number of components.

Regarding **functional dependencies** and distribution of functionality, half of the respondents reported negative impact of on correctness, but only if the interfaces were insufficiently specified. Those who objected such a dependency explicitly noted that dependencies are known and interfaces well defined. This leads to the conclusion that it is not the dependencies themselves which pose a risk to development, but rather methodical deficiencies, specifically management of dependencies and specification of interfaces. Notably this methodical deficiency can well be compensated by architecture modeling.

Finally regarding the **distribution of responsibility** in the case of highly integrated ECUs with several suppliers involved, a number of respondents report negative impact on development progress. This was attributed to shifting of development errors from one supplier to another. This dependency is seemingly intensified, if the respective functionality and responsibility is layered and one supplier provides basic system functions while another implements application level functions. This however seems to be a challenge for project management rather than a systematic penalty to development costs.

#### 5.3.4.2 Integration Effort and Test Coverage

The interviews to validate the impact of architecture on system verification and validation activities were conducted with engineers responsible for verification and validation of distinct product lines in various stages of development. In general all respondents report a total effort for verification and validation on the order of $10^6$ Euro. Individual deviation from the mean value of up to 5 million euro were attributed to functional evolution or deviation from the architectural blueprint due to product line specific requirements.

The total effort for verification and validation was related to the total functionality to be tested rather than to architectural structure. **Centralization of components** or systems was not considered relevant to the testing effort by respondents. All respondents agreed that the variability of the system does decrease test coverage, however the negative impact can be compensated by other means. One respondent mentioned that a low maturity of the system under development, i.e. a high number of errors results in increased costs due to additional test cycles. These costs can add up to several hundred thousands of euro, which is negligible compared with the total costs of system integration, or a delay of start of production.

Regarding dependencies the answers were similar to those given by component developers. Not the actual **dependencies** affect correctness, but insufficient specification and dependency management. One respondent reported a negative impact on the preparation of test platforms, due to dependencies making a concurrent upgrade of all components within a test platform necessary.

All respondents agreed on the positive impact of communality and the **derivation of product lines from a common architectural blueprint**. The costs of verification and validation of any new components would be effectively divided by the number of product lines derived from the blueprint. However incompatible evolution of the architectural blueprint and domination of specific requirements of a product line in the evolution was mentioned as a risk by three out of four respondents.

| ECU | Passenger Safety | Parking Assistance | Body Control Unit | Camera Aided Driver Assistance | Sun-Roof Control Unit | Driving Stability Controller | Chassis Controller |
|---|---|---|---|---|---|---|---|
| **Domain** | Body | Body | Body | Body | Body | Chassis | Chassis |
| **Complexity** | medium | simple | high | medium | simple | high | high |
| **Variants** | Depending on sensors and actuators<br>Depending on infrastructure | Variable function partitioning | Depending on sensors and actuators | Depending on vehicle configuration | Depending on vehicle configuration | Depending on vehicle geometry | |
| **Impact of variants on development effort** | - | - | None, same software | 20% of initial development costs | - | 20% of initial development costs | 10% of initial development costs |
| **Further impact on development effort** | - | Distribution of responsibility | Distribution of responsibility | - | - | - | Safety requirements |
| **Impact of variants on correctness** | None | None | | - | None | None | Maturity improves with each variant |
| **Impact of dependencies on correctness** | Negative, errors introduced through insufficient interface definition | None, dependencies known and interfaces well defined | None | Negative, errors introduced through insufficient interface definition (50%) | None, dependencies limited and interfaces well defined | Negative, errors introduced through interface changes | None, already highly interconnect-ted |
| **Further Factors affecting correctness** | Distribution of Functionality | Reimplementation | | Insufficient requirements management | Changes of operating system<br>Reimplementation | Reimplementa-tion | |

Table 17 Factors affecting development activities – summary of Interviews

| **Product Line** | **PL A** | **PL B** | **PL C** | **PL D** |
|---|---|---|---|---|
| **Total Integration Effort (in Eur)** | X*$10^6$ throughout<br><br>validation and verification phase | X*$10^6$ throughout validation and verification phase | - | X*$10^6$ throughout validation and verification phase |
| **Variability of Effort due to Correctness** | None | None | - | Yes, additional integration cycles (negligible compared to overall costs). |
| **Impact of Component Integration** | None | Yes | None | None |
| **Impact of System Integration** | None | Yes | None | None |
| **Impact of Additional Variants** | None | Negative impact on test coverage | None | Negative impact on test coverage |
| **Impact of Component Reuse** | Positive, costs are equally distributed on all product lines | Positive, costs are distributed across all product lines | - | Positive |
| **Further impact on integration effort** | Deviation from common architectural blueprint<br><br>The evolution of the architectural blueprint driven by a single product line (negative). | Change to existing architecture<br><br>Functional interdependencies | Architecture breaches in blueprint evolution | Architecture variability<br><br>Insufficient specification of interfaces and dependencies |

Table 18 Factors affecting verification and validation activities – summary of Interviews

## 5.4 Architectural Blueprint Related Qualities

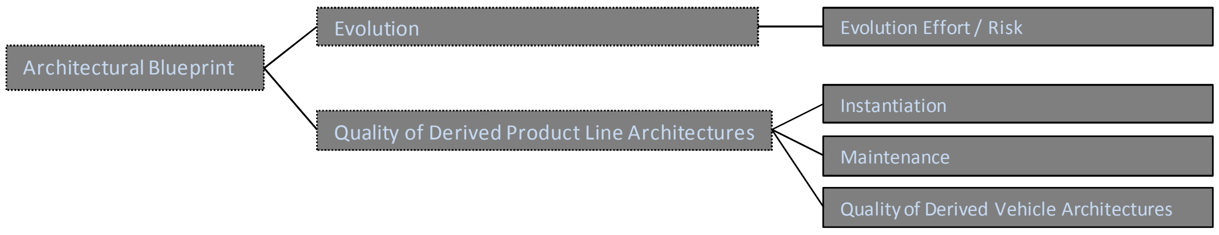

Architectural Blueprint
Evolution
Evolution Effort / Risk
Quality of Derived Product Line Architectures
Instantiation
Maintenance
Quality of Derived Vehicle Architectures

Figure 44 Quality goals at the architectural blueprint level

Analogously to the product line architecture level, we analyze activities in the life-cycle of the architectural blueprint discussed in chapter 4 for desirable quality goals, and identify structural properties which affect the quality of derived product line architectures.

The life-cycle of the architectural blueprint is one of constant evolution and consolidation or as noted by (Axelsson, 2009) one of evolutionary and revolutionary changes. With each derived product line, the functionality provided by the set of all existing product line architectures increases. The variability of existing solutions typically increases with evolutionary changes and decreases with revolutionary changes as the set of rules and common patterns is adjusted to fit the current set of requirements and variability across distinct product lines.

Regarding properties of derived product line architectures, we can describe the architectural blueprint as a set of rules, mandatory for all product lines, defining or prohibiting certain patterns while allowing product line specific optimizations to a certain extent. Intuitively the less variability in architectural structure is allowed by the blueprint, the higher the communality between the individual product line architectures and the higher the potential for re-use. At the same time, efficiency metrics possibly decrease, if optimization towards a specific set of requirements is precluded by the architectural blueprint.

Structural properties such as centralization, modularity and component variability can be likewise identified within the patterns defined at the blueprint level and inherited by derived product lines. If derived product lines do not differ in architectural structure, the observations made in the previous sections can be applied to the patterns dictated by the blueprint as well. In this chapter we concentrate on synergies between product line architectures given a variability of requirements among individual product lines.

### 5.4.1 Instantiation

The interviews in section 5.3.4 indicated that the impact of architectural structure on costs and quality in development is by far not as significant as the impact of reuse. However reuse of components developed in related product lines is subject to economic considerations and can only be considered for a given variability of requirements and concrete affected components.

In general the development of individual product lines is faced with two opposing goals – on the one hand to optimize the E/E system towards a specific set of requirements in order to reduce component costs, on the other hand to reduce development costs and improve maturity of the resulting system through reuse of components. Thus communality with respect to architecture is desirable only where requirements of individual product lines do not differ, while variability of architecture is desirable if the variability of requirements promises sufficient savings in production costs to justify increased development and validation effort.

We describe the differences in requirements at the functional architecture level either as functions available only in specific product lines, or as functions whose specific requirements in each product line differ to the extent that justifies individual development. Intuitively we expect the architectural blueprint to allow the replacement of the one component encapsulating the differing functionality, while re-using the rest of the architecture to increase maturity and lower development costs. Likewise we expect an architecture optimized for re-use to be able to deliver only the functionality and resources required by the specific product line, rather than install excess resources due to a construction kit constraint.

In case of functions available in specific product lines only or exhibiting significant differences across product lines, the architectural condition is to concentrate the respective functionality within a separate component, to allow optional equipment of that functionality. This leads to the requirement for modular architectures providing as described in the previous section architecture scalability. The discussed pros and cons of integrated and modular architectures apply here as well.

In case of functions whose product-line specific requirements allow an adaptation of the resources installed in the respective component to exploit efficiency gains, the prerequisite to maximize re-use is to preserve compatible interfaces between the variable component and the remaining architecture. Regarding outgoing interfaces this implies that functions co-located in one product line should be co-located in other product lines as well, if present. Regarding ingoing interfaces the architectural means are to preserve the system structure, i.e. the allocation of signals onto communication buses should be common in each product line.

The result from these two requirements for communality and local optimization that product line architectures ought to differ only in the presence of functions and presence of components, but not in the allocation of functions to components and components to systems.

### 5.4.2 Evolution

The ability of system architecture to evolve with the set of requirements can be discussed along the same lines as maintainability of product line architecture. In fact the distinction of changes due to requirement evolution and maintenance measures cannot be clearly drawn. If at all, the difference is the scope and time-frame of changes. Whereas maintenance activities apply to a defined product line architecture throughout the time-span of production and maintenance, and are thus limited by the number of vehicles affected by potential savings or value increase in the product line, revolutionary changes occur between architecture generations and are thus more fundamental in scope.

The costs and risks of replacement of communication infrastructure during the maintenance phase of a product-line architecture can be safely assumed prohibitive, with the introduction of a new product line architecture, i.e. a generation change, a low-bandwidth infrastructure can be replaced in order to provide for resources required by future functions. With the replacement of infrastructure technology, say the introduction of FlexRay in place of a high-speed CAN, the resulting architecture becomes incompatible with existing ones, requiring a re-implementation of all components allocated on the respective subsystem.

Likewise the re-design of the hardware platform of an ECU is typically economically infeasible within the maintenance phase of product line, but can be conducted with the introduction of a new product line in order to leverage on decreasing prices for electronic components.

Both kinds of such revolutionary changes resulting in breaches in the compatibility between the predecessor and successor product line cause significant development costs without increasing the customer value of the architecture under development. These changes however may be necessary in order to provide resources for further development or in order to realize production cost savings, but are in general more expensive the less compatible the successor technology. In order to evaluate existing architectural blueprint for its ability of evolution, the expected growth of capacity within the communication infrastructure as well as central components needs to be assessed.

With infrastructural resources, the ability of communication technology to evolve with the required bandwidth depends on the technological choice. Whereas the evolution of domain agnostic standards such as CAN and Ethernet is evolving independently of the automotive domain, domain-specific technology such as MOST or FlexRay, despite providing more bandwidth in their current specifications, but do not evolve independently of the automotive domain. As a consequence more effort needs to be put into the extension of available bandwidth once, the applications demand it, and the prices will not benefit as much from mass production.

Regarding computational resources, embedded processors evolve independently of the automotive domain, thus an increase in capacity can be extrapolated. However the additional costs in the high-end range grow faster than in the low and mid-range. In addition the availability of processors depends on the supplier’s product strategy. Because of this, a high integration of computational resources in centralized components is limited, considering evolution aspects, as evolution of functionality may be limited by unavailability of appropriate hardware.

## 5.5 Summary

In this chapter we have discussed the goals of optimization of E/E system architecture. We have given an estimate of the costs related to E/E system production, development and verification and discussed architectural means affecting these goals at various levels.

We have concluded that at the vehicle level architects strive to optimize for efficiency, while obeying constraints on topology, thermal and electromagnetic compatibility and physical bounds on functional density. The architectural means in order to increase efficiency are integration of components and systems as well as adaptation to a variable set of requirements.

At the product line level on the other hand the goal of optimization is development time and budget as well as correctness of the system. These goals are affected by the total number of errors and the test coverage attainable with the given resources. The means to improve development related goals are predominantly re-use and to a lower extent limitation of variability.

Finally at the architectural blueprint level, the strategic goal of optimization is ease of instantiation and evolution. The derived goal is to maximize reuse while allowing for product-line specific adaptation in order to realize efficiency gains in production, secondly to allow for evolutionary changes to system and component structure. These goals cannot be attained without considering the variability of requirements across product lines and the evolution of functionality.

# 6 Selected Architecture Design Strategies

In the previous chapter we have discussed the quality goals at the vehicle level, product line level and blueprint level and the impact of structural properties of system architecture on the achievement of these goals. This chapter discusses two exemplary strategies to optimize architecture towards individual goals and their effects in the presence of external constraints, such as functional dependencies and variability of requirements

The architectural means in E/E system design are in essence – centralization, introduction of variability and allocation of functions according to specific criteria such as correlation of usage, communication or utilization of physical resources. The centralization of components yields efficiency gains if mandatory components are considered. With the additional constraint of variable vehicle configurations, adaptability of physical resources becomes a complementary goal. Figure 45 presents the relationship between centralization and adaptability.

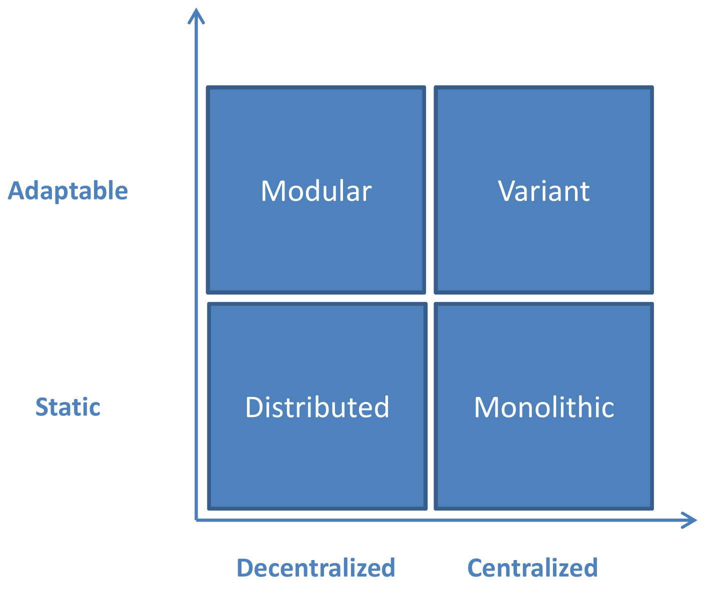


Figure 45 Classification of architecture design strategies - adaptability and centralization

If a static system is considered, the optimization strategy is centralization, resulting in a more or less monolithic system. If adaptability is introduced as a second dimension, the monolithic system requires the introduction of component variants while the distributed systems require modularity through the introduction of optional components.

An architecture strictly adhering to one of the four depicted strategies will rarely occur in practice – geometrical, functional and organizational constraints will result in a hybrid system structure realizing centralization, variability and modularity to a certain degree. Given the variability of requirements and product lines, static architectures are unlikely to become designed in practice, unless in very constrained vehicle designs. In the subsequent discussion we concentrate on the differences between architectures exhibiting modularized and centralized characteristics, as this kind of trade-off is the most common in current designs.

## 6.1 Modularity vs. Centralization

The architectural means to optimize physical resources for different vehicle configurations is to leverage on the variability of optional features and adapt the amount of physical resources to the feature set actually provided. The adaptation of resources aims primarily at costs of production, i.e. costs of electronic components and infrastructure, but depending on the implementation of variability, affects all other aspects of the vehicle and product line life-cycle.

Architecture adaptability can be attained by two strategies: *component variability* and *modularity,* i.e. functional component segregation. The strategy preferred in vehicle E/E architectures so far is component segregation mainly because of clear allocation of organizational responsibility and hence reduced coordination overhead in the development processes. Optional functions are placed on separate electronic components, which can be independently installed in the vehicle E/E system connected through configuration specific, communication infrastructure. This strategy is characterized by a direct allocation of functions to physical components and by a direct relationship between the feature set and the production costs of a vehicle.

The downside of this policy is obvious – the number of components grows with the number of optional functions. Cost savings in low-end configurations come at the expense of inefficiency of vehicle configurations at the high end, both in terms of electronic components as well as infrastructure. Apart from the inefficiency of highly equipped vehicles, the growing interconnectedness of functions renders a segregation of functions more and more difficult. Finally the variability of component configurations introduces an equal variability of the physical infrastructure, i.e. the wiring harness which needs to be manufactured individually for a particular vehicle configuration based on a customer's order.

The direct relationship between the total costs of hardware and infrastructure and the number of components, as well as positive effect of centralization on most quality goals as described in the chapter 6 constitute a direct conflict to architectural scalability by means of component segregation. The combination of benefits of centralization and the goal of adaptability leads to the implementation of variability through component variants. The benefits of adaptability by component variability is the low number of components and low average length of the event chains combined with a direct relationship between production costs and feature configuration.

Strictly speaking a further, trivial kind of adaptability can be identified – the preservation of physical resources and configuration based activation of optional features. Obviously this kind of adaptation does not provide any savings in production or vehicle weight, but allows a simplified architecture and development processes. This kind of adaptability is feasible only, if the potential savings on resources are too small to make up for additional development effort.

