Igor Škorić

Life-Critical Software Development and Testing in the Automotive Industry

Master's Thesis

Graz University of Technology

Institute for Software Technology
Head: Univ.-Prof. Dipl.-Ing. Dr.techn. Wolfgang Slany

Supervisor
Univ.-Prof. Dipl.-Ing. Dr.techn. Franz Wotawa

External Supervisor
Dr. Mihai Nica (AVL List GmbH)

Graz, September 2014

Statutory Declaration

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

Graz, _____ _____

Date Signature

Eidesstattliche Erklärung[1]

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche kenntlich gemacht habe.

Graz, am _____ _____

Datum Unterschrift

[1] Beschluss der Curricula-Kommission für Bachelor-, Master- und Diplomstudien vom 10.11.2008; Genehmigung des Senates am 1.12.2008

To my parents

Jagoda and Sead

I am what I am
because you loved me

Acknowledgments

I would like to thank my supervisor Dr. Franz Wotawa for his remarks, his engagement and guidance throughout the learning process that was the creation of this thesis. I express my thanks to my external supervisor Dr. Mihai Nica for interesting me in the topic, giving insight into his domain and for his invaluable advice.

I do not have the eloquence to sufficiently describe my gratefulness towards my parents, Jagoda and Sead Škorić, who have given me everything that a child could ever wish for and without whose unending love my beautiful life and my academic journey would not have been possible. I send my love and thanks to my sister Sanela for always being the crumb to my cookie and bringing me joy since she was born. Furthermore, I would like to thank Faruk Bajramović for being a true friend for longer than I can remember. Finally, I would like to thank Barbara Reichhardt for being my loving partner, supporting me in good and bad and putting up with me for all these years. I love you all.

Abstract

This thesis aims to analyze the available literature about software development and reliability in the automotive industry from the last five years (2008-2013, first half of 2014). The goal is to give insight into the state of the art, new developments and the general direction in which the industry might be moving in the future.

The thesis is organized in four parts. Part I gives an insight into the situation in the industry, the problems and the solutions that are being worked on. Then, the methods used to gather and analyze the data are presented. Part II seeks to reconstruct the development process that is being used at different major automotive companies and to paint a picture of their development process, future directions and challenges. Part III takes the separate processes and consolidates them into one generalized process, then identifies interesting methods that can be isolated from that and explains them. It also rates the methods on their deployment status, source reliability and evaluates if they can be classified into the categories of *Prevention*, *Validation & Verification* or *Prediction*. Part IV finally concludes the thesis with a recapitulation and personal insights.

The evaluation shows that the automotive industry is very advanced and state-of-the-art in hyper-validated high quality software and continually improving. It shows the ways that the companies use their prowess to lift up supporting fields with transitive reliability benefits for the end product, like thorough requirements management, traceability of requirements and changes and efficient and effective testing efforts. It further shows that there is clear consensus on the right direction for the future.

Zusammenfassung

Das Ziel dieser Arbeit ist die Erforschung und Analyse der verfügbaren Literatur der vergangenen fünf Jahre (2008-2013, erste Hälfte 2014) zum Thema der Softwareentwicklung und Softwarezuverlässigkeit in der Automobilindustrie. Der Zweck ist die Vermittlung des Standes der Technik und neuer Entwicklungen. Es soll weiter eine Prognose über die zukünftige Entwicklung gestellt werden.

Die Arbeit besteht im Wesentlichen aus vier Teilen. Teil I gibt Einblicke in die Situation in der Industrie und nennt Probleme und Lösungen die Anwendung finden. In Teil II wird versucht den Entwicklungsprozess in verschiedenen Unternehmen zu rekonstruieren und darzustellen. Es soll ein Bild des Entwicklungsprozesses und der damit verbundenen Probleme und Lösungen entstehen. Aus den gewonnenen Einsichten werden in Teil III Gemeinsamkeiten erarbeitet und zu einem Gesamtbild der Industrie verschmolzen. Dazu werden interessante Methoden isoliert, erklärt und in einem einheitlichen Schema bewertet. Die Zuverlässigkeit der Quellen wird genauso beurteilt wie der Grad der Umsetzung im Regelbetrieb. Die Zugehörigkeit der Methoden zu den Kategorien *Prävention*, *Verifikation & Validierung* oder auch *Voraussage* wird entschieden. Zuletzt wird im Teil IV die Arbeit zum Abschluss gebracht und persönliche Eindrücke vermittelt.

Die Auswertung zeigt, dass die Autoindustrie sehr fortgeschritten ist, vor allem im Bereich der Entwicklung von hochverifizierter Qualitätssoftware für Eingebettete Systeme. Ein kontinuierliches Streben nach Verbesserung ist zu erkennen. Es wird gezeigt, wie die Entwicklung von Fähigkeiten in diesem Bereich auch gleichzeitig den Ausbau unterstützender Felder wie Anforderungsmanagement, Verfolgbarkeit von Anforderungen und Änderungen und effektive Testmethoden vorantreibt. Weiter wird gezeigt, dass es in der Industrie einen eindeutigen Konsensus über den besten Weg in die Zukunft gibt.

Contents

List of Figures

Part I.

Introduction and Preliminaries

1. Thesis Introduction

In a time when the gap in manufacturing quality is closing, software components have become an important distinguishing factor between different brands and manufacturers and a competitive advantage. Not only does software improve efficiency and monitoring, but software-supplied or software-supported functionality is very important in customer-relevant parts of products. It seems that the average computer user has become accustomed to buggy software and receiving patches on a daily basis. This kind of software release policy is absolutely unthinkable in safety-critical, reliability-dependent applications like the automotive industry, where a software failure may cost many lives and in one sweep destroy a company's reputation and finances. At the same time the industry is pushed towards more complex software in their products, because software driven functions are a major way of innovation.

The situation presents itself as a irresolvable dilemma: How to increase functionality (and with it the complexity) of software while at the same time keeping software bugs to a minimum? Some studies estimate the amount of software in a modern car to 100 million lines of code (LOC) with an order of magnitude growth per decade [27] (also cited by Bosch [1]). If that estimate is correct, the current size would be about 500 million LOC. If only 1% of that source code is in safety critical software, that would still leave 5 million LOC of source code that needs to be as close to perfect as possible. For comparison, the Linux Kernel - broadly considered to be a finely engineered well maintained quality source code - has about 8 million LOC and had 3299 new medium and high impact defects identified in the year 2013 alone. Proprietary code seems to perform even worse on average [33]. To expect a single company to outperform that metric by multiple orders of magnitude using the same methods seems quite unrealistic. This is why safety-critical applications have moved towards model-driven software

development. It offers a level of abstraction that allows for less error-prone software components that are easier to test and re-use and in some cases even make it possible to *prove* the correctness of software components.

The automotive industry seems to have become a paragon of model-driven state-of-the-art software development over the last few years, especially the German premium vehicle manufacturers and their suppliers. Since the German government made the V-Model mandatory for its IT projects in 2005 [69] it has gained traction in the industry and with it Model Based Development (MBD). Armed with the necessary resources and stereotypically German engineering prowess it seems that the companies are making this new model-driven process work. This begs the question: How do they make it work? What are the best practices and the future prospects for MBD? This thesis takes a look at the software development processes of a handful of companies, including the *big three* premium consumer vehicle manufacturers Audi, BMW and Daimler and attempts to provide an answer to these questions.

1.1. Goals of the Thesis

The goal of this work is to give an overview of the relevant methods for software reliability in the life cycle of automotive embedded software. This task can be separated into items as follows:

- Overview and description of methods
- Classification of methods regarding
  - State-of-the-Art vs. research
  - Positioning within the V-Model
  - Relevance for
    - Prevention
    - Verification & validation
    - Monitoring
    - Prediction of reliability
    - Maintenance
- Possibly Overview of relevant tools

- Possibly statistics about software-induced failure

The individual points have found different levels of detail and certainty depending on the degree of information found in the available document collection.

Part II provides relevant information specific to an individual company. Classification of *State-of-the-Art vs. Research* is done by separating methods into the appropriate section within the company (State-of-the-Art is found in the *Recent History and Current State* section, while *Research* is found in the *Future Directions and Challenges* section). The relevant tools that the company uses are mentioned within the explanation of the process.

Part III provides the *positioning of methods into the phases of the V-Model* in the form of chapters. *Overview and description of methods* is given in a separate section within the chapter for each method. The relevant tools are mentioned as a comparison across companies.

Finally, a scoring system is applied that evaluates *State-of-the-Art* and *Source Reliability* on a 5-part scale and the relevance for *Prevention*, *Verification & Validation* and *Prediction of Reliability* as a binary value. For more on the Rating system please see sec. 14.2.

Monitoring and *Maintenance* are given their own chapters in part III rather than including them in the rating system. These two categories added little benefit as a rating value for methods of other phases but stood naturally on their own as a category.

1.2. The Process

As a first step, a literature gathering and data mining process is executed. The process uses on-line digital libraries like SpringerLink [132] to collect papers that fit selected criteria. Those papers are downloaded and processed. This collection process continues on-demand in iterations throughout the thesis creation whenever additional information is required. An explanation of the details is found in chapter 2.

Next, a review and analysis process is performed which aims to find interesting documents, assign them to companies and identify connections between them. Companies with insufficient information are pruned. Those that remain are considered in separate chapters and presented in part II. The goal is to merge the information from different sources in an effort to create the best possible overview of the company. Every company is given a short introduction in the first section of the chapter. Next, (provided sufficient information) the relevant company structure and the complete software development process for each company is explained. Each step of the process (usually phases of the V-Model) is presented in separate sub-sections. The section after that contains the research, future plans and challenges of the company. Finally, available details about the authors of the documents (only employees considered) are listed for reference.

In part III the separate companies are combined and a generalized process based on the information from part II is created. This common process again consists of the phases in the V-Model (used as a skeleton), which is extended with two additional phases called *Monitoring* and *Maintenance* and filled out with information on each phase in its own chapter. General information on every particular phase is given at the start of its chapter. After that, interesting methods in that phase are isolated and explained in distinct sections.

Every Method is selected on the property that it can be reasonably cleanly separated from other methods and that it represents an interesting aspect worth exploring. It is then explained in general and sourced to the more specific company methods from which it is generalized. Finally, it is rated on different properties which are explained in section 14.2.

2. Preliminaries

This chapter contains an explanation of the process used to select, obtain and analyze the source material used to create this work. At the beginning of investigation, it was necessary to have an orientation phase and try to identify a rough overview of the current situation. Because this is an unusual domain for the author it was necessary to learn about the basics first. Interesting points were the identification of major actors in the field, the kind of information that is available and what the best practices are. This was achieved by browsing the knowledge base and reading recent papers while looking up any information that was not familiar.

A number of things seemed to stand out quickly:

- The V-Model XT was declared the official development model and made mandatory for projects of the German government in 2005 [69]. This seems to be one reason for the broad use of it, especially in German companies which seek to do business with the government.
- Adoption of Model Based Development (MBD) seems to be promoted and/or accelerated by the use of the V-Model [68].
- Adoption of extensive multi-level iterative testing seems to be promoted in part by the use of the V-Model
- German manufacturers publish large portions of information exclusively in the German language and exclusively in German magazines, presumably to slow down digestion from competition in the USA, Japan and especially China

All of these points made it promising to focus on the German automotive industry and offer companies from other geographical areas as a contrast. Because the author is not hindered by the language barrier, the inclusion of German publications makes for a potentially greater insight while not exceeding the scope of a master's thesis. Additionally, the decision made

it easy to provide a guiding structure for the thesis, which the V-Model as almost universally used process offered.