Regarding the remaining quality goals, the criterion, whether a specific quality attribute is sensitive to architecture variability, depends on whether there is a significant difference in the respective quality goal between a basic and fully equipped vehicle configuration. Whereas efficiency-related metrics such as costs, weight and energy consumption are affected by an adaptation of architecture, other properties such as logistic effort, packaging or electro-magnetic compatibility do not benefit from architecture adaptation, simply because the vehicle design has to consider a fully equipped functional configuration as the standard case. Still these properties are affected by the consequences of implementation of variability such as variability of infrastructure and components or an increased number of components.

In the following section, we will discuss the quality properties defined in chapter 5 given the additional constraint of variable vehicle configurations. For the following considerations an impact of architecture variability on the expected properties implies a variability of the respective requirement set. In other words, a change in architecture properties is only possible if the associated functions are in fact optional. We discuss the effect of realization of architecture adaptability by stating whether the respective property is in fact sensitive to adaptation, and if so, whether there is a difference in adaptation achieved by component variability or modularity or, if that property does not benefit from adaptation, whether it is negatively affected by component variability or physical segregation and distribution of functionality.

### 6.1.1 Production

Since production costs are the primary target of architecture adaptation we expect a sensitivity of each goal related to production to the architectural means considered. The question is, under which circumstances adaptability through modularity is preferable over centralization and component variability.

#### 6.1.1.1 Cost Efficiency (Electronics)

As outlined in section 5.2 production cost-efficiency is directly related to the number of components installed. Efficiency in production benefits from centralization of functionality up to the point where available technology limits integration. Modular architectures make basic configurations efficient through adaptation of resources but reduce cost efficiency of highly equipped vehicle configurations, because the number of components of highly equipped configurations increases with each optional function. Scalability through component variability on the other hand combines the advantages of a low number of components and adaptation of architecture towards a limited set of requirements. The following hypothetical calculation illustrates the trade-offs:

Given an optional function F to be realized, and a mandatory component $C_0$, the architecture alternatives are to allocate F on a separate, optional component $C_1$ or to make $C_0$ extensible, thus introducing two component variants $C_0$' and $C_0$''. In the first case, the mandatory component $C_0$ provides merely the existing, basic functions, while all resources required for F are provided for by $C_1$. In the latter case $C_0$' requires provision for additional housing, power and circuit board space and the like, in case function F needs to be equipped, however the necessary computational resources and periphery are added on demand thus creating $C_0$''. The costs to extend $C_0$' to $C_0$'' are typically lower than the costs of the separate component $C_1$, as indicated by the discussion on component integration in the previous chapter.

Table 19 documents the average costs of both alternatives, with δ denoting the take-rate of the optional function F:

| Architectural Strategy | Average Costs of Production | | |
|---|---|---|---|
| Modular | $(1-\delta)\ C_0$ | + | $\delta\ (C_0 + C_1)$ |
| Variable | $(1-\delta)\ C_0'$ | + | $\delta\ C_0''$ |

Table 19 Average costs per vehicle with modular and integrated partitioning

We further assume the difference between the basic component $C_0$ and the extensible version $C_0$' be α, whereas the difference between a fully equipped, modular configuration consisting of $C_0$ and $C_1$ and the integrated component providing basic functions as well as F, $C_0$'' to be β. The costs of production of an average configuration depending on the take-rate are depicted in Figure 46. A typical value of α / β of 0.1 is assumed i.e. the additional costs for $C_0$' in the basic configuration are 1€ whereas the difference between the segregated and the integrated configuration if the optional features are equipped are on the order of 10€.

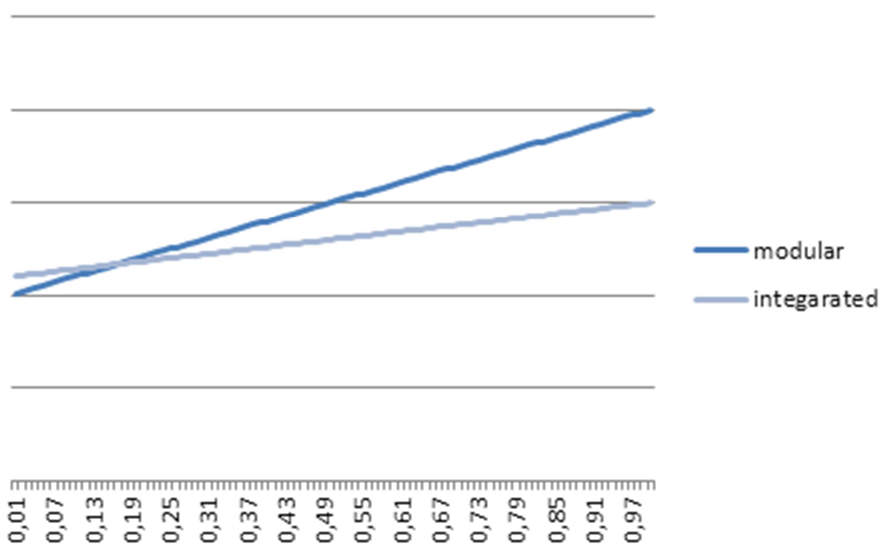


Figure 46 Production costs of Integrated and modular architectures at distinct take-rates (α = 1, β=10)

Figure 46 depicts that, depending on the ratio α/β, the business case for an integrated partitioning turns positive sooner or later. Only if α > β the break-even take-rate is above 50% i.e. the average savings through modularity outweigh the average savings through component integration up to a take rate of 50% and more. Based on documented architecture cases this is rarely the case.

#### 6.1.1.2 Cost Efficiency (Infrastructure)

Regarding the costs of infrastructure, the savings through modularity cannot be put into a simple equation as they depend on factors not captured by the architecture model, such as the geometrical placement; Another reason that the relationship cannot be captured exactly is that the benefits of adaptability blend with benefits of component integration. We can constitute a dependency between the number of components and the cost of infrastructure simply because of the reduced number of connections, thus cost efficiency naturally benefits from adaptability of architecture through component variability.

An additional advantage of centralized architectures over modular ones is that realization of scalability through component variability alone, the infrastructure is independent of the provided feature set. Each vehicle configuration needs to preserve infrastructure for each (possibly not installed) feature. On the other hand integrated architectures positively affect infrastructure cost and weight, by reducing the total number of connections between ECUs irrespective of the actual configuration.

Still, the reduction of connections typically goes along with an increase of connection lengths to between ECUs and actuators and sensors which are bound to a fixed position within the vehicle, a completely centralized architecture with a single ECU integrating all functionality would result in extensive cost and weight of infrastructure.

Still, without additional information on the extent and take-rates of optional functions and vehicle topology, the number of optional components, and the placement of periphery an exact calculation of both alternatives is not possible.

#### 6.1.1.3 Efficiency in Assembly

In the previous chapter we have characterized quality of architecture with respect to the assembly process both in terms of effort, defined as assembly time which is proportional to the number of components to be assembled and in terms of the risk of misconfiguration which is dependent both on the number of connections and variability.

Assuming a number of optional functions allocated on separate components, i.e. a fully modular architecture, the number of physical connections to be performed in assembly depends on the take-rate of optional equipment and grows linearly with the number of components. For an existing product line architecture of the 1 series BMW with 10 mandatory components and 30 optional ones and a typical valuation of component assembly time lies on the order of 1€. The potential savings due to modular designs are thus on the order of 10€ for a fully equipped vehicle.

Compared to that an integrated architecture providing for scalability by component variability exhibit nearly uniform assembly time across all configurations and lower the number of connections due to the reduced number of components. It should be noted that with a significant increase of the number of component variants, the costs of stockpiling and potential misconfiguration may become an issue. Whereas one optional function results in two component variants, the number of variants grows polynomially with the number optional features. This dependency places natural bounds on the number of component variants.

#### 6.1.1.4 Logistics

We have related the impact of architecture on logistics mainly to the variability of the infrastructure, i.e. the wiring harness. With an individual wiring harness for each vehicle configuration, the logistics process, and delivery time is dominated by the duration of shipping.

The time to deliver a vehicle does not vary among vehicle configurations hence logistics is not directly affected by architecture adaptability. However we have constituted a dependency of logistical effort on the variability of infrastructure. A customer-specific wiring harness being directly related to a vehicle configuration requires a complete cycle of production, shipping and assembly between ordering and delivery. Reducing that variability to a small number of variants independent of the configuration would result either in reduced delivery times and more cost efficient production in low cost of labor countries.

Integrated architectures realizing scalability through component variants reduce the variability of the wiring harness, assuming component variants are compatible with respect to wiring and connectors. On the other hand, an increased number of variants negatively affects logistics, due to the need of stockpiling components. Modular architectures on the other hand reduce the number of variants required to be kept in stock, but considerably increase the number of variants in the wiring harness.

Regarding the risk of logistics introduced by unexpected disturbances in the supply chain, architectures implementing scalability through component variability tend to be more resilient than modular ones. Since modular architectures often go along with a single supplier per component, the dependency of the complete production on the functioning of the supply chain poses a major risk to production. With a stop or production being valued on the order of millions of euro per day the substitution of components with variants as is the case with integrated architectures provides some risk mitigation as a side effect.

#### 6.1.1.5 Packaging

Contrary to costs and weight, a consideration of average required packaging space consumed by the E/E system is not reasonable from an economical perspective, as packaging space needs to be provided for a high-end feature configuration in each vehicle or conversely, preserved space does not yield economic gains.

### 6.1.2 Operation

We investigate whether the number of components and segregation of functionality affects resource consumption in operation. In operation, the efficiency goal is primarily energy efficiency and weight efficiency, while the remaining qualities take the form of constraints to be satisfied. The arguments follow along the same lines as on production costs.

#### 6.1.2.1 Thermal and Electro-Magnetic Compatibility

Thermal and electro-magnetic compatibility issues need to be considered for all configurations at design time and thus do not benefit from adaptation of architecture to a reduced requirement set. As outlined in section 5.2 , integration of components is limited by thermal and electromagnetic compatibility, likewise co-location of components and increased numbers of wires affects electro-magnetic compatibility. Still, the negative impact can be handled by other means than architectural structure at additional costs.

#### 6.1.2.2 Weight Efficiency

Adaptation of architecture to a reduced set of requirements applies to increase of weight-efficiency by reduction of average vehicle configurations. With modular architectures the weight of the E/E system is directly related to the number of components and thus the set of functions equipped. The more integrated the architectures, the more this dependency decreases, as the weight of a component is unrelated to its functional density. Integrated architectures exhibit a smaller difference between basic and highly equipped configurations.

Regarding the weight of infrastructure likewise a direct relationship between weight and the set of number of components can be constituted making modular architectures potentially more scalable, however the reduced number of components of integrated architectures improves weight efficiency of electronics and infrastructure across all configurations, as the number of connections remains unchanged regardless of the number of optional features installed.

The relationship is similar to the cost to take-rate diagram depicted in Table 19. Whereas modular architectures exhibit higher efficiency with lower take-rates, integrated architectures involve a minor penalty, but exhibit greater efficiency with highly equipped configurations. In addition the reduced infrastructure of centralized architectures renders integration beneficial in terms of weight efficiency.

#### 6.1.2.3 Energy Efficiency

Energy efficiency, in line with most efficiency metrics, likewise benefits from architecture adaptation towards a reduced set of requirements. The question whether highly equipped configurations do benefit from integrated or modular architectures cannot be answered without concrete configurations and distribution functional dependencies.

As outlined previously adaptation of the E/E system to the usage of functions is possible with centralized architectures as well as modular ones. The possible savings for centralized, variable architectures depend on the relation between energy consumption in ‘idle’ mode and increase of energy consumption with growing computational load. The actual figures depend on internal component design which is rather a technological than an architectural issue.

In the case of a modular architecture a decisive factor is how correlated the functions on individual physical components are and which functional dependencies exist between physical components. In general functional dependencies across electronic control units cause more components to be active for the provision of a small set of functions, however if a clustering of functions and architecture into subsets of architecture is possible due to correlation in usage patterns, the penalty of distribution can be kept within bounds.

#### 6.1.2.4 Reliability

For a given product line architecture, the provision of individual services typically does not differ for individual vehicle configurations. Thus reliability of individual functions is not affected by scalability of architecture however it is well sensitive to the difference in the number of components. The average length of event chains providing a function is correlated with the number of components, but this correlation depends on the actual requirements and implementation of the function. The correlation will be larger if more application logic and signals are shared among distinct functions, whereas isolated functions tend to be less sensitive to the degree of distribution.

### 6.1.3 Service

Efficiency related qualities ins service aim at labor more than total vehicle costs and benefit from modularity rather than centralization. Service related aspects of quality are not sensitive to system adaptation but are affected by side-effects of adaptability.

#### 6.1.3.1 Programming

We have defined quality with respect to vehicle programming issues, to the time to program a vehicle and the probability of errors occurring during programming. We assume the total extent of software and data to be dependent on the application specific requirements only and not on the actual number of components it is deployed onto. As noted in section 5.2 in practice the time for vehicle programming is dominated by a few components in the information and communication domain with a high volume of application data. Optimization in other domains hence does not yield direct monetary gains.

Due to parallelization, efficiency of programming benefits more from segregation of functionality and thus from modular architectures than from integration. Uneven distribution of software and data negatively affects programming time by extending the ‘critical path’ of programming. On the other hand, if the occurrence of programming errors is assumed random for each component, segregation of functionality potentially decreases the number of successful vehicle programming processes.

#### 6.1.3.2 Diagnosis

We have related the effort of diagnosis to the average length of the event chain, concluding that that highly distributed functions negatively affect diagnosis effort and accuracy.  Architecture adaptability by itself does not affect the efficiency of diagnosis; however the side-effect of adaptability by means of modularization, i.e. the increased number of components may in practice negatively affect the average number of components involved in providing a function because of functional interdependencies.

As outlined above this correlation is dependent on the extent of functional interconnection of logical components realizing customer functions reflected by the number of signals exchanged between components. With the increasing interconnection of functions, this dependency can be expected to increase, further negatively affecting diagnosability. Based on these considerations, we can constitute an advantage of variant architectures over modular ones.

#### 6.1.3.3 Correction and Upgrade

Regarding correction activities, we have distinguished hardware failures and software design errors to discuss the ability of distinct architectures. Correction of software design errors of vehicle architecture is covered by the efficiency of vehicle programming. Costs of correction of defective hardware components are directly related to the cost of the component to be replaced, which are lower for modular architectures consisting of a larger number of components with lower individual costs.

Regarding effort of functional upgrades, we have identified distinct costs of upgrade for various kinds of resources. For software provided functions, the implementation of architectural adaptability is transparent, provided the computational resources are being preserved.

For functions requiring additional hardware, modular architectures exhibit lower component costs of upgrade, however their installation possibly implies changes to the infrastructure thus increasing costs of labor. In the case of centralized architectures, the installation of additional resources requires a replacement of the integrated component, hence the higher average component costs of integrated architectures makes upgrades more costly. The increased costs however are compensated by reduced costs of labor, provided the advanced component is compatible with respect to infrastructure.

### 6.1.4 Development and Specification

As discussed in chapter 6, the impact of architectural design in general and centralization and modularity in particular on development and specification activities is limited compared to the functional requirements to be realized.

Whereas distribution of functionality in the presence of functional dependencies between components results in additional external interfaces, centralization of components leads to increased development overhead in the case of distributed responsibility and potentially increased costs because of component variants. There is no evidence that either external interfaces or internal complexity outweigh one another.

Without taking consideration reuse of components and synergies between product lines, we must conclude that development effort is indifferent to structural properties. Re-use of components and the ability of selective adaptation of product lines will be discussed in section 6.1.7 .

### 6.1.5 Verification and Validation

Documented estimates of verification and validation effort at the component level estimate the costs of component testing at 100.000€ per component. The integration of separate components as opposed to a modular design results in savings on that order. In case separate variants of the same component are to be tested each variant accounts at a fraction of an individual component. With a few variants of central control units, the costs of component tests do not vary significantly.

Effort of verification and validation at the subsystem, system and vehicle level is directly related to the number of test cases and the number of test-configurations to be verified. We assume that a completely modular system can be tested in a single configuration, while each component variant doubles the number of possible configurations or reduces the test coverage accordingly.

With current product lines, involving up to three variants of main components such central infotainment unit and driver display, up to six engine variants and up to 30 optional features the costs of verification and validation are on the order of 10 millions of Euro. With each additional variant, the theoretical costs for full test coverage are doubled. In practice the tests are executed selectively on each variant, thus limiting actual costs. With a totally modular system these costs can be kept constant without significant reduction in test coverage resulting in risks with respect to correctness of the system under development.

### 6.1.6 Correctness

In chapter 5.3 we documented anecdotal evidence that a high degree of interconnectedness poses a risk in development resulting in higher error rates. The hypothesis was backed by interviews attributing insufficient documentation and management of dependencies to deficiencies in quality. Nevertheless all sources support the assumption that correctness of the system is positively affected by re-use of existing components rather than structural properties and conversely negatively affected by re-development of components.

Even in the presence of external interfaces posing a risk in development, the actual number of development errors due to additional dependencies either detected in verification and validation or undetected until production cannot be predicted.

With a modular architecture the number of external interfaces increases while the test coverage remains constant, despite increased variability. Conversely with a variant, centralized architecture the external interfaces are limited, while test coverage decreases at constant budgets verification and validation. With neither architectural strategy exhibiting significant structural advantages, the main mean of optimization is synergies through reuse at the architectural blueprint level, linearly reducing development costs and test effort.

### 6.1.7 Instantiation

For a precise calculation of product line instantiation based on an architectural blueprint exhibiting centralized, variable design vs. modular structure, additional assumptions need to be made on the variability of requirements

With a sufficient differentiation of requirements between individual product lines, the instantiation of product line architecture goes along with a re-development of selected components. The costs of such a re-development are related to the complexity of the component under development and whether the component is developed by the same supplier. In that case, the costs can be assumed on the order of another component variant, i.e. 20% of total development costs. If a distinct supplier is chosen, the costs will be on the order of a complete re-development .Based on the interviews documented, we assume the effort and correctness independent of the external dependencies in the case of a modular system.

With these assumptions, centralized, variant architectures should perform worse with respect to architecture instantiation and reuse than modular ones, since the costs of a necessary re-development per component grow with the degree of centralization. Modular architectures promise the realization of product line specific requirements in selected components while preserving stable interfaces.

The selective reuse and adaptation of the architectural blueprint in individual product lines however is only possible through active management of variability and dependence at the level of the architectural blueprint. The modeling of requirement variability and dependencies of customer functions within the architecture enables the efficiency gains of modular architecture.

### 6.1.8 Evolution

We have related the ability of the architectural blueprint to evolve to the scalability of computational and communication resources in line with the functional requirements. This dependency in essence favors a centralized architecture, due to the preservation of communication resources, but limits the degree of centralization by available hardware resources. Taking into consideration development effort, evolution of individual components necessitated by the evolution of requirements and a regular re-implementation of components due to supplier changes, we can constitute a similar relationship as with architecture instantiation:

Highly integrated components result in higher costs of re-implementation and increase the risk of a complete re-design due to limited hardware resources. Conversely modular architectures, allow a selective functional evolution of components, provided external interfaces remain stable.

Analogously to product line architecture instantiation, the degree of communality linearly reduces the costs of development proportionally to the number of product lines. The realization of synergies and efficiency gains however depends on the active management of variability and dependencies.

## 6.2 Summary

We have discussed the impact of architecture on quality properties at the vehicle architecture level and the development process given the constraint of variable vehicle configurations.

We have concluded that centralized architectures implementing adaptability through component variability yield comparable savings at the vehicle level as modular architectures realizing variability through segregation and perform significantly better with higher take-rates of optional equipment. At the same time integrated architectures avoid most of the down-sides of modular architectures which suffer from increased weight and costs in high-end configurations.

To summarize the benefits and penalties of modular vs. centralized architectures, we distinguish between the positive impact of scalability either through component variants or through modularity, and the negative impact through additional variants, or through the increased number of components. Table 20 summarizes the findings.

| **Quality Goal** | **Impact of scalability by component variants** | **Impact of additional component variants** | **Impact of scalability by modularity** | **Impact of increased number of components** |
|---|---|---|---|---|
| Cost Efficiency | | | | |
| Components | + | n/a | + | - |
| Infrastructure | + | n/a | + | - |
| Assembly Efficiency | n/a | n/a | + | - |
| Logistics | n/a | - | - | - |
| Packaging | n/a | n/a | n/a | n/a |
| Thermal and Electromagn. Compat. | n/a | n/a | n/a | n/a |
| Weight | n/a | n/a | + | - |
| Reliability | n/a | n/a | n/a | - |
| Energy Efficiency | + | n/a | + | n/a |
| Vehicle Programming | n/a | n/a | + | n/a |
| Diagnosis | n/a | n/a | n/a | - |
| Correction / Upgrade | - | n/a | + | - |
| Development | n/a | - | n/a | n/a |

| | | | | |
|---|---|---|---|---|
| Verification | n/a | - | n/a | n/a |
| Correctness | n/a | - | n/a | n/a |
| Instantiation | n/a | - | n/a | + |
| Evolution | n/a | - | n/a | + |

Table 20 Impact of architecture variability on vehicle architecture properties

Concrete estimations of consequences of architectural decisions depend on an exact definition of the interdependencies between requirements and individual components. Without a model of the variability of functional requirements and functional interdependencies, the evaluation of architecture alternatives will remain purely qualitative.

Regarding the development and integration activities as well as correctness of product line architecture, we have concluded that both increased component variants and additional interfaces have little impact. The dominating factors affecting effort and risk are changes to the architecture, particularly interfaces and resulting re-implementation of components. Secondly supplier capability and process maturity affect the correctness of the resulting architecture.

Centralization and the resulting increased functional density does not seem have a significant impact on the number of errors in development. However only a few projects could be compared which involved the integration of several components. It must further be noted that integration reduces the possibility of parallel development and thus further reduction of development cycles, will be limited by centralization.

Finally, the relationship between increased variability and costs of development and correctness is not as immediate as with countable entities and their resulting costs. Increased complexity can be handled by other means than architectural structure such as processes, methods and tools. The selective test coverage in the presence of an exponential number of distinct configurations is one such mean.

# 7 Evaluation of Selected Architecture Changes

In the previous chapter we have investigated the impact of structural properties on the quality goals of architecture in isolation. The identified relationships between structure and individual quality goals have been derived either through modeling or by investigation of documented architecture changes. In case where an analysis could not be performed due to lack of documented data, interviews with domain experts were conducted.

In the subsequent chapter we want to validate whether the models and dependencies outlined in the previous chapter lead to more founded and systematic results, or even to different conclusions as the analyses investigated in chapter 3. We present three architecture changes with individual realization alternatives and discuss the alternatives with respect to the quality properties defined in chapter 6.

We try to derive the differences in quality properties from the structural model of the presented architectural alternatives. In cases where a derivation of quality properties based merely on structure is not possible with sufficient accuracy, as is the case with unit or development costs, we will quote estimates made in the process of the evaluation.

## 7.1 Architecture Change A

### 7.1.1 Overview

The presented architecture change has been requested due to the introduction of a new function $F_A$. The function partially relies on functionality provided by existing function $F_B$ which is allocated to component $C_1$. Both functions belong to the body and safety domain.

In the existing design $C_1$ implements a number of basic vehicle functions. The new function to be implemented is optional and a low take-rate of 3% s expected. Thus the omission of resources related to it in the basic vehicle configuration promises cost savings in production. Development and integration investment for each alternative have been estimated by potential suppliers and component developers and are listed in Table 22.

### 7.1.2 Alternatives

Three alternatives have been considered for realization as depicted in Figure 47. To highlight the optional and alternative configurations allocation has been illustrated implicitly, by placing functional components directly within physical components, rather than explicitly expressing realization and allocation relationships using the notation described in Appendix I.

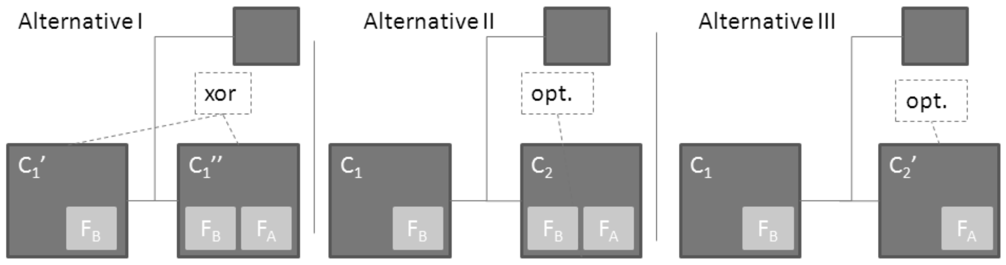


Figure 47 Architecture change A – design alternatives

**Alternative I** involves the implementation of the additional, optional function $F_A$ on the same component $C_1$ as existing function $F_B$, thus introducing two variants $C_1$' and $C_1$''. The high development costs documented in Table 21 are due to the fact that the existing component is at the limit of its capacity and the allocation of $F_A$ would require a re-design of the component's hardware.

**Alternative II** involves the implementation of both $F_A$ and $F_B$ on a separate component $C_2$, to be installed as optional equipment if $F_A$ is ordered. In the basic configuration the mandatory function $F_B$ is to be implemented on component $C_1$ (and disabled by configuration if $C_2$ is installed) resulting in additional (redundant) development costs of $F_B$ and in fact doubled maintenance costs. Nevertheless, considering development costs only, it is the least expensive alternative, mainly because of an existing implementation of $F_A$ at the supplier, which can be extended by $F_B$.

**Alternative III** involves the implementation of a new component $C_2$' providing function $F_A$ only. Due to the high functional interconnection with $F_B$, changes to the existing component $C_1$ are expected. The expected development costs are in between of the remaining alternatives.

| | **Alternative I**<br>[in 1000 €] | **Alternative II**<br>[in 1000 €] | **Alternative III**<br>[in 1000 €] |
|---|---|---|---|
| External Development | 1.200 | 200 | 800 |
| Internal Development | 800 | 400 | 500 |
| **Total** | **2.000** | **600** | **1300** |

Table 21 Architecture change A – estimated development costs

Surprisingly, in this architecture change development costs do not depend on the functional size of the components to be implemented. Alternative II requiring implementation both of $F_A$ and $F_B$ on a completely new component is the least expensive one. Rather the development costs are dominated by the supplier's product portfolio, in that case an existing implementation of $F_A$ on a separate component. Moreover the costs of the alternative which could have been expected most cost-efficient are driven by the necessary re-implementation of existing functionality. This indicates low extensibility of the existing system specifically component $C_1$.

## 7.1.3 Evaluation

The presented architecture change affects the product line architecture. We investigate the impact on derived vehicle architectures and the development related properties as well as the communality, irrespectively of the architectural blueprint.

**Electronic components**

Costs of production cannot be derived exactly from the architecture model in that early stage of development and need to be estimated by potential suppliers. As outlined in chapter 5.2 , component integration leads to cost savings on the order of 1€ to 10€ per component, thus alternative 1 with all functionality integrated can be expected the least expensive one, despite 'extensibility overhead'.

The separate costs of the individual components are listed in Table 22. As could have been expected the integrated component $C_1$'' is more cost efficient, than any combination of two separate components implementing $F_A$ and $F_B$ respectively.

| $C_1$ | $C_1$' | $C_1$'' | $C_2$' | $C_2$ |
|---|---|---|---|---|
| 26,0 € | 26,5 € | 31,5 € | 15,0 € | 21,5 € |

Table 22 Architecture change A – estimated component costs

The 'extensibility surcharge' is estimated at 0.5€ which is moderate, however the take-rate of the optional function is rather low, making the vast majority of configurations in this case more expensive than necessary. Given that we need to calculate the average costs per configuration for each of the three alternatives. The results are depicted in Table 23.

| | Average production costs take-rate = 3% [in €] | Avg. prod. costs take-rate = 30% [in €] |
|---|---|---|
| Alternative I | **26,65** | **28,00** |
| Alternative II | **26,65** | **32,45** |
| Alternative III | **26,45** | **30,50** |

Table 23 Architecture change A – average costs in production per vehicle configuration

It turns out that for low take-rates all three alternatives are alike. However with increasing take-rates the business case for the integrated solution turns significantly better.

**Infrastructure**

The costs of infrastructure depend on the number of components as well as the placement of the individual components within the vehicle. In this particular case, sensors and actuators are placed both in the front and in the back of the vehicle resulting in relatively long connections. The separate components in alternatives II and III are placed next to the mandatory component and alternative placements have not been suggested, making the lengths of the individual connections between the mandatory and optional components insensitive to partitioning decisions.

The evaluation involves merely the number of individual connections. Alternative I defines the least components is thus the most efficient in terms of infrastructure. The cost savings compared to the remaining alternatives were estimated less than 1€.

**Costs of Assembly**

The costs of assembly can be derived from the number of components, adding costs on the order of 1€ per component in the case of installation. In the case of alternative II and III this affects only the vehicles equipped with $F_A$, i.e. the estimated 3%.

Costs of variant handling in alternative I have not been considered in the initial evaluation, however management and handling costs (calculated per separate part id) are typically assumed on the order of 10.000€ per year which is negligible compared with the investment and production costs.

With increasing numbers of optional and alternative features integrated on one component and part numbers growing geometrically with optional or alternative features this could become an issue.

**Logistics**

We assess the impact of architecture on logistical effort by evaluating the impact on component and infrastructure variability.

Alternative II and III both add complexity to the customer specific wiring harness resulting in an increased number of wiring modules. The number of additional modules within the wiring harness depends on the connection with the remaining infrastructure. We can assume each variant and each separate component adding at least one module valued similarly to an individual component, i.e. with costs on the order of 10.000€

per year. The cost structures and time to delivery in general do not vary among the individual alternatives since the customer specific wiring harness is a design decision inherited from the architectural blueprint.

With respect to component variability alternative I defines one additional variant which needs to be kept in stock both in production and in service. However the same holds true for the separate components $C_2$ and $C_2$'. Thus the alternatives can be regarded equivalent.

Regarding the risk of supply chain interruptions, the supplier dependency can be reduced with alternative I, by defining two alternative implementations for the same functionality, which comes at the price of doubled integration and external development costs.

**Package Space**

Alternative I preserves the package space otherwise used by the optional components in alternative II and III. The integration of functionality in $C_1$' and $C_1$'' does not increase the package space required by $C_1$ which could have been the case, if a large number of periphery were integrated into these components.

Regarding package space for the wiring harness is not sensitive to the presented architecture alternatives since the diameter of the main routes of the wiring harness are stable across all three alternatives.

**Thermal and Electro-Magnetic-Compatibility**

The components under evaluation are not highly integrated, thus thermal and electro-magnetic compatibility are not the limiting factor of the functional integration in the case of $C_1$'' in alternative I. Electromagnetic compatibility may be an issue with the co-location of $C_1$ and $C_2$ and $C_2$' respectively, depending on the computational resource intensity of function $F_A$ and $F_B$.

**Weight**

A discussion of the impact of on vehicle weight is trivial in that case, the component integration components saves vehicle weight on the order of 0.3kg. Due to the limited length of the wiring between $C_1$ and $C_2$' and $C_2$'' respectively the differences between alternative I on the one hand and alternative II and III are not significant.

**Energy Efficiency**

To evaluate the presented alternatives, the usage patterns of $F_A$ and $F_B$ need to be defined. Both functions $F_A$ and $F_B$ are used sporadically, thus a suspension of the related components if not in use promises savings on the order of 10W. The ability to scale down the processor of the integrated component $C_1$' and $C_1$'' in alternative I in order to conserve energy and the extent of possible savings depends on the resource intensity of the remaining basic functions. With alternatives II and III the functionality related to the provision of $F_B$ can be suspended.

**Reliability**

As outlined in chapter 5.2 , reliability is sensitive to the length and distribution of the functional chains defined in the logical architecture onto physical components. In architecture alternative I all logic is integrated onto one component thus increasing average failure rate, likewise in alternative II the deployment of both $F_A$ and $F_B$ onto $C_2$ is beneficial from a reliability perspective. In case of alternative III the provision of $F_A$ depends on the correct functioning of $C_1$ and $C_2$' and their interconnecting infrastructure and is thus slightly less reliable.

In this simple example, with only two components and no safety-critical functions involved, all alternatives can be assumed equivalent. However, as documented in chapter 6.1, complex functions involving up to 5 different components can exhibit substantial differences in reliability metrics depending on architecture. In that case a required reliability of service provision could render some architecture alternatives infeasible.

**Vehicle Programming**

With respect to vehicle programming we need to evaluate whether the extent of the programming data leads to an extension of the critical path of the programming process. The amount of data required by body and safety related functions $F_A$ and $F_B$ is small compared with the information and communication domain, thus all three alternatives are equivalent with respect to programming time. If the amount of data involved was considerably higher, distributed architectures allowing parallelization of programming would perform better.

**Diagnosis**

Centralization of functionality has positive impact on diagnosis activities. While the time for diagnosis does not depend on the number of components installed, the correctness of error identification.

The valuation of potential unnecessary replacements of $C_1$ or $C_2$ / $C_2$' due to distributed functionality in alternatives II and III needs to be performed based on estimates. With 100.000 units per year, a take-rate of 3% and a failure rate of 1% the number of service cases is merely 30. The alleged costs of unnecessary replacement even if monetarized with 100€ per occurrence is certainly negligible. Even with a 30% take-rate the number of service incidents and potential erroneous replacements will not reach the economic dimension of production or development costs.

**Correction and Upgrade**

In case of failure of a component, the costs of correction are dictated by costs of parts and labor for its replacement. The integrated components $C_1$' and $C_1$'' are more costly than single the components in the modular scenarios.

Regarding upgrade activities the extension of the basic configuration in alternative I to provide $F_A$ involves the replacement of $C_1$' by $C_1$'' at the cost of $C_1$'' which is naturally higher than $C_2$ or $C_2$'' alone. Changes to the infrastructure required by alternatives II and III on the other hand may compensate the increased upgrade costs of electronic components. In any alternative the sensors required by $F_A$ need to be installed and connected to $C_1$/$C_2$ in that respect all alternatives are suboptimal, however an alternative preserving the required infrastructure or sensors has not been suggested for evaluation.

Since costs of upgrades after delivery are transparent for the OEM, the valuation of upgrade-ability of the vehicle architecture is different from the OEMs perspective needs to consider expected revenues in after-sales which is beyond the scope of this work.

**Development and Integration**

The costs of development are listed in Table 21. Costs of integration have not been considered in these estimates. As outlined in chapter 5.3 , the number of necessary test setups at the component, system and vehicle test level and is sensitive to the implementation of architecture variability.

At the component level all three alternatives are equivalent, since the additional component in alternatives II and III is compensated by the additional component variant in alternative I. At the system test and vehicle test level, the alternative component in architecture alternative I doubles the number of necessary setups both in system and vehicle setups. Likewise alternative II defining two alternative implementations of $F_B$ requires separate test setups, while the functionality can be tested in a single set of tests with the modular alternative III. The difference in required test effort can be monetarized on the order of $10^5$ per additional system configuration.