Following from the focus, the selection of companies was also made easier. Germany hosts the biggest selection of premium vehicle manufacturers, including most of the top contenders: BMW, Volkswagen Group (Audi, VW) and Daimler AG. These are joined by three major automotive part suppliers with good information availability (ZF Friedrichshafen AG (ZF), Bosch and Siemens) and select major companies from other areas for a broader base and contrast (Ford, Toyota). Putting these together results in the complete list of companies included in this thesis (see chapter 3).

The majority of documents used were retrieved from the digital libraries SpringerLink [132] and IEEEXplore [44] through the institutional access provided by the Graz University of Technology to the author. Remaining documents were gathered from miscellaneous sources like the ACM Digital Library [31], company websites and conference proceedings.

2.1. The generic V-Model

The closest thing to a generic V-Model might be the V-Model XT that is used by the German government and is maintained and developed by the Industrieanlagen-Betriebsgesellschaft mbH (IABG) [69]. The first version of this model was released in 1992 and it was continually extended and improved since then.

On one hand, this model has the advantage of being publicly available and free to use. On the other hand it is far too complex and partly out of scope, because big parts of the specification deal with the relationship between and responsibility of software development partners (OEMs and suppliers). The most prominent part of the model (the name-giving V-shape) is also the thing that was adapted broadly and is meant when the term V-Model is used in the rest of this work. A simple version with the most common features can be seen in fig. 2.1.

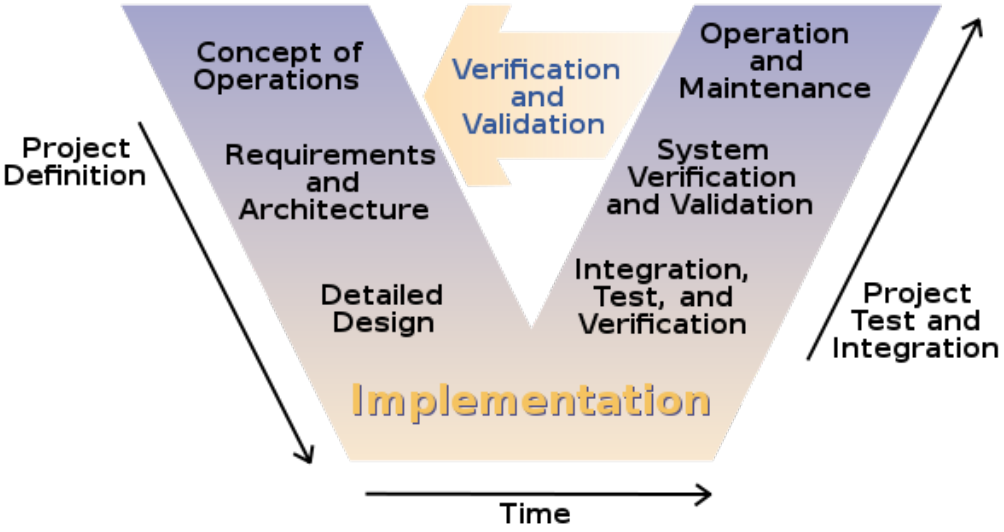

Figure 2.1.: Simple V-Model Illustration [109]

In its most basic form the V-Model describes top-down design approach in the beginning (left side, descending arm of the V) followed by implementation and then a bottom-up testing approach (right side, rising arm of the V) with integration at the completion of each step. The current state is always tested against the same level of definition on the other side of the V and a failed test means that testing is interrupted, the error is found and fixed on the left side and the process is repeated starting at the left side position where changes have been made.

2.2. Keyword Selection

The selection of keywords used for document acquisition grew and changed over time but can generally be reduced into a few notable categories.

- **Catch all**: Designed to target a set broad enough to include most of the relevant documents and used early in the document gathering to identify interesting features and serve as base for further, more

specific, queries.
Examples: *ECU*, *automotive*, *in the loop*, *maintenance*, *model based testing*, *monitoring*, *prediction*, *reliability*, *requirements engineering/management*, *(virtual) verification*

- **Technologies**: Designed to target specific technologies that were identified as relevant or interesting. Most of these were used in manual search but some have found entry into the main queries.
  Examples: AUTOSAR, Virtuelle Absicherungsplattform
- **German**: Designed to include German versions of common terms used in the relevant domain. This was necessary to leverage the advantage from german documents mentioned at the beginning of this chapter.
  Examples: *wartung*, *voraussage*, *zuverlässigkeit*, *steuergeräte*, *anforderungsmanagement*, *virtuelle Absicherungsplattform*

2.3. SpringerLink

To find interesting documents on SpringerLink it was necessary to craft a custom search string that was not possible to create with the search interface exclusively.

First Query

Listing 2.1: First Query

```
("ZF Friedrichshafen" OR "Audi AG" OR "BMW Group"
OR "Siemens Automotive" OR "Daimler AG"
OR "Ford Motor Company" OR "Robert Bosch" OR "Volkswagen"
OR "Volvo" OR "Honda" OR "Toyota")
```

This query aims to return all papers on the basis of the company name. Ambiguous company names like *Continental* that could match terms like "Continental Drift" or companies that have many divisions in different domains are disambiguated (e.g. *Siemens Automotive*). To ensure exclusion of absolutely unrelated papers the category filter was set to include only

Computer Science and *Engineering* documents. Unavailable papers were excluded with the *showAll* parameter and the year range was constrained to years from 2008 to 2013.

Listing 2.2: First Query Parameters

```
'showAll': false
'facet-start-year': 2008
'facet-end-year': 2013
'facet-discipline': '"Computer+Science"'
'facet-discipline': 'Engineering'
```

Second Query

The second query aims to replace the category filter with a keyword filter to avoid excluding relevant papers that were assigned into another category. The keywords contain English and German terms, because many relevant documents are not available in English. Disambiguated company names are used again. Finally, the term "ove" is excluded because of a large number if one-page industry news documents from the *OVE Nachrichten* (OVE News) organisation polluting the search results.

Listing 2.3: Second Query

```
("ECU" OR "requirements␣engineering" OR "in␣the␣loop"
OR "model␣based␣testing" OR "requirements␣management"
OR "virtuelle␣Absicherungsplattform" OR "steuergeräte"
OR "anforderungsmanagement" OR "virtual␣verification"
OR "wartung" OR "voraussage" OR "zuverlässigkeit"
OR "maintenance" OR "monitoring" OR "reliability"
) AND ("ZF␣Friedrichshafen"
OR "Audi␣AG" OR "Robert␣Bosch" OR "Siemens␣Automotive"
OR "Daimler␣AG" OR "Ford␣Motor␣Company" OR "BMW␣Group"
OR "Volkswagen" OR "Volvo" OR "Honda" OR "Toyota")
NOT ove
```

Because of a large number of results, the year range is contracted to only include years 2010 to 2013 for the second query.

Listing 2.4: Second Query Parameters

```
'showAll': false
'facet-start-year': 2010
'facet-end-year': 2013
```

Overlapping search results were overwritten because retrieved files have unique names. Therefore, duplicates were automatically removed.

2.4. IEEEXplore

At the beginning of document gathering the search queries used for IEEEXplore queries were still very undirected and unrefined. Among the first queries that were used were many with a single search term and variable time periods. Example queries from that time can be seen in listing 2.5. These queries were directed at finding out which *tools* were used, a direction that turned out to be less important later on in the thesis work.

Listing 2.5: Queries used at the start of document gathering

```
"AUTOSAR"
"automotive␣software␣safety-critical"
"automotive␣software␣testing␣tool"
"safety-critical␣software␣development␣tool"
"safety-critical␣software␣tools"
"automotive␣software␣testing"
```

Directed Query

After a short time it was discovered that it is possible to search for author affiliations for every document and enter a custom query with predefined terms. This made it possible to craft very effective queries and avoid unnecessary noise.

Listing 2.6: Refined query targeting affiliation

```
("Author␣Affiliations":ZF Friedrichshafen OR
"Author␣Affiliations":Audi OR
"Author␣Affiliations":BMW OR
"Author␣Affiliations":Siemens OR
"Author␣Affiliations":Daimler OR
"Author␣Affiliations":Ford OR
"Author␣Affiliations":Bosch OR
"Author␣Affiliations":Volkswagen OR
"Author␣Affiliations":Volvo OR
"Author␣Affiliations":Honda OR
"Author␣Affiliations":Toyota)
AND software
AND (automotive OR monitoring OR maintenance OR predict OR predicti
```

This query was limited to the years 2010 to 2013. Documents that were unavailable with the provided subscription (i.e. that were premium access only) were ignored.

2.5. Other Sources or Queries

Other queries to either SpringerLink or IEEEXplore were made on demand when the need arose to look up very specific documents or if it became clear that certain aspects need further investigation. If other sources like the ACM Digital Library or assorted websites were used to obtain supplemental documents not available elsewhere, then a separate storage was used for those as to keep everything in best possible order and to support refreshing the collection at a later time if necessary.

2.6. Management of Document Collection

The large number of documents (over 3000) made manual organization infeasible. A number of prominent software tools for document management

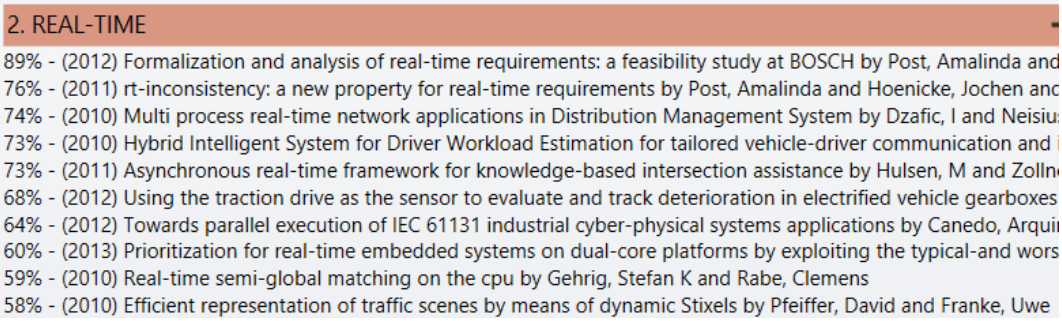

Figure 2.2.: Semantic Category Example

were evaluated and it was decided to use the tool Qiqqa (pronounced *quicker*) by Quantisle Ltd. [83]. The complete document collection was imported into Qiqqa for processing and evaluation. Qiqqa has multiple functions that allowed for an efficient work flow.

The tools that were used for this thesis are:

- OCR to make text in pictures and scanned PDF documents searchable
- **BibTeX Sniffer** to obtain accurate document meta data and create cross-references (e.g. other documents by this author)
- **Duplicate Detection** to find duplicates across source boundaries
- **Tagging** to assign documents to multiple groups
- **Annotations** like permanently highlighting words in a PDF with a virtual text marker for easier navigation and understanding anddocument annotations for indicating reading stage or importance of documents
- **Semantic Analysis** for grouping documents together based on the themes (word groups) they contain (see fig. 2.2)
- **Advanced Search** with support for fuzzy matching (run, runs, ran, running,...), proximity search (allows distance of X words between two search terms), ...

All of these functions made it easier to work with a large collection.