**Correctness**

As outlined in chapter 5.3 architectural structure affects correctness of development less than novelty and functional size of the change. In alternative I the components $C_1$' and $C_1$'' need to be re-developed, thus introducing a risk of errors undetected in integration. In alternatives II and III merely the separate component $C_2$ and $C_2$' is developed anew. The functional dependencies between $C_1$ and $C_2$', due to the sensor data shared between $F_A$ and $F_B$ introduced with alternative III make development more complex and alternative III slightly less favorable than II. In effect alternative I is considered least favorable, while alternative II and III are positive or neutral respectively.

**Maintenance and Evolution**

To evaluate the maintainability of the presented alternatives we consider resource preservation and dependencies between individual units of evolution.

Regarding (inelastic) resources we need to take into account the remaining bandwidth and package space, as well as processing capacity in the components presented. As mentioned initially, $C_1$ is at the edge of its capacity. A re-development would introduce additional resources for further functional upgrades. Alternatives II and III do not change the existing implementation of $C_1$ thus introducing an additional risk of capacity problems limiting functional evolution. However $F_B$ can be moved to $C_2$ if required freeing additional resources on $C_1$. Regarding bandwidth, the integrated design in alternative I reduces busload, while the separation of $F_A$ and $F_B$ in alternatives II and III consumes bandwidth on the associated communication infrastructure.

The introduced variability of function partitioning of alternative II in distinct vehicle configurations incurs additional development costs, since the source of signals varies in each configuration in addition. Likewise the dependency of $F_A$ allocated on $C_2$' on function $F_B$ allocated on component $C_1$ introduces an unwanted dependency between separate units of development.

All in all the architectural value of alternative I with respect to maintenance and evolution is moderately negative.

**Communality**

For evaluation of communality we need to consider the ability to install the components in parallel product lines. This depends on the compatibility or communality (identical system structure, common partitioning of functionality), but likewise on the product-line specific functionality.

In alternative I $C_1$ implements a large part of the event chain of function $F_B$ in addition to basic functions which are not further specified. We can assume the basic functions to be vehicle specific, thus integration in other product lines, probably realizing these functions alternatively will incur development costs on the same order as the (re-) implementation of $C_1$. This alternative is to be considered deficient from a communality perspective.

Alternative II places the complete mutually dependent functional block $F_A$ and $F_B$ into one physical component and thus allows an installation of the component independently of basic vehicle functions in other product lines.

If no further constraints on parallel or future product lines exist, particularly breaches in communication technology or system structure, the re-use of $C_2$ in alternative II can be expected to save development costs on the order of hundred thousands of Euros with each product line. In addition synergies in functional evolution, integration effort and correctness are possible.

Alternative III defines a separate component $C_2$' which by itself can be installed in parallel product lines with less effort than the integrated component $C_1$' and $C_1$''. However the functional dependency with $F_B$ located on $C_1$ makes it more costly to integrate with other product lines with a distinct allocation of $F_B$. The assessment of this alternative with respect to communality aspects is hence moderately negative.

### 7.1.4 Results

The main criterion for decision on an architecture alternative in practice is the net present value reflecting the business case of the architecture change. In this particular case a positive business case is easy to construct, since the implemented function raises the value of the vehicles produced, thus the question to be answered by the analysis is merely which implementation alternative is most efficient.

In this discussion we use a simplified economic calculation taking into consideration the initial investment and the expected unit costs. The results assuming 100.000 units per year are presented in Table 24 and Table 25 for take-rates of 3% and 30% respectively. Expected costs of service and assembly were not taken into account. A moderate internal interest rate of 10% has been assumed making savings in production preferable compared to savings in initial invest.

| | Invest [in 1000€] | Annual prod. Costs | NPV@10% | ΔNPV vs. most cost-efficient alternative |
|---|---|---|---|---|
| Alternative I | 2.000 | 2.665 | 16.272 | -1400 |
| Alternative II | 600 | 2.665 | 14.872 | 0 |
| Alternative III | 1.200 | 2.645 | 15.365 | -493 |

Table 24 Architecture change A – cumulative business case at 3% take-rate

| | Invest [in 1000€] | Annual prod. Costs | NPV@10% | ΔNPV vs. most cost-efficient alternative |
|---|---|---|---|---|
| Alternative I | 2.000 | 2.800 | 16.995 | 0 |
| Alternative II | 600 | 3.245 | 17.978 | -983 |
| Alternative III | 1.200 | 3.050 | 17.534 | -539 |

Table 25 Architecture change A – cumulative business case at 30% take-rate

In this case, the differences in production cost of the various alternatives negligible and the up-front investment dominates the calculation. Only at a relatively high take-rate of 30% the difference in production costs makes up the difference in development costs.

Still a difference on the order of 1 million throughout a life-span of a product line of 7 years can be tolerated given the limited accuracy of take-rate estimates. Further properties can be included in the decision. A summary of the quality discussion (reduced to a three point scale) is presented in Table 26.

| | Alt I | Alt II | Alt III |
|---|---|---|---|
| Assembly | + | +/- | +/- |
| Logistics | + | - | - |
| Package Space | + | - | - |
| Thermal Compatibility / EMC | +/- | +/- | +/- |
| Weight | + | - | - |
| Energy Efficiency | - | + | +/- |
| Reliability | + | + | - |
| Vehicle Programming | +/- | +/- | +/- |
| Diagnosis | + | +/- | - |
| Correction & Upgrade | +/- | + | + |
| Development and Integration | - | - | + |
| Correctness | - | + | + |
| Maintenance | +/- | - | + |

| | | | |
|---|---|---|---|
| Evolution | - | + | +/- |
| Communality | - | + | +/- |

Table 26 Architecture change A – summary of non-monetarized criteria

Unsurprisingly the integrated solution performs better with efficiency-related metrics, whereas development and evolution oriented aspects favor the modular architecture. In this case, a three point qualitative assessment of individual properties is adequate to characterize the individual properties due to the coarse grained model of the architecture. The economic relevance of presented quality goals as discussed in chapter 5 suggests an individual weighting of $10^6$ for communality and evolution issues, $10^5$ for maintenance and integration as well as $10^4$ for service and operation related vehicle goals.

In the original architectural decision, based on the business case for a low expected take-rate alternative II was chosen. The economic relevance of maintenance and reuse in parallel and subsequent product lines in this case suggest a stronger weighting of modularity, thus favoring alternative III.

## 7.2 Architecture Change B

### 7.2.1 Overview

The presented Architecture Change B was requested as a cost down measure due to a reduced requirement set in a new product line. Alternatives include a re-design of existing component $C_1$ to adapt the physical resources to the reduced requirement set and/or the re-partitioning of its functionality remaining components $C_2$ and $C_3$ which host the logical functions $L_2$ and $L_3$ respectively. The functionality of $C_1$ to be retained includes the sensors and related logic required by $C_2$ and $C_3$, i.e. the logical components $S_2$ and $S_3$ potentially requiring re-allocation. The design space is constrained by the fact that, due to safety constraints, $S_3$ cannot be partitioned on $C_2$.

Apart from the intended architecture change, a re-implementation of $C_2$ is planned to exploit price drop in electronic components, thus costs of functional changes to $C_2$ can be covered by that project.

### 7.2.2 Alternatives

In total, four alternatives presented in Figure 48, have been investigated in detail. The estimated costs of development are listed in Table 27.

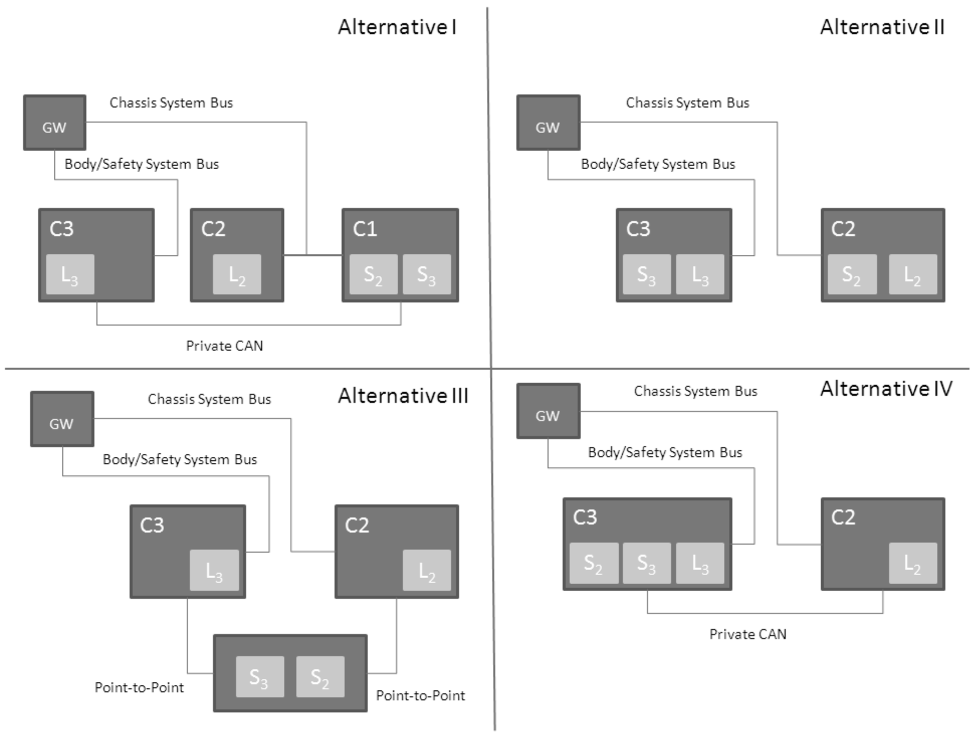


Figure 48 Architecture change B – design alternatives

**Alternative I** involves the re-design of $C_1$ by stripping all but the sensors and logic required for the provision of $L_2$ and $L_3$. This is the most costly implementation alternative, as the re-design of $C_1$ in order to reduce hardware resources and thus unit costs, involves a re-design of the hardware. The costs of implementation of $L_2$ and $L_3$ are almost negligible in that case.

**Alternative II** involves the allocation of $L_2$ onto $C_2$ and $L_3$ onto $C_3$ respectively. Since $C_2$ is scheduled for re-design in any case, the costs of implementation of $L_2$ can be neglected and $L_3$ are decisive. The difference in implementation costs to alternative I can be attributed to changes in the infrastructure, which were included in that cost estimate.

**Alternative III** involves the implementation of $L_2$ and $L_3$ on an integrated component. The integration yields synergies in development of 500k € as opposed to development of two independent components.

**Alternative IV** involves the allocation of $L_2$ and $L_3$ onto $C_3$. Due to the safety criticality of $C_3$, development is more expensive than if $L_2$ was implemented on a lower safety-level component.

Table 3 lists the expected costs of implementation in various scenarios.

| **Implementation** | **Alt. I** in 1000€ | **Alt. II** in 1000€ | **Alt. III** in 1000€ | **Alt. IV** in 1000€ |
|---|---|---|---|---|
| $L_2$ separately | | | | |
| $L_2$ on $C_2$ | | - | | |
| $L_3$ separately | | | | |
| $L_3$ on $C_3$ | | 1500 | | |
| Redesign $C_1$ | 5000 | | | |
| $L_2$ and $L_3$ on $C_3$ | | | | 1400 |
| $L_2$ and $L_3$ separately | | | 1000 | |
| **Total** | **5000** | **1500** | **1000** | **1400** |

Table 27 Architecture change B – estimated development costs

The estimates indicate that implementation on safety critical components is unsurprisingly more expensive than implementation on non safety-critical ECUs. Secondly synergies in implementation reduce development costs: implementing all required functionality on one component is still less costly than development of two independent ECUs, at least given the functional size and complexity of the change at hand. The differences in total development costs however are dominated by unrelated circumstances, i.e. the fact that one component is scheduled for re-implementation anyway. Differences in implementation costs of the listed alternatives add up to 4 million which is significant enough to justify a design decisions, still this amount can be easily compensated by the differences in production costs.

### 7.2.3 Evaluation

**Cost Efficiency (Electronic Components)**

The costs of the individual components have been estimated by potential suppliers and are listed in Table 28.

**Cost Efficiency (Infrastructure)**

The total cost of infrastructure is determined by the number of connections and the distinct kinds of interconnection. The costs of connection are included in the economic analysis and listed along with electronic components. Unsurprisingly, the alternatives involving a dedicated connection in addition to the existing communication buses are less cost-efficient than alternative II eliminating a component and additional connections altogether.

The estimated costs of additional wiring amount to 1€ in the case of alternative I and IV, and 2€ in case of alternative III.

| | Alternative I | Alternative II | Alternative III | Alternative IV |
|---|---|---|---|---|
| $C_1$ | 35 | - | - | - |
| $C_2$ | 110 | 130 | 110 | 111 |
| $C_3$ | 20 | 25 | 20 | 40 |
| $C_{S2S3}$ | - | - | 21 | - |
| Infrastructure | 1 | - | 2 | 1 |
| Total | 166 | 155 | 153 | 152 |

Table 28 Architecture change B – cost of electronic components

**Assembly**

With all of the considered components being mandatory, the relevant metric in the alternatives considered is the number of components. Alternatives III and IV involving only two components promise efficiency gains on the order of 1€.

**Logistics**

With respect to the logistics of the wiring harness, all alternatives are equivalent, since no configuration specific infrastructure is involved. Likewise alternative component implementations are not affected by the scenarios.

**Package Space**

The elimination of $C_1$ in alternatives II, III and IV results in reduction of required package space. Minor package space savings arise from the externalization of $S_1$ and $S_2$ into a separate component due to reduced periphery connections on the two remaining components.

**Thermal and Electro-Magnetic-Compatibility**

The functional density after component integration remains within technologically reasonable bounds, thus thermal or electro-magnetic compatibility do not pose a development risk.

**Weight**

The elimination of component $C_1$ in alternatives II, III and IV results in savings on the order of 0.3kg. This is partially compensated by the introduction of the sensor cluster $C_{S2S3}$. The additional infrastructure in alternative II, III and IV incurs additional weight on a similar order. However the monetarization of the efficiency gains in terms of fuel consumption or $CO_2$ emissions is marginal.

**Energy Efficiency**

The usage patterns of the services involved are not further documented. Assuming constant operation or hot standby due to required response times, the scenarios are equivalent.

**Reliability**

For the calculation of reliability, the distribution of event chains needs to be considered. The distributed implementation in alternative II, III and IV reduces expected reliability of the functions deployed on $C_2$ and $C_3$ respectively, with alternative IV performing slightly better due to improved reliability of $F_3$.

**Vehicle Programming**

With the code of $L_2$ and $L_3$ being subject to vehicle programming, all alternatives are equivalent, involving the programming of two components connected to independent communication buses.

**Diagnosis**

The distribution of the functions $F_2$ and $F_3$ between $C_2$ and $C_3$ on the one hand and $C_1$, $C_{S2S3}$ in scenarios II and V may result in worsened diagnosis of failures and potentially unnecessary exchange of components in case of errors. Alternative IV is slightly better, since the realization of function $F_3$ is concentrated on a single ECU, while $F_2$ is distributed onto $C_2$ and $C_3$. Alternative II is obviously the preferable solution from a diagnosis perspective, as it defines the shortest possible event chains.

**Correction**

With all scenarios considered equal with respect to vehicle programming, the costs of component replacement determine the costs of correction of individual vehicle configurations. With variability between individual vehicle configurations not affected by this architecture change, the costs of replacement of individual components can be assumed proportional to costs of electronic components. With an equal probability of failure for all components, alternative I and II yields average costs of while alternatives III and IV.

**Upgrade**

Since no optional functions are involved, the proposed architecture change is insensitive to upgrade scenarios.

**Development and Integration**

Costs of development are listed in Table 27. Costs of integration and validation can be assumed similar to the costs of development. Since no component alternatives are involved, the effort at the subsystem and system level is equal each alternative. Alternatives eliminating one component promise savings on the order of $10^5$€ for basic component functions.

**Correctness**

Regarding correctness the assessment need so focus on the necessary re-implementation of components, as extensive development activities involves additional integration and validation activities and bears the risk of errors remaining undetected during integration.

Despite high development costs, scenario II and V involve the fewest functional changes. In fact the logic on component $C_2$ and $C_3$ remains untouched. Alternative IV likewise requires no changes to the functionality in $C_2$, but the re-design of $C_3$ bears development risks.

The valuation of such risks is bound to major uncertainty, as a rule of thumb an increased number of service cases by 0.2% in the first year of production is assumed. With 500.000 units produced and service costs on the order of 100€ per incident, warranty costs of $10^5$ euro can be attributed to alternative II and 50.000 to alternatives III and IV.

**Maintenance and Evolution**

To evaluate the maintenance effort and risk of the presented alternatives we consider resource preservation and dependencies between individual units of evolution and development.

The resource intensity of $L_2$ and $L_3$ respectively is not documented. Critical resources in this respect are communication bandwidth on the system communication buses as well as package space. The bandwidth intensive communication between $L_2$ and $S_2$ is routed through the chassis CAN in alternative I which is less favorable in terms of resource preservation. Likewise the reduction of physical resources within $C_1$ can be considered negative.

Alternative II eliminates requirements of external communication altogether, at the expense of redundant sensors on $C_2$ and $C_3$ respectively, while alternative III and IV introduce additional resources into the system. With respect to package space, alternatives II and IV preserve eliminate one component and preserve resources in that respect.

Individual evolution of units of development is simplified with alternative II, alternatives I and III separate the functional logic of $L_2$ and $L_3$, but introduce a common dependency on a third component. Alternative IV eliminates redundant sensors required by $L_3$ and $L_2$, but introduces a dependency of $C_2$ on $C_3$ increasing the risk of changes in future.

**Communality**

For the consideration of reuse the architectural design in the remaining product lines has to be considered. The reason for the architecture change was a significant difference in requirements between the existing product lines and the one under development, thus a breach in the implementation is inevitable.

The functional partitioning of the remaining product line architectures is compatible with alternative I, i.e. a separate implementation of $L_2$ and $L_3$, and shared sensors on a separate component, thus alternative III allowing a reuse of both $C_2$ and $C_3$ preserved communality with those parallel product lines. In the same manner, alternative VI allows a reuse of component $C_3$, whereas alternative II, on a short term, requires a re-design of $C_2$ and $C_3$ and is thus the weakest design alternative regarding reuse.

From a long term perspective both integrated components $C_2$ and $C_3$ are potential candidates for reuse in subsequent product lines generations, especially with the external dependencies eliminated. These however come at the cost of additional development and integration. Especially if $F_1$ (the functionality not required in the product line at hand) depends on sensor logic $S_1$ and $S_2$.

### 7.2.4 Results

In this simple example with no optional functions and components involved the business case of the change is the primary argument for the decision. The cumulated business case of all alternatives assuming a production output of 500.000 units per year and a total life-span of 7 years is depicted in Table 29.

| | Invest | Production Cost per year [in 1000€] | Production Costs NPV [in 1000€] | Cumulated NPV [in 1000€] | Δ NPV [in 1000€] |
|---|---|---|---|---|---|
| Alternative I | 5000 | 83000 | 302.347 | 307.248 | -28.999 |
| Alternative II | 1.500 | 77500 | 282.312 | 283.813 | -5.564 |
| Alternative III | 500 | 76500 | 278.670 | 279.170 | -921 |
| Alternative IV | 1.400 | 76000 | 276.848 | 278.249 | 0 |

Table 29 Architecture change B – cumulative business case

The net present value is calculated at 30% interest over-weighting up-front investment in comparison to cost savings stretched throughout several years.

The business case shows that savings on the order of 5€ easily make up for differences in up-front investment of several millions of euro. Secondly scenario I can be eliminated from the list of candidates, as none of the non-monetarized criteria will make up the difference of 30 million in production costs.

| | Alternative II | Alternative III | Alternative IV |
|---|---|---|---|
| Assembly | + | +/- | + |
| Logistics | +/- | +/- | +/- |
| Package Space | +/- | + | +/- |
| Thermal Compatibility / EMC | +/- | +/- | +/- |
| Weight | + | +/- | +/- |
| Energy Efficiency | +/- | +/- | +/- |

| | | | |
|---|---|---|---|
| Reliability | + | - | +/- |
| Programming | +/- | +/- | +/- |
| Diagnosis | + | - | +/- |
| Correction & Upgrade | - | + | - |
| Development and Integration | + | +/- | + |
| Correctness | -- | +/- | - |
| Maintenance | + | +/- | +/- |
| Communality | - | + | - |
| Evolution | + | + | - |

Table 30 Architecture change B – summary of non-monetarized criteria

A critical inspection of the remaining scenarios reveals that the decision is actually one between economic gains in case of alternative III and VI and the modular, however redundant implementation in scenario II, promising additional savings in weight and infrastructure. The question is hence, whether increased modularity and thus improved maintenance make up for the difference in present value. Proponents of the integrated scenarios will argue, that the dependency of two independent ECUs on a common sensor cluster does not pose a design deficiency, making alternative III more preferable to III and VI.

From a communality standpoint, alternative III allows the reuse of current component designs, whereas scenario VI involves the re-design of $C_3$ and introduces a dependency reducing the potential for reuse of $C_3$ or $C_2$ in isolation.

Functional evolution cannot be assessed without specifying evolution scenarios. Regarding scarce resources, i.e. bandwidth on system communication buses, both the co-location of event chains and introduction of additional resources through additional connections is beneficial.

In the original architecture change alternative III has been chosen. Empirical data on the consequences of its lower modularity, i.e. reliability and diagnosis effort, compared to alternative II for instance is not yet available. Major issues with development or integration have not been reported.

In this case the introduction of additional evaluation criteria has not led to a different result in evaluation. If a model of functional requirements and requirements variability of parallel product lines was available and a roadmap of functional evolution the analysis could have provided more founded results.

## 7.3 Architecture Change C

### 7.3.1 Overview

The analyzed architecture change C was requested as a cost-down measure and involves a re-design of two central components $C_A$ and $C_B$ and potential integration of those components in order to reduce costs of production. Both components are mandatory and included in basic vehicle configuration, providing functions $F_A$ and $F_B$ respectively in addition $C_B$ implements a number of optional functions: $F_{C1}$, $F_{C2}$ and $F_{C3}$ thus scalability either through variants or segregation of optional functions promises cost savings in production. Separate configuration of $F_{C1}$, $F_{C2}$ and $F_{C3}$ has not been specified, thus alternatives involving only one of the optional features will need to be realized through configuration. The functional size involved is not further documented, however the total costs of design alternatives involving a re-development gives a hint that a major functional change is involved.

### 7.3.2 Alternatives

The alternatives considered are depicted in Figure 49.

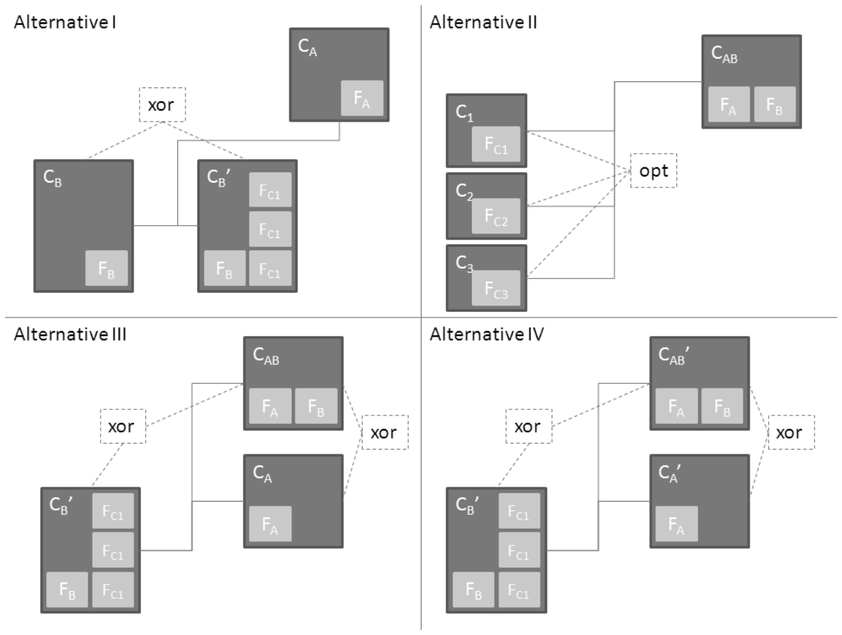


Figure 49 Architecture change C – design alternatives

**Alternative I** involves the fewest changes to the existing design. The changes include merely a re-design of the software to the altered requirement set based on the existing hardware platform. Component $C_A$ remains unchanged, while $C_B$ is designed extensible providing optionally $F_{C1}$, $F_{C2}$ and $F_{C3}$. Needless to say it also promises the least savings in production.

**Alternative II** involves the integration of both components $C_A$ and $C_B$ into $C_{AB}$ and the allocation of optional functions on external sensors and actuators. Thus one integrated, central component as well as three periphery components are to be re-implemented.

**Alternative III** likewise involves the integration of $C_A$ and $C_B$ into $C_{AB}$. The optional functions in a full featured configuration are to be provided as in alternative I, i.e. the original component $C_B$ is to be installed in that case.

**Alternative IV** is structurally equivalent to alternative III but involves a complete re-design of the hardware platform of $C_{AB}$' in order to reduce hardware resources.

The costs of development have been estimated by potential suppliers and are listed in Table 31

| | **Alternative I** in 1000 € | **Alternative II** in 1000 € | **Alternative III** in 1000 € | **Alternative IV** in 1000 € |
|---|---|---|---|---|
| External development | 900 | 7000 | 4000 | 9000 |
| Internal development | 800 | 1500 | 1500 | 2000 |
| **Total** | 1.700 | 8.500 | 5.500 | 11.000 |

Table 31 Architecture change C – estimated development costs

Unsurprisingly Alternative I is the least expensive in development. Regarding the remaining alternatives, the difference between the individual component implementations is considerable. Even though an integrated component $C_{AB}$ is to be developed in all cases, the development costs of periphery components (Alternative II) sum up to 3 million euro. A re-development of hardware to squeeze out potential resource savings to lower production costs amounts to 5 millions of euro.

### 7.3.3 Evaluation

#### Electronic Components

The costs of the individual components have been estimated by potential suppliers and are listed in Table 32. As outlined in chapter 5.1 integration of components yields savings on the order of 10€ compared to physically segregated architectural designs.

| $C_A$ | $C_B$ | $C_B$' | $C_{AB}$ | $C_{AB}$' | $C_1$ | $C_2$ | $C_3$ |
|---|---|---|---|---|---|---|---|
| 70€ | 16€ | 26€ | 83€ | 76€ | 10€ | 6€ | 6€ |

Table 32 Architecture change C – costs in production

#### Infrastructure

Naturally the alternatives involving more physical segregation lead to higher costs of infrastructure. The geometrical placement of the individual components is not specified in the architecture change, but the costs of wiring are determined by the number of connections rather than by their individual lengths. The estimated costs are listed subsequently:

**Alternative I** involves a single connection between $C_A$ and $C_B$/$C_B$' with each configuration the costs of wiring are estimated at 2€.

**Alternative II** involves no additional wiring in the basic configuration, but up to three distinct connections if optional equipment is ordered. The costs have been estimated at 2€ per individual connection.

**Alternative III and IV** are equivalent and involve no additional wiring in the basic configuration but additional connections with the optional equipment.

### Assembly

The costs of assembly can be attributed to the number of individual components. According to typical business cases the costs sum up to 1€ per connected ECU. The order of installation and potential impact on diagnosis on the assembly line cannot be estimated based on the information available.

### Logistics

As with the previously discussed architecture changes the general dependency of the customer-specific wiring harness is not changed by individual measures. Only a conceptual change in infrastructure should eliminate the dependency of the wiring harness.

Alternative II introduces the most variants to the infrastructure, due to three independent components which can be combined arbitrarily. With respect to the logistics of the wiring harness, the remaining alternatives are equivalent.

Regarding logistics of components, we consider the number of component variants. Alternative I introduces at least two distinct variants, but a single supplier dependency with $C_A$. Alternative II results in no additional variants at all, but four distinct components all of which depend on a single supplier. Alternatives III and IV add no variants at all and allow the substitution of $C_{AB}$ with an equivalent implementation based on $C_B$' and $C_A$. Expressed qualitatively, alternatives III and IV perform best, followed by alternative I and finally alternative II.

### Package Space

The available package space is considered for a fully equipped configuration, i.e. one with $F_{C1}$, $F_{C2}$ and $F_{C3}$ equipped. All four alternatives are at least feasible in terms of packaging. Alternative II involves the placement of three, the remaining alternatives up to two separate components.

### Thermal and Electro-Magnetic-Compatibility

The high integration of $F_A$ and $F_B$ in alternative III and IV poses a risk to the thermal and electro-magnetic compatibility. This risk is cannot be quantified without additional data on resource intensity and computational capacity of $C_{AB}$, but in case it becomes imminent should cause additional development costs and project delays up to the magnitude of the estimated development budget.

### Weight

In terms of weight, alternatives III and IV involving one component and no additional infrastructure in the basic configurations promise savings on the order of .5kg in about 50% of the produced vehicles. Alternatives I and II define more components and additional infrastructure either in the basic configuration or with optional features installed.

**Energy Efficiency**

The usage patterns of $F_A$, $F_B$ and $F_{C1}$, $F_{C2}$ and $F_{C3}$ respectively are not further specified. Assuming completely uncorrelated usage and no inter-dependencies between the functional components involved, the modular architectures should exhibit higher energy efficiency through suspension of unused components.

**Reliability**

Without further specifying the distribution of functionality and dependencies within the vehicle, all alternatives must be considered equivalent.

**Vehicle Programming**

The separation of functionality into two components as proposed in alternatives I, III and IV should lead to improved times of vehicle programming. It should be noted that the volume of data and times of component programming do not extend the critical path of vehicle programming, due to the limited volume of data in the body and safety domain.

The risk of process crashes during vehicle programming increases with the number of components involved, hence alternatives III and IV could be considered slightly better than alternative I and II in the basic configuration.

**Diagnosis**

With the distribution and further dependencies not further specified, all alternatives must be considered equivalent in terms of diagnosis.

**Correction / Upgrade**

For an evaluation in terms of correction and upgrade activities, the component costs and effort to replace the respective components needs to be considered.

Replacement of $C_A$ or $C_B$ independently in case of malfunction, as should be required in alternative I can be expected less costly than the replacement of integrated $C_{AB}$ in any other alternative because of component costs. A defect in the components providing $F_{C1}$, $F_{C2}$ or $F_{C3}$ will result in a replacement of a highly integrated component in all but alternative II.

The expected costs of upgrade in alternative II are likewise comparably low, due to the low production costs of all three optional components. In alternative I the same upgrade of optional function $F_{C1}$, $F_{C2}$ or $F_{C3}$ requires the replacement of $C_B$, and an installation of two additional components including the required infrastructure in alternatives III and IV making the latter two designs least extensible.

**Development and Integration**

Development effort has been estimated by potential suppliers and is listed in Table 31. Alternative I most cost efficient as the status quo, with changes only to the software. In absolute numbers, alternatives II and III are comparable in terms of effort, while the re-design of the hardware platform of alternative IV requires significant costs and project time.

From an integration standpoint alternative II is preferable with no alternative configurations and only one set of system and vehicle tests to be executed (provided no further variability in the architecture). The remaining alternatives exhibit component alternatives in distinct configurations, requiring additional tests at the system and vehicle level. The synergies in testing of component $C_A$, being present in both configurations make alternative I slightly more efficient in terms of integration and validation.

**Correctness**

For the analysis of expected correctness we consider the extent of functional changes in the individual alternatives. The extent of (external) development is an estimate of the extent of functional changes and thus the expected number of maturity of architecture at the time of production.

In addition to changes to the functional size of the architecture, structural metrics could, to a smaller extent, affect the risk in development and integration. All components reside in the body and safety domain hence organizational responsibility should not be an issue. The integration of functionality from multiple suppliers on a single component can be considered a risk to system correctness however the interviews documented in chapter 5.3 do not support this assumption.

**Maintenance / Evolution**

To evaluate the maintainability and evolution of the presented alternatives we consider resource preservation and dependencies between individual units of evolution and development.

High integration of $C_{AB}$ in alternatives II, III and IV reduces the amount of computational resources available for further functional upgrades. If additional functions beyond the currently installed resources ere to be provided, alternative II and IV will require a re-design. The functional upgrade scenario for alternatives III and IV is slightly more positive, only additional basic functions beyond current design would require the high-end configuration to become the standard incurring additional costs in production.

As a risk in development and evolution, the distinct allocation of function $F_B$ in alternatives III and IV should involve additional configuration and maintenance effort as well as potential risks, if other functions depending on $F_B$ will need to be kept aware of the function's allocation.

The independent evolution of $F_A$, $F_B$ should be less risky with proposed design alternative I. $F_{C1}$ through $F_{C3}$ on the other hand can evolve independently in architecture alternative II. For an appropriate weighting of these scenarios in the total evaluation, the expected functional changes in the respective functions should be further investigated. However a maintenance of $C_1$, $C_2$ and $C_3$ is questionable as independent components given the trend of component integration.

**Communality**

In order to assess the impact of the proposed architecture change on communality of the product line architecture, the proposed solutions need to be assessed with respect to existing designs and the ability to reuse the presented components in parallel product lines.

The existing design is represented by alternative I. $F_A$ and $F_B$ are separately allocated onto $C_A$ and $C_B$. Optional functions are allocated on a variant $C_B$'. The introduction of an integrated version $C_{AB}$ in alternative II, III and IV precludes the installation of $C_{AB}$ in existing product lines introducing additional variability in the partitioning of $F_A$ and $F_B$ in distinct product lines. Parallel product lines will not benefit from lowered in production costs after re-implementation of $C_{AB}$, or if so, only at substantial investment in development.

All of the functions $F_{C1}$, $F_{C2}$ and $F_{C3}$ are not specific to the product line at hand thus an independent installation of $C_1$ through $C_3$ made possible through alternative II does not promise sufficient efficiency gains due to resource adaptation in parallel product lines.