2.6.1. Tagging, Grouping and Cleaning

After all documents were imported, de-duplicated and generally cleaned up, it was necessary to group related documents together. This was done through the tagging system in Qiqqa. There were multiple types of groups used:

Tag Pattern	**Meaning**
C: [CompanyName]	Directly connected to [CompanyName]
S: [CompanyName]	References the company [CompanyName]
T: [ToolName]	References the software [ToolName]
L: [Language]	Is written in human language [Language]

Finding the appropriate documents to tag was made easier by search functions like "search only on first page" to find affiliations. After the desired tags were made the library was purged of untagged documents to mitigate the impact of undirected search and addition from IEEEXplore mentioned in section 2.4 and also from SpringerLink's inability to search by affiliation (e.g. hundreds of documents from people named after companies).

This left an ≈650 documents in the library which grew to ≈780 after miscellaneous supplemental documents and on-line sources were added on demand throughout the time of work.

Part II.

Literature Review

3. Part II Introduction

Part II of this thesis serves to record the findings specific to each of the companies that are included in it. The inclusion of a company is decided on three factors. The first factor is the size of the company's premium car sales numbers and the second is sales and revenue in total. The third factor is the degree of information volume in the available corpus. These criteria lead to the following selection:

1. Audi AG (chapter 4)
2. BMW Group (chapter 5)
3. Daimler AG (chapter 6)
4. ZF Friedrichshafen AG (chapter 7)
5. Siemens AG (chapter 8)
6. Bosch GmbH (chapter 9)
7. Volkswagen AG (chapter 10)
8. Ford Motor Company (chapter 11)
9. Toyota Motor Company (chapter 12)
10. Other companies (chapter 13)

Provided that sufficient information is available the following structure is followed for each chapter describing a company:

1. **Introduction to the company** (front matter): General information about the company that serves to give a reader unfamiliar with it an idea on factors like the size, location and business specialty.
2. **Relevant Company Structure**: Information on the organizational aspects of the company
3. **Recent History and Current State**: The development process at the company is laid out and explained.

4. **Future Directions and Challenges**: Methods and technologies that the company has shown interest in pursuing, has done research on or has expressed plans of implementing.
5. **Sources at Company**: A list of authors, their positions/departments at the company and the references to sources they authored or co-authored.

Chapter 13 contains methods that could either not be attributed to any company or to one that has no chapter of its own. For this reason it does not have any of the structure mentioned but is an exception that only contains methods.

4. AUDI

Audi is the second best selling luxury car maker in the world [64] with 1.57 million cars sold in 2013 alone. The company is based in Ingolstadt and almost exclusively owned by the Volkswagen Group (99.55%) [8]. It has employed 71k people on average in the fiscal year 2013 [7].

4.1. Relevant Company Structure

Most of the publications of Audi are made by employees from the main location of Ingolstadt. Departments seem to generally have their own development engineers and teams (e.g. Powertrain [52], Chassis Electronics [125], Electrical Engineering (EE) [140], Engineering Technology (ET) [134]) but there is a separate department for shared resources like HiL test beds for functional testing (e.g. Department EE-65 - *HiL functional testing*).

There is also a number of documents originating from Audi Electronics Venture GmbH (AEV) in Gaimersheim, which is only about five kilometers away from Ingolstadt. This company is a wholly owned subsidiary of Audi AG. It seems to have departments with multiple responsibilities like *Tools, Frameworks, Mobile Applications* [101] or *Modeling, Code Generation, Testing* [50] and work closely with Audi AG on projects [50].

4.2. Recent History and Current State

Audi uses the V-Model [68] that was developed by the German government and is required for any of its projects [134]. This model is broadly used in the automotive industry as the *de facto* standard[97, 3, 12].

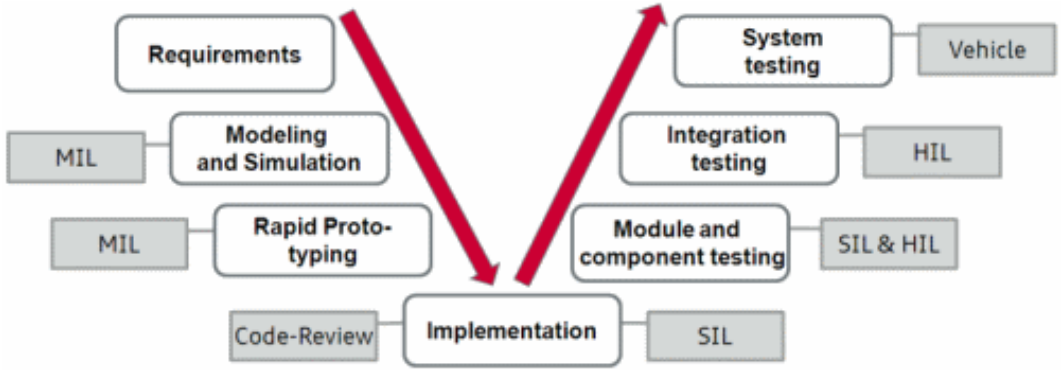

Figure 4.1.: Functional Software Development Model at AUDI [134]

According to AUDI the relevant portion of the V-Model consists of 7 steps which can be seen in figure 4.1 and which for the most part have clearly associated testing methods of Model-in-the-Loop (MiL), Software-in-the-Loop (SiL), Hardware-in-the-Loop (HiL), code-review and vehicle testing.

4.2.1. Requirements Management

The first step in the development process is the specification of requirements for the module or component. For this purpose the software IBM Rational Dynamic Object Oriented Requirements System (DOORS) is used [78]. Requirements can be entered manually but are also automatically generated from imported test cases created in subsequent steps of the V-Model [78].

4.2.2. Modeling and Simulation

This step introduces models into the process and is the first defining step required for Model Based Development (MBD) and Model Based Testing (MBT). AUDI uses the modeling tool Markov Test Logic (MaTeLo) for behavior models and MatLab/SimuLink (ML/SL) for component models [21].

Behavior Models

Behavior models are used to generate test cases for the Extended Automation Method (EXAM) framework [21]. Requirements can be imported from DOORS and linked with the test cases. When test cases are exported to EXAM this link is preserved and enables traceability throughout the test cycle [94]. MaTeLo behavior models can be customized with profiles and are combined with ML/SL models to act as a test oracle [94].

Component Models

AUDI uses ML/SL to create models of components and modules which are broken down in elementary parts called *functions*. Those functions are modeled to conform to the Model-Based Function Specification (MBFS) standard independent of the used modeling tool. A database is maintained which holds the MBFS models and the standardized (named) signals between them. The database facilitates sharing and reuse of functions and signals [52].

4.2.3. Rapid Prototyping

To allow for frontloading of tests AUDI has commissioned the creation of INCA SimuLink® Integration Package (INCA-SIP) (fig. 4.2) which is a ML/SL toolbox that connects ML/SL to ETAS GmbH INCA (INCA) and enables the validation, calibration and testing of ML/SL models in place of real Electronic Control Unit (ECU) prototypes (see MiL) [52, 46].

4.2.4. Implementation

The implementation phase has lost on importance with the shift to MBD because the actual source code is generated "almost automatically" [134]. As a result this step is now focused on techniques like code-review and back-to-back testing of software components against models [134].

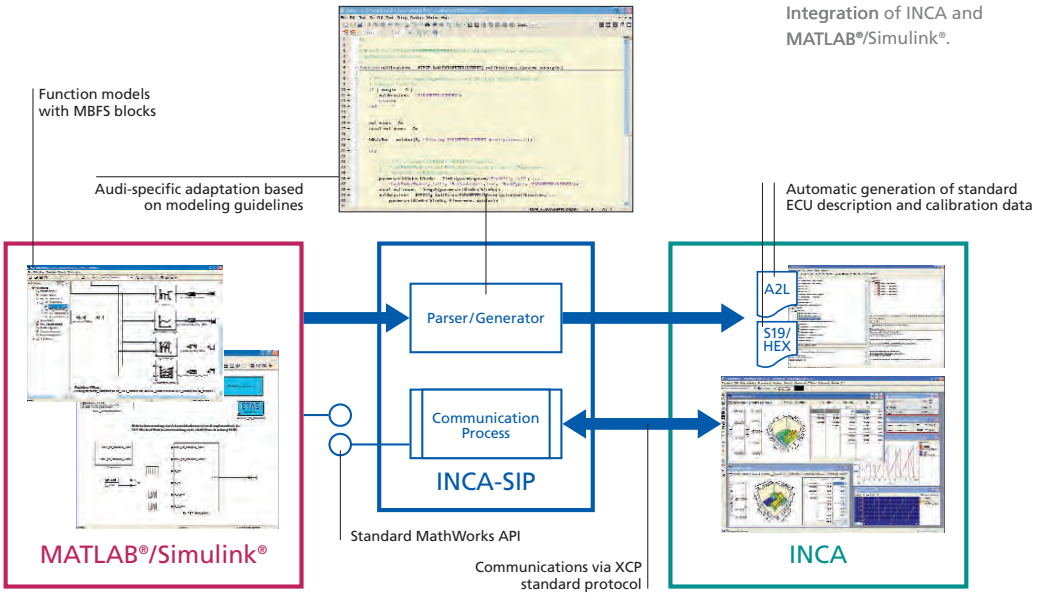

Figure 4.2.: Illustration of INCA-SIP architecture [52]

4.2.5. Module and Component Testing

Module and component tests are performed with SiL simulations. First, EXAM test cases are imported from MaTeLo, checked and corrected in the EXAM Modeler (if necessary) and optionally supplemented with additional (manual) test case definitions [94]. Next, the test suite is performed, test reports are generated and any tests that fail are investigated. Continuous traceability ensures that every failed test can be traced to the original requirement that it seeks to fulfill and/or model that it was generated from. After either the component model, the behavior model or the requirement definition is corrected, the changes are propagated to subsequent steps and the test suite is executed again. This process is repeated until all tests pass successfully [94, 140].

4.2.6. Integration Testing

In this step the software is transferred to a prototype ECU and installed into a HiL test bed. This allows the connection of the ECU to other components which can be real hardware parts or can be simulated themselves [134]. The system behavior specification and system simulation is performed with Berner & Mattner Modena (MODENA) [78]. The same testing loop as explained in 4.2.5 is performed except in this step HiL is used instead of SiL.

Types and Amount of HiL in Use

AUDI uses three different sizes of HiL test beds. There were approximately 150 Single-HiL, 20 System-HiL and 15 Network-HiL in use in 2012 [15].

4.2.7. Systems Testing

The final step in the development process is testing of a ECU prototype in an environment simulator and then in a real prototype vehicle. Audi has an

environment simulation platform called Virtual Test Drive (VTD) that can replay test drives but can also make those configurable through parameters that change the weather and other environment variables. Additionally, it was extended to include a traffic and environment simulation [100, 99].

4.2.8. Integrated Testing Framework

The Extended Automation Method (EXAM) used at AUDI is part of the Integrated Testing Framework (ITF) (fig. 4.3) concept. The ITF is designed to be modular and all components have to be implemented on the basis of a rich client platform. AUDI reported in 2009 that the ITF is headed for deployment to many different companies throughout the Volkswagen (VW) Group [78].

The actual tools being used for the general areas named in figure 4.3 at AUDI are:

- **Modeller**: MaTeLo and EXAM Modeler
- **Generator**: MaTeLo
- **Analyser**: MaTeLo
- **Requirements management system**: DOORS

4.3. Future Directions and Challenges

Audi has set sights on improved simulation precision and reach, heterogeneous simulation on a shared platform, easier deployment and better traceability of product changes in development and research into formal methods to support the current development and testing techniques.