### 7.3.4 Results

For the calculation of optional equipment configurations two scenarios with distinct take-rates of $F_{C1}$, $F_{C2}$ and $F_{C3}$ have been considered as listed in Table 33.

| Opt. Functions | None | $F_{C1}$ | $F_{C2}$ and $F_{C3}$ | $F_{C1}$, $F_{C2}$ and $F_{C3}$ |
|---|---|---|---|---|
| Scenario 1 | 60% | 20% | 10% | 10% |
| Scenario 2 | 40% | 30% | 15% | 15% |

Table 33 Architecture change C – assumed take-rates of optional functions

A summary of the economic evaluation at the vehicle level of the presented alternatives is depicted in Table 34.

| | Alternative I | | Alternative II | | | | Alternative III | | Scenario IV | |
|---|---|---|---|---|---|---|---|---|---|---|
| Opt. Functions | none | $F_{C1}$, $F_{C2}$, $F_{C3}$ | none | $F_{C1}$ | $F_{C2}$, $F_{C3}$ | $F_{C1}$, $F_{C2}$, $F_{C3}$ | none | $F_{C1}$, $F_{C2}$, $F_{C3}$ | none | $F_{C1}$, $F_{C2}$, $F_{C3}$ |
| $C_A$ | 70 | 70 | | | | | | 70 | | 70 |
| $C_B$ | 16 | | | | | | | | | |
| $C_B$' | | 26 | | | | | | 26 | | 26 |
| $C_{AB}$ | | | 83 | 83 | 83 | 83 | 83 | | | |
| $C_{AB}$' | | | | | | | | | 76 | |
| $C_1$ | | | | 10 | | 10 | | | | |
| $C_2$ | | | | | 6 | 6 | | | | |
| $C_3$ | | | | | 6 | 6 | | | | |
| Wiring | 2 | 2 | - | 2 | 4 | 6 | - | 2 | - | 2 |

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Assembly | 2 | 2 | 1 | 2 | 3 | 4 | 1 | 2 | 1 | 2 |
| Total | 86 | 96 | 83 | 93 | 95 | 105 | 83 | 96 | 76 | 96 |
| Scenario 1 avg. | 90 | | 88,4 | | | | 88,2 | | 84 | |
| Scenario 2 avg. | 92 | | 91,1 | | | | 90,8 | | 88 | |

Table 34 Architecture change C – costs of Individual configurations

The cumulative business case including up-front investment in development and assuming 500.000 produced units per year and a life-span of seven years is listed in Table 35 and Table 36 for the distinct take-rate scenarios respectively. The internal interest rate (expected project rate of return) lies at 30% which over-weights investment, and under-weights unit costs.

| | Production Cost per year [in 1000€] | Production Costs NPV [in 1000€] | Dev. Cost [in 1000€] | Cumulated NPV [in 1000€] | Δ NPV [in 1000€] |
|---|---|---|---|---|---|
| Alternative I | 45000 | 163.923 | 1.700 | 165.624 | -1.628 |
| Alternative II | 44200 | 161.009 | 8.500 | 169.509 | -5.514 |
| Alternative III | 44100 | 160.645 | 5.500 | 166.145 | -2.150 |
| Alternative IV | 42000 | 152.995 | 11.000 | 163.995 | 0 |

Table 35 Architecture change C – cumulative business case scenario I

With lower take-rates of 20%, 10% and 10% of the optional functions, alternative IV is the most cost-efficient., however the difference in net present value compared to the two next economical alternatives is below two million euro. This amount calculated throughout a life-span of seven years can well be tolerated if another alternative was preferable for quality reasons.

| | Production Cost per year [in 1000€] | Production Costs NPV [in 1000€] | Dev. Cost [in 1000€] | Cumulated NPV [in 1000€] | Δ NPV [in 1000€] |
|---|---|---|---|---|---|
| Alternative I | 46000000 | 167.566.314 | 1.700.000 | 169.266 | 0 |
| Alternative II | 45550000 | 165.927.079 | 8.500.000 | 174.427 | -5.161 |
| Alternative III | 45400000 | 165.380.667 | 5.500.000 | 170.881 | -1.614 |
| Alternative IV | 44000000 | 160.280.822 | 11.000.000 | 171.281 | -2.015 |

Table 36 Architecture change C – cumulative business case - scenario II

Assuming slightly higher take-rates of 30%, 15% and 15% of the optional functions $F_{C1}$, $F_{C2}$/$F_{C3}$ and $F_{C1}$, $F_{C2}$ and $F_{C3}$ respectively, alternative I is the most cost efficient. Once again, two other options III and IV are similar in monetary terms. Given these figures, alternative II, despite being the most modular, can be eliminated from the list of candidates, whereas alternative III should be considered with lower priority, as its net present value is lower than the preferred candidates for both take-rate scenarios.

The remaining alternatives are in fact quite similar. The optional functions installed in highly equipped configurations are integrated in a single component. The structural difference between alternative I on the one hand and alternative III and IV on the other hand is the integrated component $C_{AB}$ to be installed in basic configurations in order to reduce average production costs.

The summary of non-monetarized criteria for architecture alternatives I, III and IV resulting from the discussion above are listed below:

| | Alternative I | Alternative III | Alternative IV |
|---|---|---|---|
| Logistics | | | |
| Infrastructure | +/- | - | - |
| Electronics | +/- | + | + |
| Package Space | +/- | +/- | +/- |
| Thermal Compatibility / EMC | +/- | - | - |
| Weight | - | + | + |
| Energy Efficiency | +/- | +/- | +/- |
| Reliability | +/- | +/- | +/- |
| Programming | + | +/- | +/- |
| Diagnosis | +/- | +/- | +/- |
| Correction & Upgrade | + | +/- | +/- |
| Development and Integration | + | +/- | - |
| Correctness/Maturity | + | +/- | - |
| Maintenance / Evolution | + | +/- | +/- |
| Communality | + | - | - |

Table 37 Architecture change C – summary of non-monetarized criteria

Without further weighting of the individual criteria, the comparison of non-monetarized properties of alternative I is the favorable one. Alternatives III and IV perform better with basic configurations, increasing predominantly weight and energy efficiency and reducing some logistical risks. Alternative I on the other hand reduces variability of infrastructure and vehicle configurations and promises better performance in maintenance and evolution scenarios. Taking into consideration that the economic weighting of correctness, integration and communality oriented quality properties is an order of magnitude higher than service and operation related ones, alternative I should be chosen.

The decisive criterion in this architecture change is expected correctness. Prospective savings of less than 2 million euro or less with more optimistic take-rate estimates do not justify the additional risk of errors undetected in integration and validation introduced through re-implementation of a large part of functionality.

An additional argument in this architecture case is, whether the newly implemented component $C_{AB}$ promises considerable cost savings in production if installed in future or parallel product lines. None of this is documented and the extent of product-line specific functionality implemented on $C_{AB}$ needing re-design in parallel product lines can be considerable. Based on the presented discussion alternative I remains the preferred design choice.

## 7.4 Summary

We have reviewed three documented architectural changes with respect to quality properties and their economic relevance as discussed in chapter 5. The original architecture decisions were made purely based on a business case calculation and the architect’s technical recommendation, with the business value being given preference.

The presented re-evaluation of architectural cases shows that evaluation based on structural analysis and the extended set of criteria yields more founded results pure business case calculation. Particularly with alternatives involving optional equipment of uncertain take-rates and relatively small differences in net present value, the structural analysis is an improvement to the existing practice. Contrary to previous quality models presented in Chapter 2, the evaluation criteria are not postulated, but derived from the activities related to the system’s life-cycle and economically justified.

Still the results are limited due to the insufficient definition of requirements variability across distinct product lines and functional evolution scenarios. For a significant improvement in prediction accuracy, more detailed models of the functional architecture and logical component architecture are necessary

.

# 8 Summary and Outlook

## 8.1 Summary

The intention of this work was to make architectural decisions in the domain of automotive E/E systems more transparent and the evaluation of architecture alternatives more systematic. The means to achieve this was to draw a holistic model describing desirable properties of E/E systems and investigate the consequences of early decisions regarding system structure on the achievement of the previously defined properties by the system under development.

The review of related work and current practice in chapter 2 and 3 revealed that generic quality models on the one hand are not well suited to investigate the specific issues related to in-vehicle E/E systems. However we have adopted the life-cycle or activity-based approach. Work related to the automotive domain and common heuristics of architecture evaluation on the other hand lacks a concise argumentative chain from architectural properties to goals and a priorization of quality goals in order to trade-off design alternatives. The model presented here closes this gap by separating architectural means and goals and investigating the economic relevance of individual goals according to related activities in the life-cycle.

In chapter 4.1, in order to identify the key properties defining system quality, we have discussed the life-cycle of E/E systems, identifying the *vehicle level* related to production, operation and service activities, the *product line architecture level* concerned with development, verification and correctness issues and the *architectural blueprint level* at which communality between individual product lines and evolution over time can be identified as quality goals.

Analogously in chapter 4.2, in order to identify the architectural means for optimization, we have reviewed the layered architecture model, consisting of a functional layer, defining requirements, a logical layer, describing their realization and interdependencies, the physical layer containing the physical resources and, relevant for the analysis of infrastructure, the topology layer. We have identified the architectural means based on this model as: integration and segregation of components, introduction of variants and allocation of functionality according to certain criteria, such as functional dependencies, communication intensity or correlation of use.

Given the architectural means and goals for optimization, we have discussed the relationship between architectural structure and properties related to the vehicle's production, operation and service as well as development, verification and evolution in chapter 5. Based on a number of architecture cases and documented consequences of changes, we must conclude that architectural structure foremost affects cost-efficiency at the vehicle level. Even though properties such as reliability or vehicle programming are affected by system structure, there are more effective means for their optimization thus these qualities do not justify architectural changes.

The number of electronic components and systems or conversely – functional density of components and subsystems, are the most dominating structural properties. Even though component centralization seems to be the preferred mean to reduce production costs, at least equal gains can be achieved through adaptation of physical resources to reduced sets of requirements, defined by distinct vehicle (feature) configurations or product line specific requirements.

The second most important goal in system development by economic relevance and consequences, is efficiency of verification and hence correctness. From the perspective of the OEM as a systems integrator, this directly relates to the number of alternative system configurations to be tested as well as component variants across distinct product lines and the extent of individual development in each product line. In architectural means this translates to compatibility of functional allocation across product lines as a structural property. As opposed to that, we have found only weak evidence that the degree of interconnection significantly increases integration effort or significantly decreases correctness. If at all, these errors have been attributed to methodical deficiencies, rather than architectural decisions.

It should be noted that the impact of architectural structure on development and verification effort within a single product line is not as immediate as on production and is dominated by the extent of functional changes as well as component redesigns. Since no development projects exhibiting comparable functional size but differing in architectural structure were documented for analysis, the relationship between structure and development related properties denoted in chapter 5 was established mainly by expert interviews.

The conclusions drawn from chapters 4 and 5 led to the analysis of common optimization strategies in the domain of automotive E/E systems – centralization and modularity – in chapter 6. Given the constraint of variable configurations we investigated the expected consequences of radically modular and centralized architectures. With current, highly distributed architectures, centralization improves most quality properties, mainly by increasing physical resource efficiency and reducing the length of event chains concurrently. As of today, the downsides of centralization, in the presence of highly variable requirements, do not observably reduce quality of E/E systems.

In chapter 7 we analyzed three architectural changes according to the presented evaluation method. Our findings show that in fact development and production costs dominate the architectural decision. However if, given the uncertainty of take-rates of optional equipment, the design alternatives are equivalent, further criteria need to be applied. We could show that the presented criteria can well complement the business-case oriented analysis, and with otherwise equivalent architecture alternatives, precedence should be given to communality and integration efficiency. This favors alternatives reducing variability and increasing communality, rather than to those reducing functional dependencies.

As concluded in chapter 7 the accuracy of architecture evaluation as presented here could benefit greatly, if more detailed models reflecting variability of requirements across product lines were available. Such models would also aid the definition of evolution scenarios to assess architectural stability and hence the costs and risks introduced through re-implementation of components.

## 8.2 Conclusion

The primary contribution of this work was to improve the quality of automotive E/E system architecture evaluation by presenting a broader range of quality and optimization goals, discussing their economic relevance and investigating the impact of architecture on the achievement of these goals. We have shown that the presented method of structural analysis and qualitative assessment leads to more founded decisions and expectedly lower life-cycle costs. Still the presented method depends on models sufficiently capturing requirement variability.

Based on the analyzed architecture cases and documented consequences of changes, we can prioritize the catalogue of desirable properties by economic relevance and the impact of architectural structure on these properties. The predominant goals are cost-efficiency in production, development and validation effort and correctness.

**Cost-efficiency in production** - we must conclude that architectural means foremost affect cost-efficiency at the vehicle level. Given the number of units produced it is also the most economically relevant property. Both the number of electronic components and the total number of systems are the most dominating structural properties. Even though component centralization seems to be the key to reducing production costs, at least equal gains can be achieved through adaptation of physical resources to reduced sets of requirements, defined by distinct vehicle (feature) configurations or product line specific requirements.

**Development and integration efficiency** - the second most important goal in system development by economic relevance and consequences, is efficiency of implementation and integration. From the perspective of the OEM as a systems integrator, this directly relates to the number of alternative system configurations as well as component variants across distinct product lines. All of the interviewed domain experts noted that both effort and quality are foremost positively affected through reuse of components developed and validated in parallel product lines. Anecdotal evidence indicates savings of 80% of initial costs and errors in integration with reuse or adaptation of existing components. Architectural structure indirectly affects the extent of such individual development through the definition of external interfaces and compatible allocation of functions and components across distinct product lines.

**System correctness -** insufficient correctness of the in-vehicle electronic systems can have astronomical economic impact if a recall already produced vehicles becomes necessary. In practice such critical errors are rare, as automotive E/E system development aims at eliminating most errors prior to mass production. Deficient system quality in late development stages rather leads to additional development and validation cycles than the release of deficient systems. A reduction of the number of errors both detected undetected during integration, in line with development effort, goes along with an increase of communality between individual product lines, which is affected by the same architectural properties as discussed above. Secondly the test coverage depends on the number of alternative vehicle configurations, defined by alternative and optional components.

The remaining properties within the quality model defined herein can be subdivided into optimization goals and design constraints. The economic gains of optimization goals such as duration and accuracy of diagnosis, vehicle programming, and energy efficiency through architectural means are noticeable, but lie below those mentioned above. In addition these properties are sensitive to other measures than the architectural means mentioned above. Properties reflecting constraints on system design on the other hand, such as packaging space, thermal and electromagnetic compatibility etc. are to be considered in architecture evaluation, due to prohibitive costs of violating such constraints, but a reduction beyond a certain threshold does not yield additional monetary gains.

In total, even though total development costs per product line are lower than costs in vehicle production almost by an order of magnitude, as outlined in chapter 5.1, the architectural business cases cited in chapter 7 document that in decisions on architectural changes the economic impact is equally affected by production costs as by up-front investment in development. The estimated development costs of the architectural changes were between 0.5 million Euros for adaptation of existing components and 11 million Euros for a complete redevelopment, with an average necessary investment of 3.5 million Euros. The resulting differences in production costs over a 7 year period were between 1 million Euros and 26 million Euros with an average of 6.5 million Euros. This indicates that despite higher absolute costs of production vs. development, the fraction affected by architectural decisions is comparable for both development and production. The related architectural decisions were based on an anticipated business value of individual alternatives of 6 million Euros on average, i,e, the same order of magnitude as initial development costs.

In all of cited examples only development and production costs within one product line have been considered. If reliable statements on the possible on adaptation of involved components in parallel product lines could have been made, distributing the development costs over more units, potentially more economically suitable optimizations could have been implemented. More importantly, the adaptation would have improved correctness of subsequent product lines.

Such statements on the adaptation of components in parallel product lines could have only been made, if the requirements but also functional allocation and interfaces across product lines were captured and available to the architect. Thus, as concluded in chapter 7, the accuracy of architecture evaluation as presented here could benefit greatly from architecture and requirements models. More importantly, by extending the horizon of architectural decisions, the extent of re-development in individual product lines could be reduced if such an adaption in parallel product lines was conceived at design time. Proactively modeled variability and evolution of requirements and managed architecture would improve the degree of reuse in an increasingly heterogeneous portfolio of product lines. Reuse of components in turn could yield savings on the order of millions of Euro per product line, as indicated by the results in chapter 5.3.

If a general conclusion beyond the analysis of individual changes or product lines is to be drawn based on the findings in this work, then how requirements and constraints affect vehicle architectures today and how these architectures will perform given the changes expected in future.

The introduction of variability into system architecture in order to reduce the amount of resources installed is the main driver for complexity and complexity related costs. Besides centralization, most efficiency gains can be attained through adaptation of architecture to a reduced set of requirements. On the other hand, variability drives subsequent life-cycle costs. The efficient implementation of architecture variability is an immediate goal for system development. Depending on the take-rate and dependency of requirements on physical resources, either component variability or modularity is the preferred strategy.

Depending on the architectural strategy employed, precedence can be given to production or development, verification and validation and correctness. With centralized architectures production costs can be reduced on the order of 10%-20% at the expense of exponentially increasing costs of complexity at the higher end of implemented variability. With modular architectures costs of complexity grow significantly slower at the prize of reduced efficiency in production.

Figure 50 and 51 qualitatively depict this relationship for centralized and modular architectures respectively, denoting the economic scales of increased variability and related costs of production, development, verification and correctness.

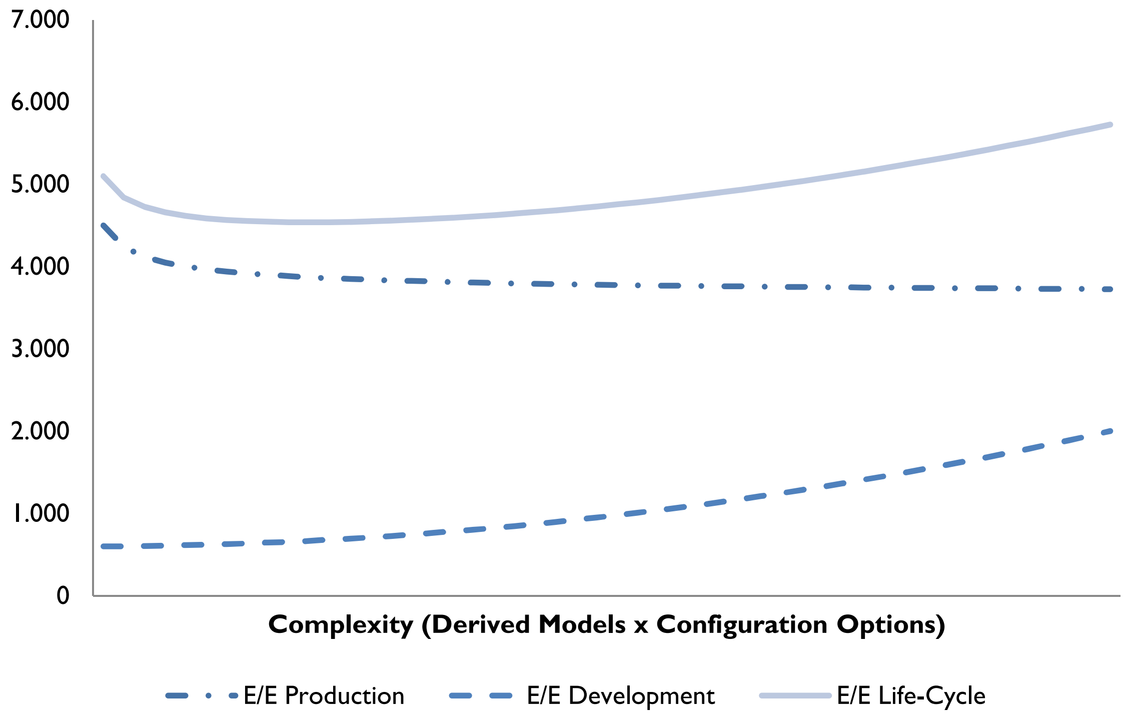


Figure 50 Life-cycle costs of centralized architectures with increasing complexity (in $10^6$ Eur)

The x-axis represents architectural complexity as a result of variability of requirements and can be attributed to the number of optional functions times the number of models derived from the respective product line architecture.