4.3.1. Expansion of Simulation Capabilities

The vision of AUDI for the year 2020 is to expand the detail and extent of its simulation and testing equipment [15]. The dominant themes seem to be:

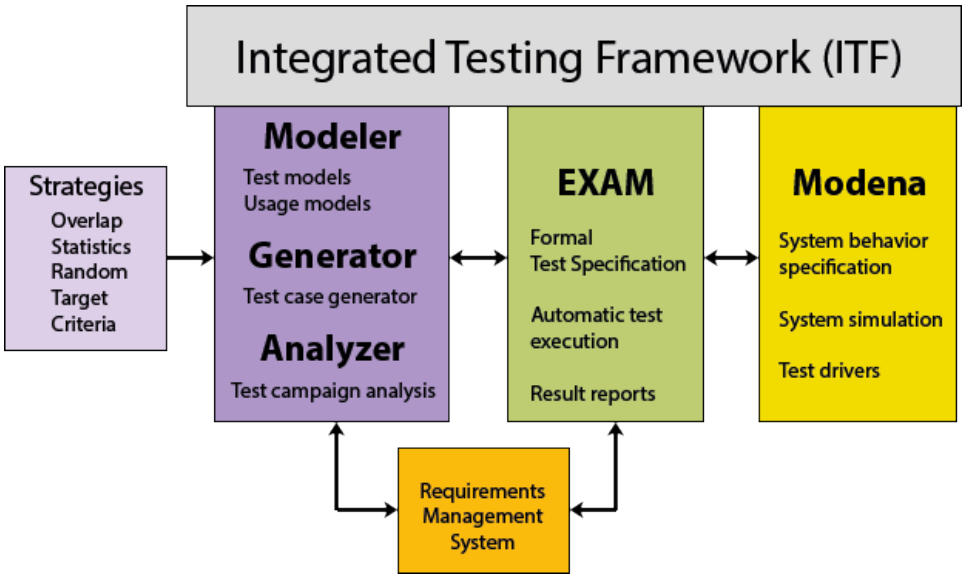

Figure 4.3.: Integrated Testing Framework [78]

- **Intensify front-loading**: Move methods to earlier steps wherever possible. (for example use SiL where currently HiL is required)
- **Increase model complexity**: More detailed models offer better representation of the real subjects.
- **Simulate more entities**: Infrastructure, Traffic and others.

4.3.2. Co-Simulation Platform

Wholly owned AUDI subsidiary Audi Electronics Venture GmbH is researching a different approach to entire-system simulation. In this approach, the integration of models is tool-agnostic. Models simulated in different tools are encapsulated and managed by the platform (figure 4.4). Validation was done on real vehicle test drive data. A complete model of a vehicle was implemented and a entire-system simulation was successfully performed [101].

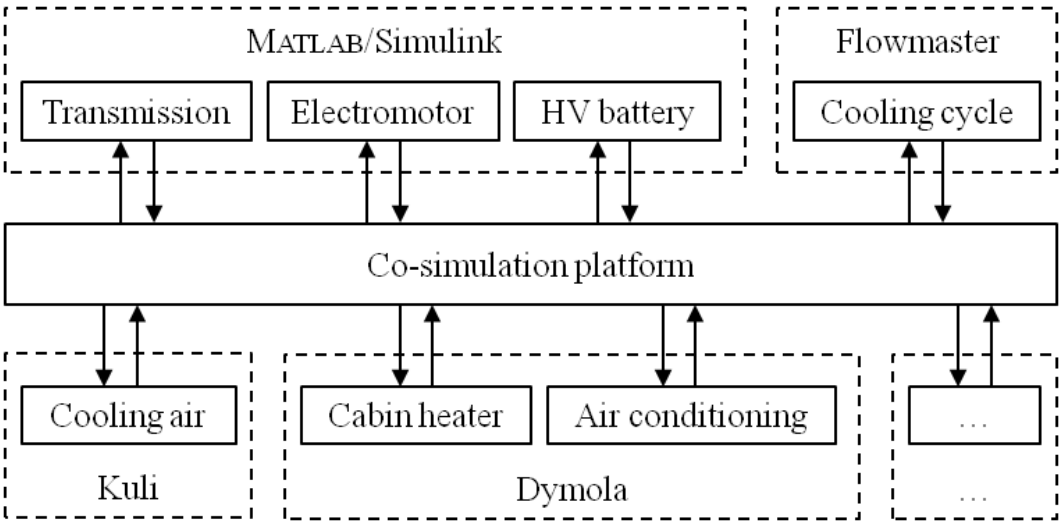

Figure 4.4.: Co-Simulation Platform

4.3.3. Runtime Configuration of Measurement

AUDI is using Universal Measurement and Calibration Protocol (XCP) in pre-series development as a means to reduce the complexity of testing by reducing the required software modification between development and deployment. By using AUTomotive Open System ARchitecture (AUTOSAR) an XCP configuration can be automatically generated. Afterwards, the ECU can be flexibly used without code modifications or dynamic code injection (memory issues) [26].

This approach can enable on-line monitoring and variable manipulation, thus allowing for fast test case set-up without overhead. It further decreases the required bandwidth and buffer space for monitoring by allowing constant data flow between the monitored components and external logging and debugging applications (transmitting buffer contents and freeing buffer) and configuring the monitoring at runtime to filter for interesting values [26].

Other possible uses of XCP are Software-implemented Fault Injection (SWIFI) (by writing faulty values to variables at a certain time) and tracing

of ECU events by attaching data aquisition (DAQ) events to them (task traces) [26].

4.3.4. Petri Nets and Logic of Action

There have been efforts at AUDI to create a "formal approach to verify predefined causal dependencies in the test cases" [134] of EXAM that is based on Logic of Action (LA) and Petri nets. The efforts have been successful in multiple case studies and show significant improvements in test case quality and cost of testing. The engineers were able to scale the method up to 10k steps without problems. There are no known limitations on the kind of test cases that can be converted in this manner [134].

4.3.5. Business Rule Ontology

For the formalization of expert knowledge Audi has contributed 1.3 million € to the ONTORULE project for the development of a Business Rule Management System (BRMS) based on domain ontologies [121]. This allows experts to create and maintain references between text documents and business rules. It also allows the annotation of documents with references to the ontology and automatic retrieval of text passages that are relevant to concepts [108, 123, 121].

4.3.6. Multiple Functions on same ECU

Audi performed a case study in which ways to operate functions of different Automotive Safety Integrity Level (ASIL) levels on the same ECU were researched (fig. 4.5). The experts were successful with ensuring ISO26262 compliance of a mixed-level ECU by using a combination of multiple approaches [125].

The first piece of the given solution is distributing the execution of tasks with high ASIL level and long execution time over multiple periods that still finish within the cycle time deadline but give smaller tasks the potential to "fit

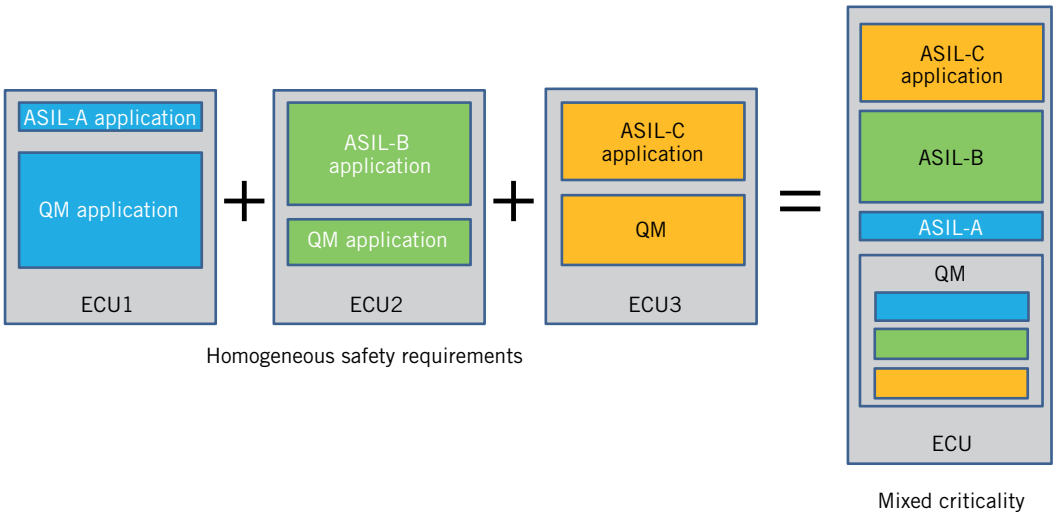

Figure 4.5.: Multiple ASIL levels on one ECU

between" the execution periods. The second one is using AUTOSAR timing protection with Criticality Aware Priority Assignment (CAPA) scheduling to alleviate the absolute need to avoid criticality inversions (e.g. lower level task interrupts higher level task) because that interruption can be managed by the timing protection mechanism [125].

4.4. Sources at AUDI AG

- DI (FH) **Johann Gabler** [52, 53] (Ingolstadt)
  works in the Development Area for Rapid Prototyping Tools in the Software and Function Development Section for Powertrain
- DI **Daniel Köhler** [52] (Ingolstadt)
  is Project Manager and Function Developer in the Software and Function Development Section for Powertrain
- **Gerhard Kiffe** [78] (Ingolstadt)
  is responsible for testing, test methodology and test automation on HiL-simulators
- **Sebastian Siegl** [78] (Gaimersheim)

is PhD student developing model-based methods for functional validation (2009)
- **Florian Netter** [101] (Gaimersheim)
  Tools, Frameworks, Mobile Applications
- **Sebastian Thiel** [134, 140] (Ingolstadt)
  Department *I/ET-84* (2011)
  Department *I/EE-65* Hardware-in-the-Loop functional testing (2010)
- **Frank Derichsweiler** [134] (Ingolstadt)
  Department *I/EE-65* Hardware-in-the-Loop functional testing
- **Dirk Zitterell** [140] (Ingolstadt)
  Department *I/EE-65* Hardware-in-the-Loop functional testing
- **Maximilian Miegler** [95, 99] (Ingolstadt)
  Department *I/EE-65* Hardware-in-the-Loop functional testing
- **Mirko Nentwig** [95, 99, 100] (Ingolstadt)
  Department *I/EE-65* Hardware-in-the-Loop functional testing
- **Peter Rosina** [108, 123, 121] (Ingolstadt)
  Department *I/ET-81* (2012)
- **Dr. Karsten Schmidt** (Gaimersheim) [125]
  Responsible for Autosar and Architecture
- **Markus Buhlmann** (Ingolstadt) [125]
  Team Leader Software and Drive Dynamics
  Chassis Electronics Development Division

5. BMW Group

Bayrische Motorenwerke AG (BMW) is the best selling luxury car maker in the world [64] with 2 million cars sold in 2013 and a workforce of 110k at the end of the year [18]. The company and is based in Munich (Germany) but its production is distributed to more than a dozen of sites [19].

5.1. Relevant Company Structure

Relevant BMW publications almost exclusively originate in Munich, which is to be expected. The declared association of employees often does not state their department within the company but rather the project and team they are assigned to or responsible for. This seems to be a common theme for BMW and might hint at the way the company works internally as well. An example of a project centered around a product or functionality is the VAP project.

One thing that seems to come up often when referring to departments is the occurrence of the word *development* (e.g. ***Development*** *Department of Driver Assistance and Active Safety* [2], *Mechatronic* ***Development*** *Department* [82], *Department of Interior* ***Development*** [82], *Department of Electronics* ***Development*** [82]). Unfortunately, this does not lead to a clear conclusion about the structural meaning of this use. One thing that is expressed clearly is that the field of simulation is contained in the IT department [91].

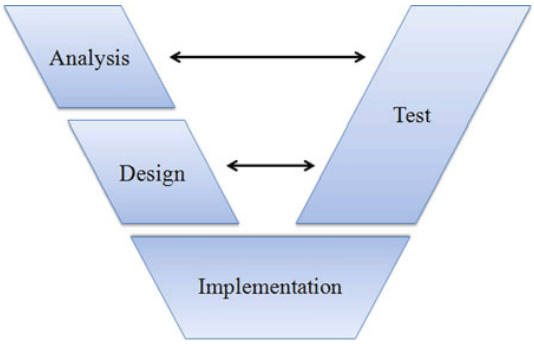

Figure 5.1.: V-Model according to BMW [97]

5.2. Recent History and Current State

The development process at BMW is based on the V-Model (see fig. 5.1) which was modified and extended to conform to the AUTOSAR paradigm of function-based development. This is in contrast to the classical process of Electronic Control Unit (ECU)-based development. [97]

This model is split into four phases called *Analysis*, *Design*, *Implementation* and *Test*.

5.2.1. Analysis Phase

The analysis (of requirements) is done with a software development process in mind. This analysis takes two documents as input. One document (called *Requirement Specification Function*) contains the functional requirements for the vehicle user (to e.g. accelerate the vehicle). The other document (called *Requirement Specification Control Unit*) contains technical requirements (e.g. sensors and actuators). These inputs are merged to produce a number of entries in a IBM Rational Dynamic Object Oriented Requirements System (DOORS) database. These entries (informally, but in great detail) describe functionality which is independent of the used hardware and can be reused for other vehicles in the future.

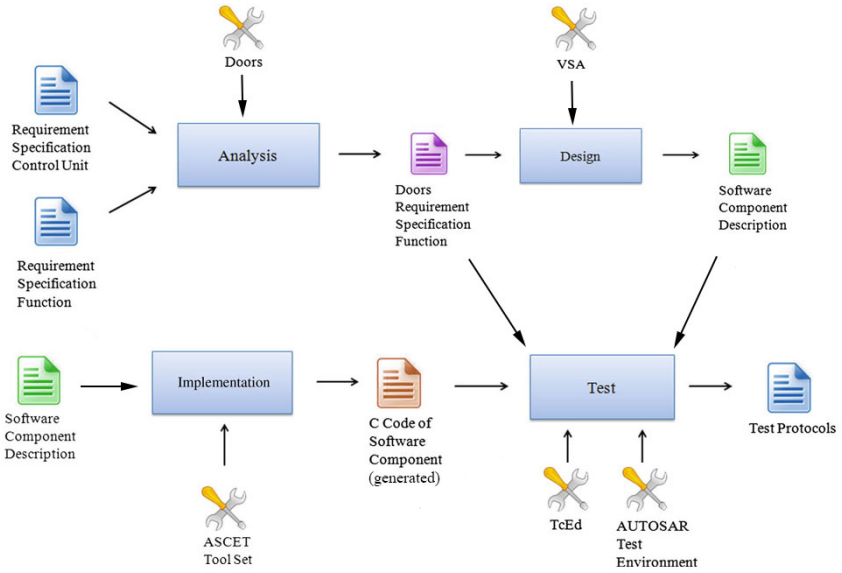

Figure 5.2.: Development Workflow at BMW [97]

The kind of functionality that the typical customer can notice (see, feel, understand, recognize) is developed in-house at BMW, because that kind of functionality greatly influences the impression and product experience of the customer and facilitates the differentiation from other brands. An example of this would be the cabin controls and displays, scheduling of convenience tasks (keeping headlights on for seconds after locking, remotely turning on auxiliary heating in the winter,...) or even the behavior of the vehicle itself. Counterexamples could be "under the hood" behaviors like the type of Real-Time Operating System that is being used or the protocol used for car-to-car communication.

At this point all the interfaces of the functionality are already defined through the signals used in the requirements specification [97].

5.2.2. Design Phase

The design phase is centered around Model Based Development (MBD) in AUTomotive Open System ARchitecture (AUTOSAR). Every body function that is relevant to the user is modeled in a single atomic AUTOSAR component. BMW uses the AUTOSAR enabled tool Vehicle Systems Architect (VSA) by Mentor Graphics for this task. This tool supports enforcing the adherence to the fixed syntax of AUTOSAR and offers plausibility checks. The output file is a standardized AUTOSAR XML (ARXML) file that is to be used as input for the Advanced Simulation and Control Engineering Tool (ASCET). The defined artifacts are either interfaces or runnables [97].

Interfaces are groups of signals. They use either asynchronous messaging similar to events or synchronous client/server communication similar to Remote Procedure Calls(RPCs). Runnables are *internal behaviours* (what that component actually does in the end) of the component that are set up in this phase (as place-holders) but definitively modeled in the implementation phase (section 5.2.3) [97].

One essential decision in this phase is the partitioning and correct assignment of behaviours to runnables. This is presumably mostly the weighing of alternatives and proper componentization [97].

5.2.3. Implementation Phase

The ARXML file generated in the design phase (section 5.2.2) is imported to the ASCET software produced by the company Engineering Tools, Application and Services (ETAS). ASCET extracts the complete AUTOSAR information and makes it available in its own model editor. The behavior that was informaly described in the analysis phase and assigned to runnables in the design phase is now formally modeled (e.g. data flow diagrams or finite state machines(FSMs)) using the AUTOSAR information from the ARXML and grouped together. The result of this are (a-/synchronous) AUTOSAR interfaces [97].

ASCET generates source code in The C Programming Language (C) out of the models, which is then compiled and linked with conventional tools

inside a AUTOSAR test environment like Berner & Mattner MESSINA (MESSINA) that provides the so called *Single Sided* Run-Time Environment (RTE) for AUTOSAR [97].

5.2.4. Test Phase

In this phase tests are performed that seek to verify the software component (SWC) against the requirement specification. This is done by using the AUTOSAR interfaces of the SWC. Test cases are created which aim to simulate inputs to the AUTOSAR interfaces and verify the correctness of the outputs produced by the SWC. The internal behaviors of the SWC are not checked directly, only trough the interface responses. This approach makes the testing process a typical black box test [97].

The tool used for creating test cases is an internal editor of BMW called Test Case Editor (TcEd). TcEd can also import the ARXML file and provide the contained information as artifacts, similar to ASCET. This makes it easier to accurately model the inputs and expected outputs of the SWC [97].

5.2.5. Virtual Validation Platform (VAP)

The Virtuelle Absicherungsplattform (VAP)[1] is a system consisting of consumer grade PC hardware with x86 architecture, the Linux kernel and the Xenomai Real-Time Operating System (RTOS). It is used in every phase of the V-Model except analysis and hardware integration testing (see fig. 5.3) [87].

Before development the VAP can be used to validate base software (BSW) stacks by using known ECUs and the VAP making the BSW the only unknown component. This is done by abstracting away the controller hardware and letting the SWC and BSW (like AUTOSAR or Geneva In-Car Infortainment (GENIVI)) run on the abstraction layer instead of real hardware (fig. 5.4) [87].

[1] virtual validation platform

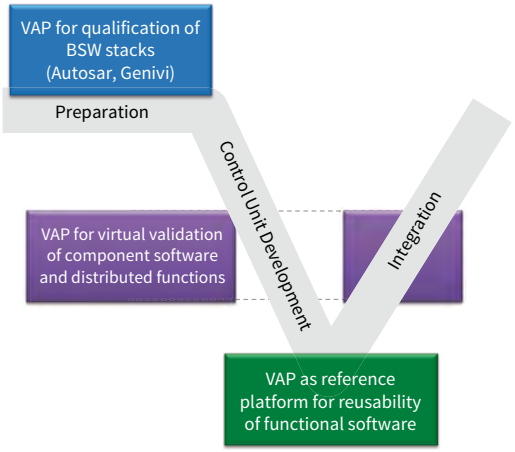

Figure 5.3.: VAP in the V-Model [97]

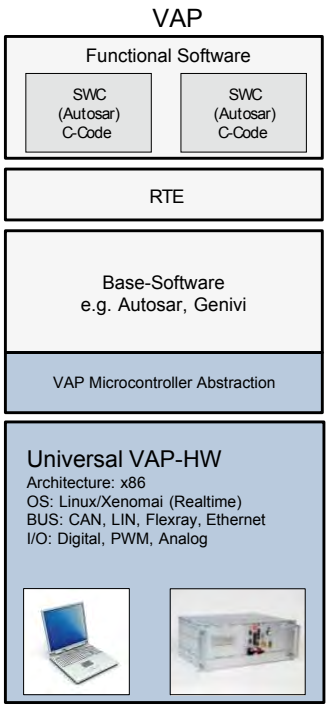

Figure 5.4.: The system under test (SuT) (grey) and the VAP-specific parts (blue) [97]

During the development the VAP can be used for virtual validation (frontloading) of the ECU and distributed functions, during implementation it can be used as a reference platform [87].

Goals of the VAP

One goal of the VAP is to decouple the development and testing of a SWC from the particular piece of hardware that it is meant to run on. This allows for frontloading of development and validation with a multitude of positive effects like cost reduction, faster completion, easier re-usability and more mature software [87].

Another goal is to validate BSW stacks on a reference platform. For this process the VAP is loaded with a reference SWC created for compliance testing together with the BSW under test. In this way the BSW can be evaluated. One application of this is to provide ways for suppliers to prove AUTOSAR compliance of their BSW.

The VAP software

The VAP software suite is based in the Provetech Tool Suite by the MBtech Group, particularly the tool Provetech:RP which was extended to include an AUTOSAR compatible RTE which in turn enables it to run AUTOSAR software components. Additionally, a complete AUTOSAR software stack called *Volcano VSTAR* by Mentor Graphics was integrated (fig. 5.5). Other required software like code generators were developed together with both companies. The configuration of tools and hardware is done in AUTOSAR compliant configuration files. [87]

The VAP tools

Development on the VAP is done with the *Volcano VSX Tool Suite* by Mentor Graphics which includes the AUTOSAR authoring tool Vehicle Systems Architect (VSA), the ECU configuration tool Vehicle Systems Builder (VSB) and code generators included in the *VSTAR* module (see fig. 5.5).

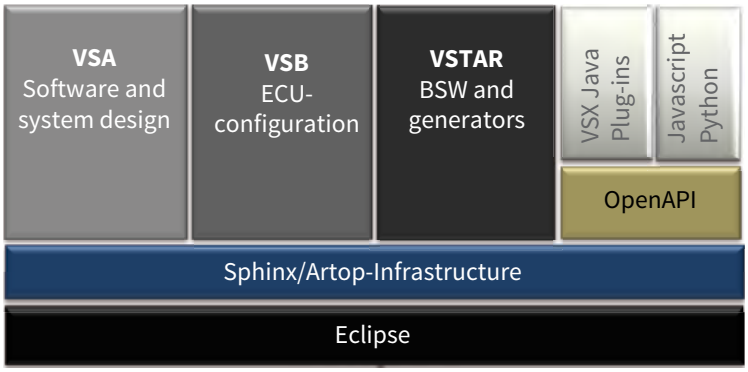

Figure 5.5.: VAP software and tools [87]

The VAP and Environment Simulation

The VAP can be connected with a complete environment simulation platform based on CarMaker (CM) which enables the simulation of a test drive complete with roads, pedestrians, traffic, maps, GPS and so on [88].