The costs related to vehicle production are several times the costs of development and verification. Given average production costs of E/E components in the range of 4.000€ to 5.000€, depending on configuration and 1.000.000 vehicles produced, the target for optimization through adaptation to a reduced set of requirements lies on the order of $10^9$ Euro, in the extreme case, if the E/E system was individually tailored to the vehicle configuration. These cost savings are almost completely realized with current modular or variant designs.

The additional costs of architecture complexity, such as product-line specific development, verification and validation costs as well as warranty costs, lie on the order of several $10^8$ Euro and double in the extreme case, but can be kept within limits with moderate complexity. Since the depicted relationship is qualitative, a figure for moderate complexity is hard to define, but taking today's architectures with 10 derivatives and 30 optional or alternative functions as a landmark, these can be placed in the center of x-axis in Figure 50 or figure 51 respectively.

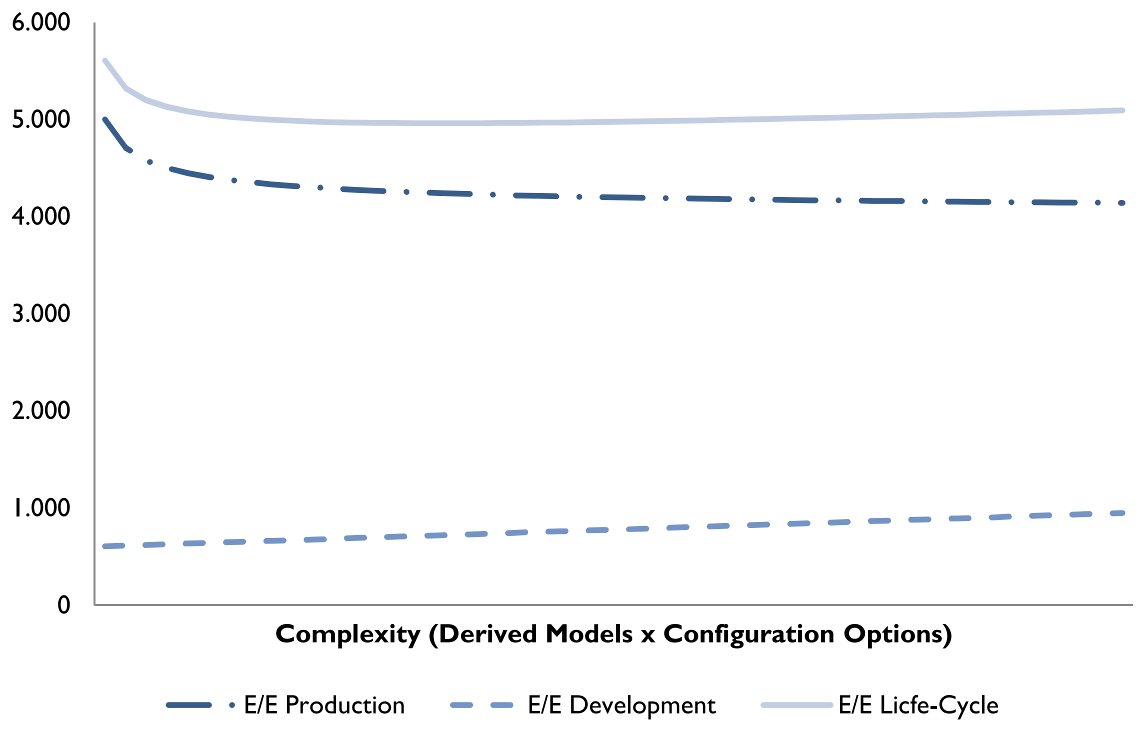


Figure 51 Life-cycle costs of modular architectures with increasing complexity (in $10^6$ Euro)

With modular architectures depicted in Figure 51, the picture is similar regarding the involved economic scales. Likewise major savings can be realized by exploiting requirement variability and adaptation of the E/E system, however as outlined in chapter 5 the overall costs remain higher than with centralized architectures. The costs of complexity on the other hand scale better with increasing number of components, due to the reduced number of alternative system variants to be tested and potentially increased reuse of components.

Since current designs exhibit both modular and centralized characteristics, the state of current architecture design can be located somewhere between the two diagrams, at a point at which the degree of interconnection and variability can still be handled by development processes. Further efficiency gains through exploitation of variability of requirements and models will be compensated by increased costs of complexity, i.e. limited reuse of components, increased costs of verification and reduced test coverage.

The goal for architecture design and hence the subject of architecture evaluation should target two of the depicted relationships: reduce the extent or re-implementation and variants and thus make the curve of development and verification less steep and centralize the fraction of non-variable components and extend the communality base across distinct product lines thus reducing costs of production to a level attained by centralized architectures. This requires an active management of requirements variability within the architecture.

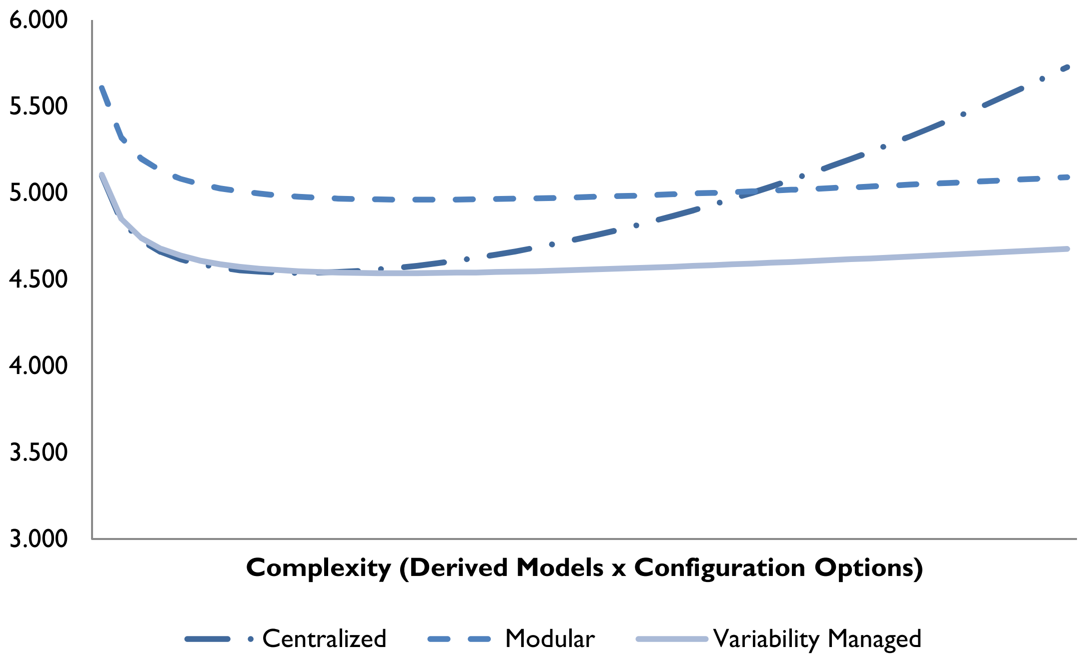


Figure 52 Life-cycle costs of modular, centralized and variability managed architectures (in $10^6$ Euro)

With variability managed architectures, E/E systems can be optimized with respect to costs of development and costs of production simultaneously as depicted in Figure 52, resulting in linear costs of complexity and reduced costs of production at any given level of variability and interdependencies. Life-cycle costs can be kept almost constant even with an increasing number of product lines and individual differentiation of derivatives and vehicle specific feature configurations.

The prerequisite for an adequate management of architecture variability is the availability of models reflecting requirements variability and dependencies, i.e. the higher layers of the architecture. The resulting means of architecture optimization are then the centralization of common, invariant functionality on central components and externalization of individual functions on separate, modular components.

## 8.3 Open Issues for Further Research

In the process of this work, we have applied generic assumptions about the architecture and cited incidental evidence for the postulated relationship between architectural structure and quality, based on documented architecture cases and interviews. Further research on that subject should aim at gaining a realistic picture of the variability and dependencies of functional requirements and their realization and at improving the quality of evidence to for the observed relationships.

In order to improve the accuracy of the architecture model a more in-depth study of the individual functions and their representation in the logical architecture is necessary. Whereas the presented work has considered dependencies in the logical architecture to be defined by the functional requirements of applications and immutable to architectural design, an investigation of the refinement of functional requirements into logical design of event chains will allow more founded conclusions about functional dependencies between individual functions and potential for reduction of external communication and interface variability. In effect this will lead to a refined level of abstraction within architecture evaluation down to software modules. The consideration of generic functional primitives as part of architecture and software modules as units of reuse will enable more synergies in development through architectural decisions. Secondly an analysis of the dependency of functions on physical resources and the interconnection between components realizing distinct functions should reveal, what degree of architecture variability is really required.

Regarding the postulated relationship between structure and quality outlined in chapter 5, evidence in this work was based on individual, documented examples and expert interviews. Empirical evidence is limited by the fact that concrete projects differ in more than just architectural structure. It is unlikely that a statistical analysis of development projects will yield satisfactory results; rather further work in this field ought to improve the quality of data gathered through interviews and make development projects comparable by classifying functional size and complexity of the involved functional changes.

The last and most fundamental point to consider in future work on this subject is to extend the analysis from a negative definition of quality in terms of costs and risks to a business value and potential oriented definition. This shift of perspective should change the focus of architectural design from optimization for costs under given constraints to optimization for business value under uncertainty of customer’s preferences and individual perception of value.

# References

**A2A** Automotive Industry Seeking Electronic Solutions to Four Main Issues [Online] // Auto to Electro Forum. - May 31, 2010. - May 31, 2010. - http://www.a2e.com.

**Averbeck Timo** Kostenmanagement im Prozess der Absicherung und Erprobung ni der Automobilindustrie - Effekte und Einsparpotenziale durch Baukästen [Master Thesis]. - Technische Unversität Kaiserslautern : , 2006.

**Avizienis Algirdas [et al.]** Basic Concepts and Taxonomy of Dependable and Secure Computing [Journal] // IEEE Transactions on Dependable and Secure Computing. - Los Alamitos : IEEE Computer Society, 2004. - 1 : Vol. 1. - pp. 11 - 33.

**Axelsson Jakob** Cost Models for Electronic Architecture Trade Studies [Conference] // Sixth IEEE International Conference on Complex Computer Systems (ICECCS'00). - Tokyo : IEEE Computer Society, 2000. - p. 229.

**Axelsson Jakob** Evolutionary Architecting of Embedded Automotive Product Lines: An Industrial Case Study [Conference] // Joint Working IEEE/IFIP Conference on Software Architecture, European Conference on Software Architecture. WICSA/ECSA 2009. - Cambridge : [s.n.], 2009. - pp. 101 - 110.

**Axelsson Jakob** HW/SW Codesign for Automotive Applications: Challenges on the Architecture Level [Conference] // Fourth International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC). - Magdeburg : [s.n.], 2001. - p. 121.

**Badstuebner Jens** Billig ist nicht Low-Cost [Journal] // Automobil Industrie. - 2008. - Vol. 6. - pp. 32-33.

**Bahsoon Rami and Emmerich Wolfgang** Evaluating Software Architectures: Development Stability, and Evolution [Conference] // Proceedings of the ACS/IEEE International Conference on Computer Systems and Applications. - Los Alamitos : IEEE Computer Society Press, 2003. - pp. 47 - 56.

**Baldwin Carliss Y. and Clark Kim B.** Modularity in the Design of Complex Engineering Systems [Book Section] // Complex Engineered Systems: Science Meets Technology / book auth. Ali Minai Dan Braha and Yaneer Bar Yam. - New York : Springer-Verlag, 2006.

**Banerjee Prithviraj and deWeck Olivier L.** Flexibility Strategy – Valuing Flexible Product Options [Conference] // Proc INCOSE ICSE Conference Synergy Between Systems Engineering and Project Management. - 2004.

**Barbacci Mario [et al.]** Quality Attributes [Report]. - Pittsburgh, PA : Software Engineering Institute, Carnegie Mellon University, 1995.

**Barbacci Mario R. [et al.]** Steps in an Architecture Tradeoff Analysis Method: Quality Attribute Models and Analysis [Report] / Software Engineering Institute, Carnegie Mellon University. - Pittsburgh, PA : Software Engineering Institute, Carnegie Mellon University, 1998.

**Barner Joerg** A Lightweight Formal Method for the Prediction of Non Functional System Properties [PhD Thesis]. - Erlangen : Unversität Erlangen, 2005.

**Basili Victor R.** Software modeling and measurement: the Goal/Question/Metric paradigm [Report]. - College Park : University of Maryland at College Park, 1992.

**Basili Victor R., Caldiera Gianluigi and Rombach H. Dieter** The Goal-Question-Metric Approach [Book Section] // Encyclopedia of Software Engineering. - New York : John Wiley and Sons, 1994.

**Bass Leonard J., Klein Mark and Bachmann Felix** Quality Attribute Design Primitives and the Attribute Driven Design Method [Conference] // PFE '01 Revised Papers from the 4th International Workshop on Software Product-Family Engineering. - London : Springer-Verlag UK, 2002. - pp. 169-186.

**Berander Patrik [et al.]** Software quality attributes and trade-offs [Report]. - Bleklinge : Lars Lundberg, Michael Mattsson, Claes Wohlin, 2005.

**Bettencourt da Cruz David** POSAAM - Eine Methode zu mehr Systematik und Expertenunabhängigkeit in der qualitativen Architekturbewertung [PhD Thesis]. - Online : http://d-nb.info/996917705 , 2009.

**Bielefeld** Bielefeld, Offene Systemarchitekturen und Modellbasierung, Komplexitätsbeherrschung in der Automobilinsdustrie // BMW Presentation. - München : BMW Group, 2005.

**Blanco Sebastian** How much does software add to the cost of today's vehicles? How about tomorrow's electric cars? [Online] // Green Autoblog. - Jun 08, 2010. - http://green.autoblog.com/2010/06/08/how-much-does-software-add-to-the-cost-of-todays-vehicles-how/.

**BMW Group** BMW Vehicle Configurator Web Site [Online]. - 2011. - Mai 2011.

**Bondavalli Andrea [et al.]** Dependability Analysis in the Early Phases of UML Based System Design [Journal] // Journal of Computer Systems Science and Engineering. - 2001. - Vol. 16. - pp. 265-275.

**Bosch Jan and Bengtsson PerOlof** Assessing Optimal Software Architecture Maintainability [Conference] // Proceedings of the Fifth European Conference on Software Maintainability and Reengineering. - Lisbon : IEEE Computer Society, 2001. - p. 168.

**Broy M., Deissenböck F. and Pizka M.** A Holistic Approach to Software Quality at Work [Conference] // 3rd World Conference for Software Quality. - 2005.

**Broy Manfred [et al.]** Automotive Architecture Framework: Towards a Holistic and Standardised System Architecture Description [Article] // Computer. - [s.l.] : IEEE Computer Society, 2009. - 12 : Vol. 42 . - pp. 98 - 101.

**Broy Manfred [et al.]** Software intensive systems in the automotive domain: challenges for research and education [Book Section] // SAE world congress. - [s.l.] : SAE, 2006.

**Broy Manfred [et al.]** Umfassendes Architekturmodell für das Engineering eingebetteter Softwareintensiver Systeme. [Report] / Technische Universität München. - 2008.

**Broy Manfred** Service Oriented Systems Engineering: Specification and design of Services and Layered Architectures - The Janus Approach. [Conference]. - 2005.

**Broy Manfred, Deissenboeck Florian and Pizka Markus** Demystifying Maintainability [Conference] // Proceedings of the 2006 international workshop on Software Quality. - New York, NY : ACM, 2006. - pp. 21-26.

**Chen DeJiu and Törngren Martin** A Metrics System for Quantifying Operational Coupling in Embedded Computer Control Systems [Conference] // EMSOFT '04 Proceedings of the 4th ACM international conference on Embedded software . - New York, NY : ACM, 2004. - pp. 184-192.

**Crosby P.B.** Quality is free: the art of making quality certain [Book]. - New York : McGraw-Hill, 1979.

**Dannenberg J. and Kleinhans C.** The Coming Age of Colaboration in the Automotive Industry [Article] // Mercer Management Journal. - 2004. - 18.