5.3. Future Directions and Challenges

BMW sees the future in function-oriented instead of ECU-oriented development. One reason for this is the steady increase of different applications for software in the vehicle, all of which would need their own ECU. The current situation is that about 200 ECUs would be needed, which is seen as an unmanageable amount. In the function-oriented perspective, multiple functions can be combined in the same ECU [97].

5.3.1. Progress of the VAP

It seems like the VAP is here to stay and that it will continue to improve and gain in features and stability. One thing that is definitely planned is the consolidation of the VAP and the CM platform. This is seen as logical next

step, because both platforms run on the same hardware and software [88, last paragraph].

5.3.2. Diagnosis Function Improvement

BMW is analysing data sent back from incidents that come in for repair to improve the diagnostic functions used in-field by their cars, which make up about 40% of total software in the vehicle. This data is about 5 gigabytes per day and requires techniques commonly used for for big data and statistical processing to process. The goal for the future is that this diagnosis should be integrated with development efforts and to create a closed-loop fully automatic diagnosis feedback process [79].

5.4. Sources at BMW

- **Robert Siwy** [97] (Munich)
  Software Development Engineer (2013)
  Responsible for creation of ASCET models of chassis functions and their integration in ECUs. Responsible for concept and realization of model library. (2006 [124]).
- Dr. Ing. **Marcus Martinus** [88, 87] (Munich)
  Teamleader Software Integration and Virtual Verification (2013)
  Projectleader Virtuelle Absicherungsplattform (VAP) (2012)
- DI **Markus Deicke** [39] (Munich)
  PhD student working on Virtuelle Absicherungsplattform (VAP)
- **Jens Kohl** [79] (Munich)
  Validation & Verification System Functions
- **Agnes Kotucz** [79] (Munich)
  Diagnosis After Sales
- **Johann Prenninger** [79] (Munich)
  Diagnosis After Sales

6. Daimler AG

Daimler is the third best selling luxury car maker in the world [64] with 1.56 million units sold in 2013 but also a major truck maker in the global market with 484k units sold [37].

Daimler has a workforce of 274k employees in total and 97k in the luxury car division. The company headquarters are based in Stuttgart (Germany) [37].

6.1. Relevant Company Structure

Most of Daimler publications seem to originate from Sindelfingen (company headquarters) and Böblingen, which turns out to be only a few kilometers away from Sindelfingen and house the main location for body construction. Both locations seem to house a part of the division Group Research and Advanced Engineering (GRAE) to which most relevant documents are attributed. GRAE has multiple departments (e.g. *E/E-Architectures & Standards*), teams (e.g. *E/E-System Design and Applications*) and projects (e.g. *model-based AUTOSAR-Toolchain*) [93, 92].

6.2. Recent History and Current State

Daimler uses a customized version of the V-Model for Model Based Development (MBD) of software for safety-critical functions (fig. 6.1) [92, 93]. For this purpose, Daimler divides the V-Model into seven phases of 4 phase types (bold) [92]:

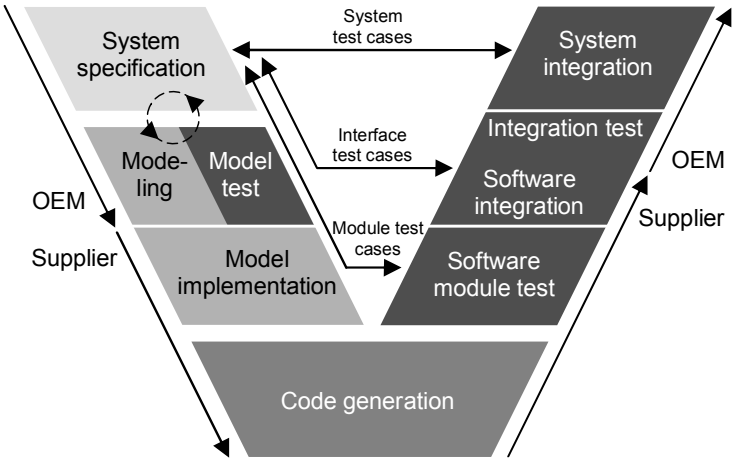

Figure 6.1.: The V-Model used at the Daimler AG [92]

1. System **Specification**
2. **Model**ing and Model **Test**
3. **Model** Implementation
4. **Code** Generation
5. Software Module **Test**
6. Software Integration and **Test**
7. System Integration **Test**

6.2.1. System Specification

The specification of the system is done in a textual representation at Daimler. This textual representation is the only reference artifact for the rest of the development life cycle (Single Source Principle). This has the advantage that all requirements are available before any part of the software component (SWC) is developed [92]. The tool used to create this textual representation is IBM Rational Dynamic Object Oriented Requirements System (DOORS) [110].

6.2.2. Modelling and Model Test

All models are created with the AUTomotive Open System ARchitecture (AUTOSAR) standard in mind. Daimler uses MatLab/SimuLink (ML/SL) for model development. The created models are tested based on a test plan. Test cases are created which consist of the input and the expected output of the system under test (SuT). If the expected output is not produced, the test case has failed [92]. The software used for test creation and management includes Berner & Mattner TESTUS (TESTUS) [126] and QTronic TestWeaver (TestWeaver) [54].

Daimler has developed a number of dynamic test methods for the functional validation of models. These include the Classification Tree Method (CTM/ES), Time Partition Testing (TPT), Model Based Testing (MBT), structural coverage criteria and back-to-back testing [92].

6.2.3. Supplier Handover

After testing is completed models are transferred to the supplier together with the system specification, the test plan and the implemented model tests. The supplier has the responsibility to perform *Model Implementation*, *Code Generation*, *Software Module Tests* and *Software Integration* together with the base software (BSW). This responsibility ends only after the Electronic Control Unit (ECU) is finished and Daimler can take over again to perform component, system and integration tests [92, 93].

6.2.4. Component, System and Integration Tests

After receiving the finished component, the Original Equipment Manufacturer (OEM) is in a difficult situation. The ECU is only available in hardware with pre-compiled software binaries. Software sources or models are not available and Software-in-the-Loop (SiL) tests can not be performed. The ECU software can not be tested and validated without hardware dependencies and real-time constraints in this situation. Reverse engineering

efforts would need to be employed which are error prone, unreliable and expensive [90, 54].

TriCore Simulation

Daimler has solved the problem in part by simulating the ECU hardware, in particular the TriCore processor. The company QTronic has extended their software QTronic Silver (Silver) to execute binaries that were compiled for the TriCore processor on a simulated CPU and enable the binaries to be run without the target hardware being available. An additional advantage of this approach is the possibility of (much) faster than real-time simulation of test runs.

Silver not only performs the simulation of the processor but simulates the communication of the ECU with other components and software as well (fig. 6.2) [90]. This allows for virtual calibration and measurement of the simulated ECU with Universal Measurement and Calibration Protocol (XCP) over Car Area Network (CAN). It does have it's limitations, however [89]:

- instruction accurate, but not cycle accurate: cannot predict exact execution times
- based on TriCore and PowerPC specification: 'silicon bugs' are not simulated
- on chip devices not modeled: cannot run basic software, such as device driver

6.2.5. Software-in-the-Loop Testing

The first cycle of testing is executed on virtual ECUs being simulated in Silver. The test process is managed by TestWeaver which offers the option to either replay sequences of input values from pre-recorded vehicle test drives or generate them on its own given the available input variables. The second mode also offers state space exploration (e.g. tries to include all possible gear shifts) and failure state induction (e.g. tries to drive the system into states that violate system requirements) [54].

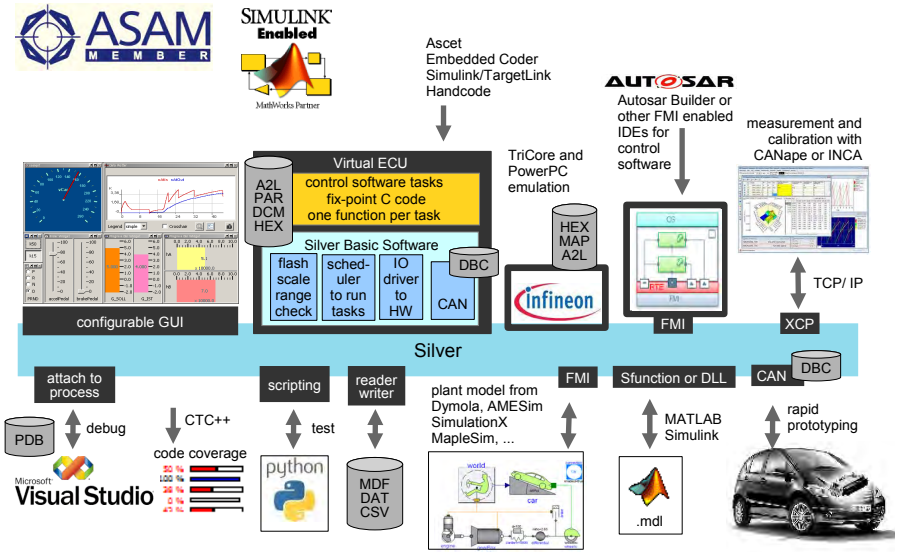

Figure 6.2.: QTronic Silver Architecture [75]

The test suite may execute on a single instance or on multiple instances of virtual ECUs simultaneously (e.g. variants) and several times faster than in real-time (fig. 6.3). While tests are running, data is being recorded in the software tool Vector Informatik CANape (CANape). This data allows for statistical analysis of conditions that are marginally defective and may signal a future problem [54].

6.2.6. Hardware-in-the-Loop Testing

After the component has passed SiL testing it moves on to Hardware-in-the-Loop (HiL) testing. Daimler uses HiL test-beds by dSPACE GmbH and the automation software Provetech-TA by MBTech [126].

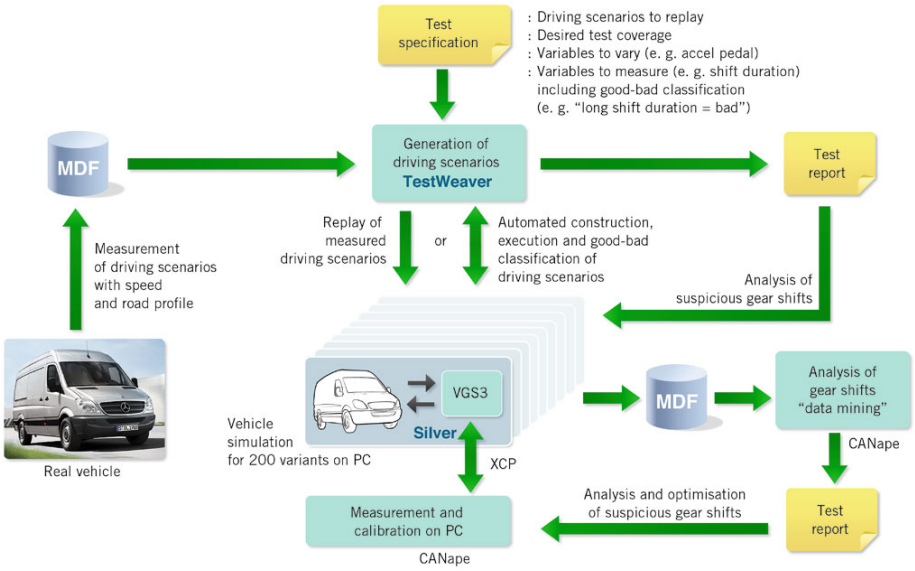

Figure 6.3.: QTronic TestWeaver and Silver Testing Procedure [54]

6.3. Future Directions and Challenges

The future for Daimler seems to contain a move to improved virtualization through their Virtual Integration and Testing (VIT) platform, which is being developed and improved at the moment. Further improvements in the analysis and management of requirements are on the way, which primarily serve the relief of an expert bottleneck and increased efficiency. Finally, software quality is the continued target of efforts which might be helped with the use of constraint solving techniques.

6.3.1. Virtual Integration and Testing Platform

Daimler has executed a pilot project together with dSPACE GmbH which seeks to create virtual ECUs that behave identical to hardware ECUs. They communicate over the same protocols as hardware ECUs and can be tested and deployed without any modification [67].

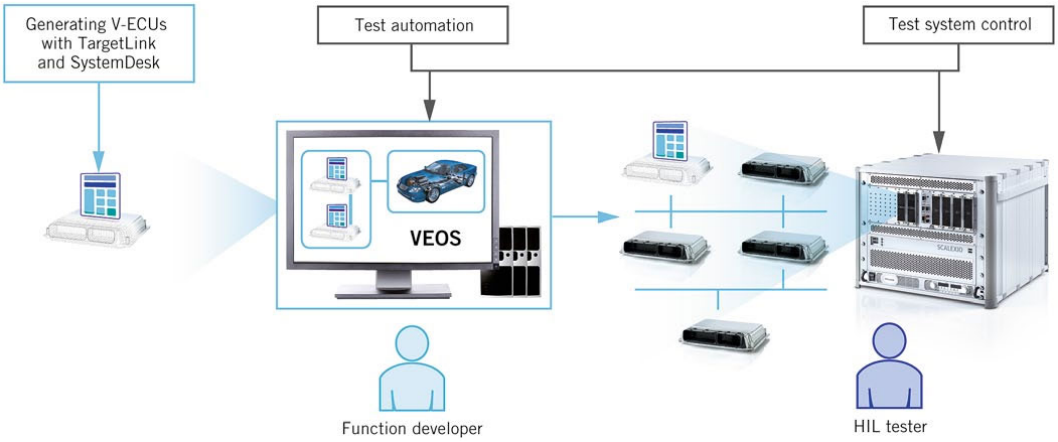

Figure 6.4.: Virtual Integration and Testing Platform [66]

This project is called Virtual Integration and Testing (VIT) and its main component is the VEOS simulation platform together with TargetLink and SystemDesk for model generation (fig. 6.4). All of these are produced by dSPACE GmbH [67] and only support Windows OS [43].

VIT seems to compete with Silver as they have significant functionality overlap. It is possible that Daimler seeks to replace Silver. Reasons for the move might be the shortcomings of Silver listed in section 6.2.4 [67].

6.3.2. Review with Categorized Requirements (ReCaRe)

The reviewers at Daimler often have to manage huge document collections when reviewing specifications. Combined with referenced documents the specification material can add up to over 3000 pages [110]. This amount of information is not well suited for manual human processing which is why it is usually assisted by software. This is however not always trivial. The mentioned documents are authored in natural language, which computers are not well suited to process.

The biggest problem this vast document collection poses, is the loss of overview for the reviewer. This leads to holes in specification and coverage,

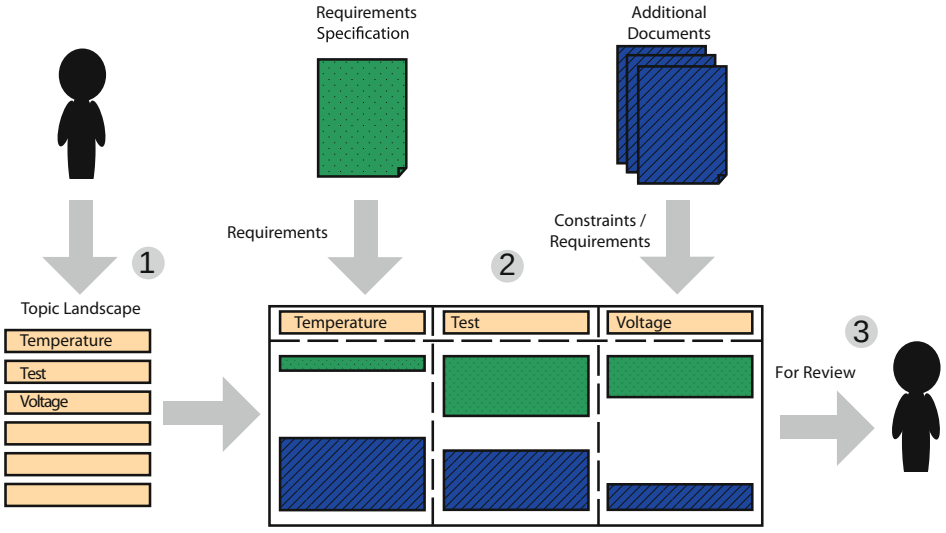

Figure 6.5.: **Re**view with **Ca**tegorized **Re**quirements (ReCaRe) Illustration [110]

overlapping or contradicting specifications and wasted time and resources both for the reviewer (slow progress) and the developer (additional iterations).

Daimler has conducted an evaluation of Natural Language Processing (NLP) methods to help ease this problem in the future. They have developed a method and associated software tool called **Re**view with **Ca**tegorized **Re**quirements (ReCaRe) which aids the reviewer by imposing a *topic landscape* on the body of documents. The tool connects to DOORS and retrieves the natural language requirements. The requirements and specification documents are analyzed and parts of both belonging to the same topic (e.g. *temperature*) are combined into new documents (fig. 6.5). This allows the reviewer to keep the overview by only looking at one aspect of the specification at a time [110].

This method has shown to be feasible, mostly being hindered by limited training data . This work is to continue and one of the main research directions will be on how to get enough training data automatically with minimal human intervention and time investment [110].

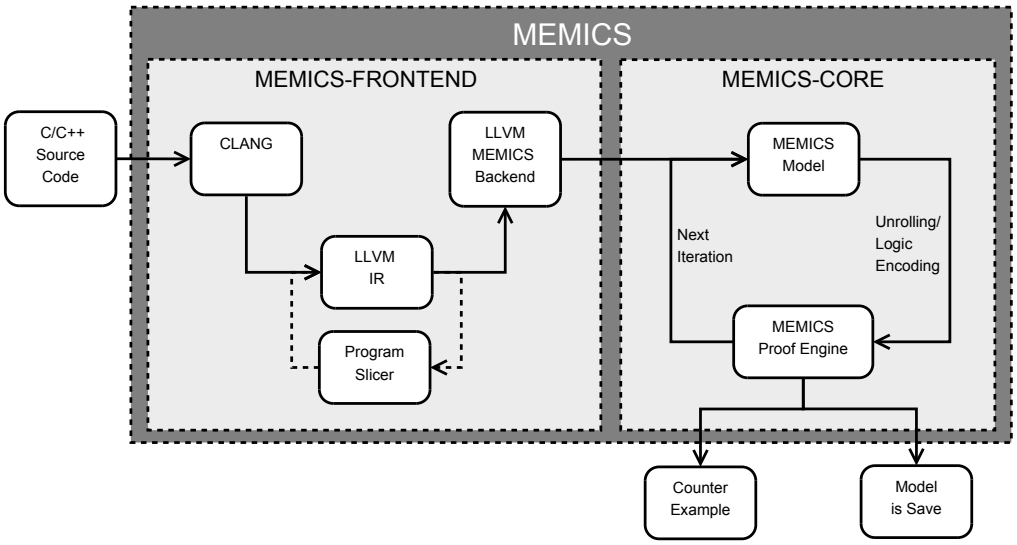

Figure 6.6.: Memory Interval Constraint Solving (MEMICS) Architecture [104]

6.3.3. Memory Interval Constraint Solving (MEMICS)

In an effort to reduce race conditions and other errors - particularly resulting from the use of modern multi-core systems - Daimler has developed the static software checking tool Memory Interval Constraint Solving (MEMICS) together with the Kiel University. MEMICS uses the Low Level Virtual Machine (LLVM) to generate a suitable model for its constraint solver (MEMICS Intermediate Representation (IR)) which can then be unrolled into Static Single Assignment (SSA) logic formulae which are solved in the proof engine (fig. 6.6) [105, 104].

The tool has shown promise in evaluations while suffering from the same problems that similar solutions have. The biggest problem is the high complexity and the resulting limitation on the size of programs that can be checked with reasonable computational resources [105, 104].

6.4. Sources at Daimler AG

- **Artur Honisch** (Sindelfingen) [66, 67]
  in charge of the VIT Platform Project (2013)
- **Matthias Simons** (Stuttgart) [90]
  Development Engineer, HPC: 050/G007 (2012)
- **Uwe Spieth** (Sindelfingen) [92]
  Group Research & Advanced Engineering
- **Dipl.Inform. Alexander Michailidis** (Böblingen) [93, 92]
  PhD student in the department *E/E-Architectures & Standards* in the division Group Research & Advanced Engineering (2010)
- **Dr.Ing. Thomas Ringler** (Böblingen) [93, 92]
  Project leader for the model-based AUTOSAR-Toolchain in the division Group Research & Advanced Engineering (2010)
- **Dr.Ing. Bernd Hedenetz** (Böblingen) [93, 92]
  Head of the team *E/E-System Design and Applications* in the division Group Research & Advanced Engineering (2010)
- **Johannes Traub** [105, 104]
  E/E-Technologies and Software-Technologies (2013)
- **Daniel Ott** (Ulm) [110]
  Group Research & Advanced Engineering (2013)
- **DI Carsten Rustige** (Stuttgart-Untertürkheim) [126, 16]
  Sub-Project leader in Software Development (2011)
  Responsible for the software development of the Atego Hybrid and the supervision of TESTUS in the development department of robot-driven gearboxes in the commercial vehicles sector (until 2011)
- **Stefan Gloss** (Stuttgart) [54]
  Engineer (2013)

7. ZF Friedrichshafen AG

ZF Friedrichshafen AG (ZF) is a German supplier of Powertrain, Chassis, Commercial Vehicle and Industrial technology. They also supply steering systems in a joint venture with Robert Bosch GmbH. ZF generated 17.4 billion in sales in 2012 with 75k employees in 121 locations [51].

In 2012 ZF was 9th in the ranking of automotive part suppliers in the world and 5th in Europe [34].

7.1. Recent History and Current State

ZF uses a variation of the widely used V-Model for software development (fig. 7.1). The model is split horizontally to better visualize the fact that the top four phases are performed by a different person than the bottom five phases. This results from recent efforts by ZF to standardize and combine their testing procedures and tools. The goal is to allow easier cooperation with changing Original Equipment Manufacturer (OEM) partners and also enable the externalization of testing into a new location that was opened in the Czech Republic [12].