**Davis L., Gamble R. F. and Payton J.** The impact of component architectures on interoperability [Journal] // J. Syst. Softw.. - New York, NY : Elsevier Science Inc., 2002. - 1 : Vol. 61. - pp. 31-45.

**Deissenböck Florian, Wagner Stefan and Pizka Markus** Kostenbasierte Klassifikation von Qualitätsanforderungen [Conference] // SE Konferenz 07. - 2007.

**Deming W. E.** Out of the Crisis: Quality, Productivity and Competitive Position [Book]. - Cambridge, MA : MIT Press, 2000.

**Differding Christiane, Hoisl Barbara and Lott Christopher M.** Technology Package for the Goal Question Metric Paradigm [Report] / Universität Kaiserslautern, Fraunhofer Institute for Experimental Software Engineering. - 1996.

**Dinkel Michael and Baumgarten Uwe** Modeling Nonfunctional Requirements: a Basis for Dynamic Systems Management [Conference] // Proceedings of the Second International Workshop on Software Engineering for Automotive Aystems . - New York : ACM, 2005. - pp. 1-8.

**Dobrica Liliana and Niemelä Eila** A Strategy for Analzing Product Line Software Architectures [Report] / VTT Technical Research Centre of Finland. - 2000.

**Dobrica Liliana und Niemelä Eila** Attribute Based Product LIne Architecture Development for Embedded Systems [Conference] // Proceedings of the 3rd Australasian Workshop on Software and Systems Architectures. - Sydney : IEEE, 2000.

**Dromey R. Geoff** A Model for Software Product Quality [Article] // IEEE Transactions on Software Engineering. - [s.l.] : IEEE Press, 1995. - 2 : Vol. 21. - pp. 146-162.

**Dudenhöffer Ferdinand** Ausfallrate durch Elektrik/Elektronik steigt weiter [Article] // ATZ Automobiltechnische Zeitschrift. - München : Carl Hanser Vlg., 2004. - Vol. 11/2004.

**EE-VERT Consortium** Energy Efficient VEhicles for Road Transport [Report]. - 2009.

**Egyed A., Medvidovic N. and C.Gacek** Component-based perspective on software mismatch detection and resolution [Conference]. - 2000. - Vol. 147.

**Ehrlenspiel Klaus, Kiewert Alfons und Lindemann Udo** Kostengünstig Entwickeln und Konstruieren [Buch]. - Berlin Heidelberg : Springer-Verlag, 2007.

**Engel Avner und Browning Tyson R.** Designing systems for adaptability by means of architecture options [Journal] // Systems Engineering. - 2008. - Bd. 11. - S. 125-146.

**Eppinger Steven and Salminen Vesa** Patterns of Product Development Interactions [Conference] // International Conference on Engineering Design. - 2001. - pp. 283-290..

**Eppinger Steven D and Salminen Vesa** Patterns of Product Development Interactions [Conference] // Proceedings of the 13th International Conference on Engineering Design. - 2001.

**Feigenbaum A. V.** Total Quality Control [Book]. - New York : McGraw-Hill Professional, 1983.

**Felskau Peter** Bewertung von IT-Architekturen im Fahrzeug [Master Thesis]. - Karlsruhe : Universität Karlsruhe (TH), 2006.

**Fenton Norman E and Ohlsson Niclas** Quantitative Analysis of Faults and Failures in a Complex Software System [Journal] // IEEE Transactions on Software Engineering. - 2000. - Vol. 26. - pp. 797-814.

**Fowler Martin** Patterns of Enterprise Application Architecture [Book]. - Amsterdam : Addison-Wesley Longman, 2002. - ISBN 0321127420 .

**Freund U., von der Beeck, M., Braun, P., and Rappl, M.** Architecture Centric Modeling of Automotive Control Software [Conference] // SAE 2003 World Congress & Exhibition. - [s.l.] : SAE International, 2003.

**Frischkorn Hans-Georg** Automotive Architecture Requirements // Summer School Architectural Paradigms for Dependable Embedded Systems. - Vienna : Technische Universität Wien, 2005.

**Frischkorn Hans-Georg** In Architekturen Denken [Interview]. - http://www.dspace.de/shared/data/pdf/company/nl/d6_in_architekturen_denken.pd : dSPACE, Juli 20, 2007.

**Fröberg Joakim** Engineering Automotive Electronic Systems: Decision Support for Successful Integration [Report] / Mälardalen University. - 2007.

**Fröberg Joakim, Wallin Peter and Axelsson Jakob** Towards Quality Assessment in Integration of Automotive Software and Electronics: An ATAM approach [Conference] // Proceedings of the 6th Conference on Software Engineering and Practice in Sweden. - Umeå, Sweden : Umeå University, 2006.

**Funk Jeffrey L.** The Product Life Cycle Theory and Product Line Management: The Case of Mobile Phones [Journal] // IEEE Transations on Engineering Management. - 2004. - Bd. 51. - S. 142-152.

**Ghosal A. [et al.]** An Initial Study on Monetary Cost Evaluation for the Design of Automotive Electrical Architectures [Journal] // SAE International. - 2008.

**Goseva-Popstojanova Katerina and Trivedi Kishor S.** Architecture-based approach to reliability assessment of software systems [Journal] // Performance Evaluation. - 2001. - Vol. 45. - pp. 179-204.

**Gunzert Michael** Komponentenbasierte Softwareentwicklung für sicherheitskritische eingebettete Systeme [PhD Thesis]. - Stuttgart : Universität Stuttgart, 2003.

**Gustavsson Håkan and Axelsson Jakob** Evaluating Flexibility in Embedded Automotive Product Lines Using Real Options [Conference] // 12th International Software Product Line Conference, 2008. SPLC '08.. - Limerick : IEEE Computer Society, 2008. - pp. 235-242.

**Gustavsson Håkan and Persson Erik** An Industrial Case Study of Design Methodology and Decision Making for Automotive Electronics [Conference] // ASME 2008 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference (IDETC/CIE2008). - New York : ASME, 2008.

**Gustavsson Håkan und Sterner Jan** A Framework For The Evaluation Of Resource Efficiency In Automotive Embedded Systems [Conference] // In Proceedings of the ASME 2008 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference. - New York : ASME, 2008.

**Häggander D. [et al.]** Maintainability Myth Causes Performance Problems in Parallel Applications [Report] / Department for Software Engineering and Computer Science. - Ronneby : University of Karlskrona, 1999. - pp. 288-294.

**Hamdan A. [et al.]** Erfahrungen mit der Messung der Wartbarkeit von Steuergeräte-Software [Journal] // Informatik LIVE!. Beiträge der 35. Jahrestagung der Gesellschaft für Informatik. - Bonn : Gesellschaft für Informatik, 2005. - Vol. 2.

**Hansen Paul** Keynote Speech // 4th Integrated Electronic Solutions Forum. - Frankfurt : Mentor Graphics, 2010.

**Hansson Hans, Norstrom Christer and Punnekkat Sasikumar** Reliability Modelling of Time-Critical Distributed Systems [Conference] // FTRTFT-2000: Sixth International School and Symposium on Formal Techniques in Real-Time and Fault-Tolerant Systems,. - Pune, India : Springer Verlag, 2000. - pp. 94-105.

**Hartmann Nico** Automation des Tests eingebetteter Systeme am Beispiel der Kraftfahrzeugelektronik [PhD Thesis]. - Karlsruhe : Universität Karlsruhe, 2001.

**Helfand Gloria, Rogozhin Alex and McManus Walter** Automobile Industry Retail Price Equivalent and Indirect Cost Multipliers [Report] / U.S. Environmental Protection Agency. - 2009.

**Henttonen K. [et al.]** Integrability and Extensibility Evaluation From Software Architectural models - A Case Study [Journal] // Open Software Engineering Journal. - 2007. - Vol. 1.

**Hoffmann Martin** Quality Evaluation of Software Architecture with Application to OpenH.323 Protocol [Master Thesis]. - Jyväskylä : University of Jyväskylä, 2006.

**Hofmeister Christine [et al.]** Generalizing a Model of Software Architecture Design from Five Industrial Approaches [Conference] // Proceedings of the 5th Working IEEE/IFIP Conference on Software Architecture (WICSA). - 2005. - pp. 77--86.

**Horstmann Marc** Verflechtung von Test und Entwurf für eine verslässliche Entwicklung eingebetteter System im Automobilbereich [PhD Thesis]. - Braunschweig : Technische Universität Braunschweig, Institut für Verkehrssicherheit und Automatisierungstechnik, 2006.

**Iansiti Marco und Khanna Tarun** Technological Evolution, System Architecture and the Obsolescence of Firm Capabilities [Artikel] // Oxford Journals: Industrial and Corporate Change. - [s.l.] : Oxford University Press, 1996. - 2 : Bd. 4. - S. 333-361.

**IEEE and ISO/IEC** Systems and software engineering - Recommended practice for architectural description of software-intensive systems. - New York : IEEE, 2007. - pp. c1--24.

**Ishikawa K.** What is Total Quality Control? - The Japanese Way [Book]. - Englewood Cliffs, NJ : Prentice-Hall, 1985.

**ISO** ISO 9126-1:2001, Software engineering - Product quality, Part 1: Quality model. - 2001.

**Juran J.M.** Juran's Quality Control Handbook [Book]. - New York : McGraw Hill, 1988.

**Kazman Rick [et al.]** ATAM: Method for Architecture Evaluation [Report] / Software Engineering Institute. - Pittsburgh, PA : Carnegie Mellon University, 2000.

**Kebemou Augustin** A Partitioning-Centric Approach for the Modeling and the Methodical Design of Automotive Embedded Systems Architectures [PhD Thesis]. - Berlin : Technische Universität Berlin, 2008.

**Knox Stephen T.** Modeling the Cost of Software Quality [Journal] // Digital Tech. J.. - Acton, MA : Digital Equipment Corp., 1993. - Vol. 5. - pp. 9-17.

**Kröll Markus** Methode zur Technologiebewertung für eine ergebnisorientierte Produktentwicklung [PhD Thesis]. - Stuttgart : Universität Stuttgart, 2006.

**Lankford J.** Measuring system and software architecture complexity [Conference] // Proceedings of the IEEE Aerospace Conference, 2003. - 2003. - Vol. 8.

**Laprie Jean-Claude and Randell Brian** Basic Concepts and Taxonomy of Dependable and Secure Computing [Journal] // IEEE Trans. Dependable Secur. Comput.. - Los Alamitos, CA : IEEE Computer Society Press, 2004. - Vol. 1. - pp. 11-33.

**Larses Ola** Applying Quantitative Methods for Architecture Design of Embedded Automotive Systems [Conference] // Proceedings INCOSE International Symposium. - 2005.

**Larses Ola** Architecting and Modeling Automotive Embedded Systems [PhD Thesis]. - Saarbrücken : VDM Verlag Dr. Mueller e.K., 2005.

**Larses Ola** Modern Automotive Electronics from a Dependable systems perspective [Report] / Mechatronics Lab, Department of Machine Design Royal Institute of Technology, KTH. - Stockholm : Royal Institute of Technology, 2003.

**Larses Ola** Modern Automotive Electronics from an OEM perspective [Bericht] / Mechatronics Lab, Department of Machine Design Royal Institute of Technology, KTH. - Stockholm : Royal Institute of Technology, 2003.

**Magna** The Fully Networked Car - Trends in Car Communication. - Geneva : [s.n.], 2005.

**Mårtensson Frans** Software Architecture Quality Evaluation, Approaches In an Industrial Context [Report] / Blekinge Institute of Technology. - 2006.

**Matinlassi Mari, Niemelä Eila and Dobrica Liliana** Quality-driven Architecture Design and Quality Analysis Method [Report] / VTT Technical Research Centre of Finland. - 2002.

**McCall Jim A., Richards Paul K. and Walters Gene F.** Factors in Software Quality [Report]. - Sunnyvale CA : General Electric Co, 1977.

**Navet Nicolas [et al.]** Trends in Automotive Communication Systems [Artikel] // Proceedings of the IEEE. - 2005. - 6 : Bd. 93.

**Navet Nicolas [et al.]** Trends in Automotive Communication Systems [Conference] // Proceedings of the IEEE. - 2005. - Vol. 93. - pp. 1204-1223.

**Negele Herbert and Pfletschinger, Thilo and Wenzel, Stefan and Getto, Gerhard** Definition and Application of Consistent Risk Management in Automotive System Development [Presentation]. - Bremen : Gesellschaft für Systems Engineering, 2005.

**Negele Herbert** Systems Engineering Challenges and Solutions from an Automotive Perspective // Keynote Presentation, INCOSE Symposium 2006. - 2006.

**Neufville [et al.]** Screening for Real Options “In” an Engineering System [Journal] // Processdings of 2006 International Symposium on Systems Engineering. - 2006.

**Nord Robert [et al.]** Integrating the Architecture Tradeoff Analysis Method with the Cost Benefit Analysis Method [Report] / Carnegie Mellon Software Engineering Institute. - 2003.

**Obermaisser Roman and Tagliabo Fulvio** An Integrated Architecture for Future Car Generations [Article] // Realtime Systems: Special Issue on Component-Oriented Dependable Real-Time Systems . - München : Springer Science and Business Media, 2005. - 1-2 : Vol. 36.

**Olejniczak Robert** Systematisierung des funktionalen Tests eingebetteter Software [PhD Thesis]. - München : Technische Universität München, 2007.

**Ozkaya Ipek, Kazman Rick and Klein Mark** Quality-Attribute-Based Economic Valuation of Architectural Patterns [Report] / Software Engineering Institute. - 2007.

**Parnas David L.** On the Design and Development of Program Families [Article] // IEEE Transactions on Software Engineering. - 1976. - Vol. 1.

**Reichart Günter** Systems Engineering – ein neues Entwicklungsparadigma für die Automobilindustrie? [Conference] // 9. Euroforum Jahrestagung. - 2005.

**Reif** Automobilelektronik. Eine Einführung für Ingenieure [Book]. - Wiesbaden : Vieweg Friedr. + Sohn, 2006.

**Rhines Walden C.** System Approaches to Integration of Automotive Electronic Components // Keynote Speech at Integrat Integrated Electronic Solutions Forum. - 2007.

**Rhodes Donna H., Valerdi Ricardo und Roedler Garry J.** Systems Engineering Leading Indicators for Assessing Program and Technical Effectiveness [Journal] // Systems Engineering. - 2008. - Bd. 12. - S. 21–35.

**Rumpf-Steppat Oliver J. [et al.]** System-Integration, Verification and Release of Complex in-vehicle Networks: Methodology and Process [Journal] // SAE International. - 2005.

**Sangiovanni-Vincentelli A. and Martin G.** Platform-based design and software design methodology for embedded systems [Journal] // Design and Test of Computers, IEEE. - 2001. - Vol. 18. - pp. 23-33.

**Schiffauerova Andrea and Thomson Vince** A Review of Research on Cost of Quality Models and Best Practices [Journal] // International Journal of Quality and Reliability Management. - 2006. - Vol. 23. - pp. 647-669.

**Schlosser Joachim** Architektursimulation verteilter Steuergerätesysteme [PhD Thesis]. - Berlin : Logos Verlag, 2007.

**Schorer Stefan** Architekturbewertung des Daten-Bordnetzes in der Fahrzeugtechnik [Master Thesis]. - München : Fachhochschule München, 2007.

**Shewhart W. A.** Economic control of quality of manufactured product [Book]. - Milwaukee : American Society for Quality Control, 1931.

**Tahan Meir and Ben-Asher Joseph** Modeling and Optimization of Integration Processes Using Dynamic Programming [Journal] // Systems Engineering. - 2008. - Vol. 11. - pp. 165-185.

**Thiel Steffen [et al.]** A Case Study in Applying a Product Line Approach for Car Periphery Supervision Systems [Journal]. - [s.l.] : SAE International, 2001. - pp. 43-55.

**Thiel Steffen** A Framework to Improve the Architecture Quality of Software-Intensive Systems [PhD Thesis]. - Duisburg-Essen : Universität Duisburg-Essen, 2005.

**Thiel Steffen and Hein Andreas** Modeling and Using Product Line Variability in Automotive Systems [Journal] // IEEE Software. - 2002. - Vol. 19. - pp. 66-72.

**Tindell Ken [et al.]** Safe Automotive Software Development [Article] // Design, Automation and Test in Europe Conference and Exhibition. - [s.l.] : IEEE, 2003. - pp. 616 - 621 .

**Vyas Anant, Santini Dan and Cuenca Roy** Comparison of Indirect Cost Multipliers for Vehicle Manufacturing [Report] / Center for Transportation Research. - 2000.

**Wagner Stefan and Deissenboeck Florian** An Integrated Approach to Quality Modelling [Conference] // WoSQ'07: ICSE Workshops 2007. Fifth International Workshop on Software Quality. - [s.l.] : IEEE Computer Society, 2007. - p. 1.

**Wallin Peter and Axelsson Jakob** A Case Study of Issues Related to Automotive E/E System Architecture Development [Conference] // 15th IEEE International Conference on Computer Based Systems. - Belfast : IEEE Computer Society, 2008. - pp. 87-95.

**Wild Doris [et al.]** An Architecture-Centric approach towards the Construction of Dependable Automotive Software [Conference] // Proceedings of the SAE 2006 World Congress. - 2006.

**Willutzki Paul and Pigorsch Christian** Gesamtkostenanalyse verteilter Funktionen [BMW internal]. - 2006.

**Willutzki Paul** Einfluss der Kommunikations-Netzwerktopologie auf die Verfügbarkeit wichtiger verteilter Funktionen im L6-Bordnetz [BMW internal]. - 2006.

**Zehbold Cornelia** Lebenszykluskostenrechnung [Book]. - Wiesbaden : Springer Gabler , 2001. - ISBN 978-3409121538.