7.1.1. Requirements Engineering

This part includes the phases called *System Requirements* and *Software Requirements* in the V-Model (fig. 7.1) and is performed by an independent tester [12]. ZF uses IBM Rational Dynamic Object Oriented Requirements System (DOORS) for requirements management and ensures traceability of

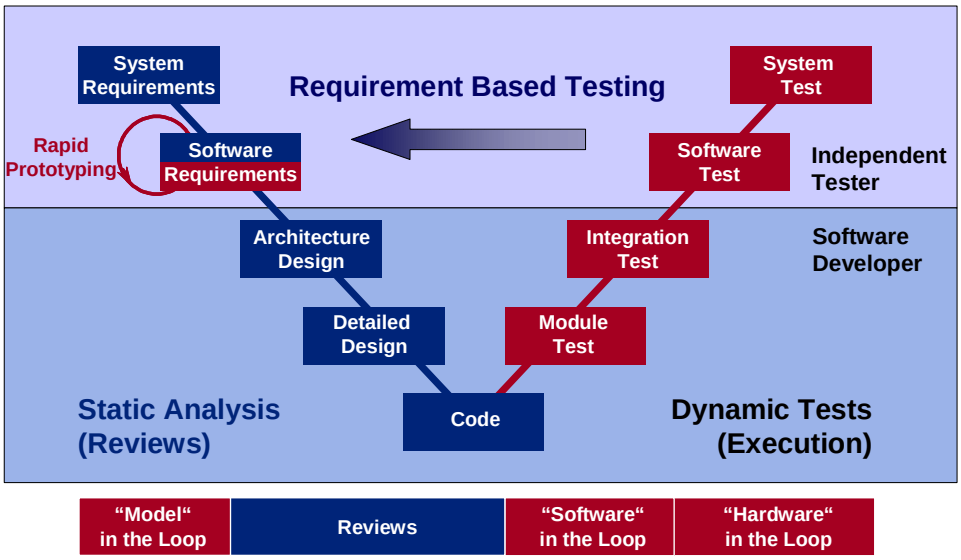

Figure 7.1.: The V-Model used at ZF [12]

Natural Language Processing (NLP) representations to semi-formal representations and source code by using an Unique ID (UID) called *Requirement-ID* which is saved in a DOORS text-field [102].

Some requirements have to be checked constantly. To cover these cases a special configuration file for QTronic TestWeaver (TestWeaver) is created. This is a file written in The C++ Programming Language (C++) that contains so-called *Watcher Instruments*, which are similar to invariants and are checked and logged every millisecond. After requirements have been completed they are exported as Extensible Markup Language (XML) and then imported into the Eclipse. The transformation language *XPAND* of the Eclipse Modelling Framework (EMF) is used to automatically create source code comment fields with the DOORS requirement text [102] and the Requirement-ID.

The actual implementation of the test cases is contained in a protected region of a special source code file. Changes in requirements propagate to the comment fields and are reviewed by using the *diff* tool of the source control software [102].

Test Scripts

Alternatively, more conventional test scripts can be used in a multitude of formats and programming languages. Beside the custom script language provided by the in-house developed tool ZF Friedrichshafen SoftCar (SoftCar), there is support in SoftCar for test case libraries in C/C++, Python or Visual Basic. This allows for direct import of libraries generated with software that is in use at partners of ZF. One example of this would be the Extended Automation Method (EXAM) tool used by Audi [102].

7.1.2. Software Design and Static Analysis

This part consists of three phases in the V-Model called *Architecture Design*, *Detailed Design* and simply *Code*. The executing person is the *Software Developer* and (presumably) one or more *Reviewers*. As the names of the phases suggest, this part is where the actual software component is planned, designed and programmed. There are no indicators that any kind of source code generation from models is performed at all. Since debugging is performed with the Visual Studio Debugger it appears likely that Microsoft Visual Studio is used for coding [102, 12].

Compilation

At the end of this part the completed Electronic Control Unit (ECU) software is compiled either into a firmware binary (*.hex file) for deployment into a physical ECU or into a Microsoft Windows binary (*.exe file) for testing as virtual ECU in the SoftCar co-simulation platform [102].

7.1.3. Module and Integration Testing

The testing phase has received some improvements and refinements with the inclusion of TestWeaver as a regular part of the testing infrastructure in cooperation with the very mature SoftCar suite.

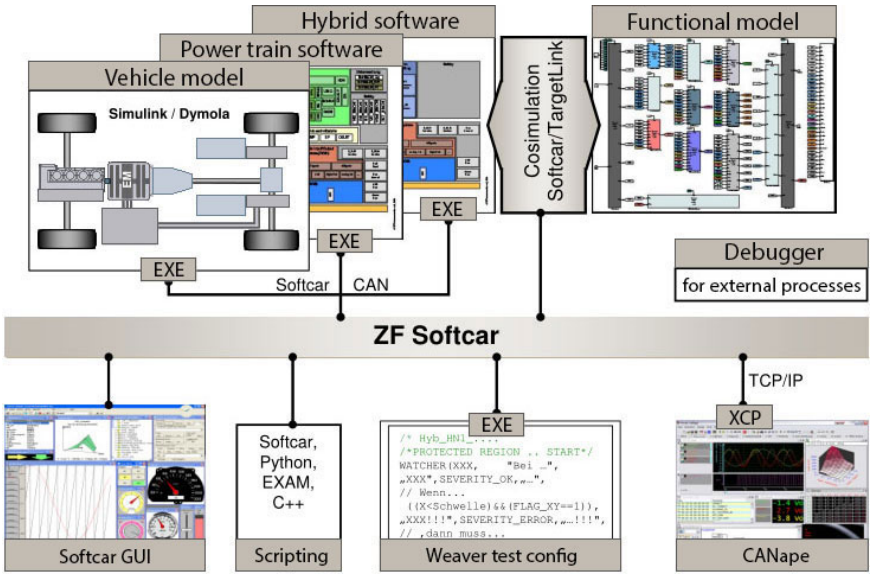

Figure 7.2.: SoftCar Architecture [102]

SoftCar

After the ECU software code is compiled into a *exe* file, it can be tested. ZF has their own simulation platform called SoftCar which they use for co-simulation of environment and ECU models, for gathering simulation data, execution of test cases, monitoring and debugging (fig. 7.2) [102].

The simulation with SoftCar can be interrupted at any time and variables of every model in the simulation can be inspected and modified with the Visual Studio Debugger. All the simulated ECU and models are connected over a simulated Car Area Network (CAN) [102].

SoftCar offers its own facilities for simulation control, monitoring and execution, but has interfaces to delegate that function to other software like ETAS GmbH INCA (INCA) or Vector Informatik CANape (CANape) using Universal Measurement and Calibration Protocol (XCP) on TCP/IP over Ethernet [102].

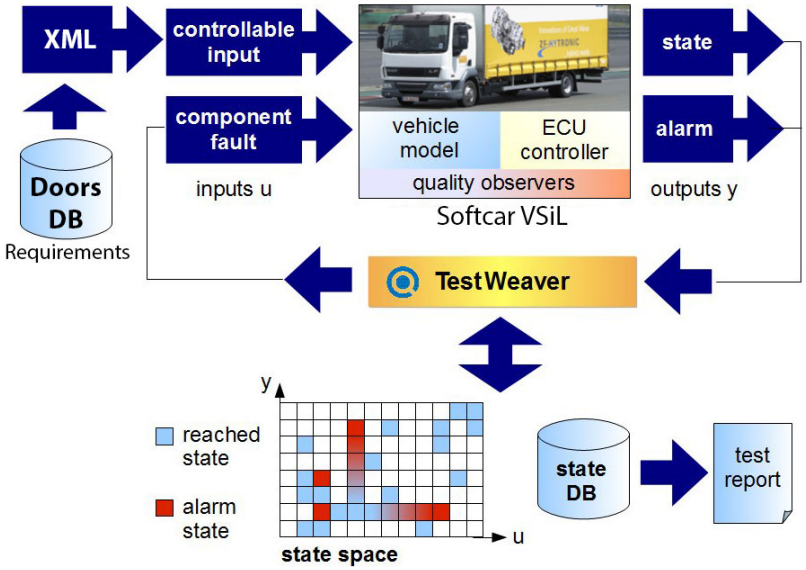

Figure 7.3.: SoftCar and TestWeaver [102, 133]

TestWeaver

One important aspect that is missing in SoftCar is the possibility to define states and ranges that should be checked continuously, i.e. not only at the end of test case execution (e.g. "battery should never be empty"). Another is the lack of automatic state exploration and the function to automatically try to induce states violating system requirements. This are some of the reasons that ZF is extending SoftCars functionality by using TestWeaver [102].

TestWeaver not only supplies the missing functions to SoftCar, but also adds more. It creates a protocol of all used paths and outcomes and allows for a replay of signal combinations which led to undesirable outcomes [102].

7.2. Future Directions and Challenges

ZF seems to have plans to decouple the testing process from the development process as much as possible. Presumably this is meant to enable the option

of offering each one up as a service to customers. ZF has made several steps to prepare this decoupling, starting with the encapsulation of their own testing infrastructure and connecting it to the outside by strictly defined interfaces. This makes it possible to operate a testing process independent from the infrastructure used by customers [12].

7.2.1. Statistical Analysis

ZF reported in 2010 that they are using statistical sampling theory for evaluation of software reliability based on operational experience (extensive road testing). When an evaluation is performed possible state transitions are identified and their occurrence is modeled with probabilities from collected data. Then, the value of relevant environment parameters at the time of that state transition (e.g. what was the speed when gear switched from a to b) is modeled with distributions. At that point, statistically dependent data needs to be filtered [131].

For this purpose a tool was developed that allows the statistical analysis of data on several correlation metrics. Additionally, it supports heuristic analysis to extract data that is negligibly correlated (tolerable bounds set by user) with a genetic algorithm. The tool is an internal development and runs on Microsoft Windows OS [131].

ZF uses component reliability estimates to in turn estimate the reliability of the system as a whole by modeling distinct components as statistically independent, exponentially distributed random variables and combining them into a single variable that is hypo-exponentially distributed. From this a conservative lower bound of reliability can be estimated for any confidence level [131].

7.3. Sources at ZF Friedrichshafen AG

The documents published by ZF employees usually do not contain position or department information. This is only the case to some degree in two of the documents.

- **DI Alexander Banerjee** [10]
  Advanced Development, division Commercial Vehicle Technology
- **Ing. Dr. Maik Würthner** [10]
  works in System Testing and is responsible for the development of Prevision GPS, division Commercial Vehicle Technology
- **DI Cosmin Tudosie** [10]
  works in Function Development Gearbox for Heavy Vehicles, division Commercial Vehicle Technology
- **Frank Bitzer** [131]
  LPE2-FB/Functions Basic Development
- **DI (FH) Lothar Beller** [12]
- **Ing. Pavel Turjanica, Ph.D.** [12]
- **Ing. Dr. Martin Neumann** [102, 103]
  Head of Lean Development ZF Production System [86]
- **DI Mario Nass** [102, 103]
- **DI Carsten Paulus** [102, 103]

8. Siemens

Siemens is a multinational company with 362k employees, 220k of those in Europe. It is based in Germany and produces a wide range of products including those supplying the automotive industry [127].

8.1. Relevant Company Structure

Despite the fractured nature of Siemens with publications originating from Germany, USA, Turkey, India and more, there is a clear concentration of relevant documents from Siemens Corporate Research & Technology in Princeton (New Jersey, USA).

8.2. Recent History and Current State

Because of the wide range of products in different domains Siemens is extremely difficult to condense into a congruent picture. Assertions that might hold for one division of Siemens might be the total opposite of what one would assert about another division.

8.2.1. Organizational Aspects of Software Reliability

The development at Siemens involves organizational factors like code walk-throughs, inspections and pair programming. It also uses early design phases called rough upfront design (RUD) and *walking skeleton* [24, 23].

The RUD is the earliest design phase and involves fundamental decisions about architecture, modularization and key qualities. It is designed to give a first impression of the viability and scope of a project and its dependencies. The *walking skeleton* is a kind of minimum viable product (MVP). It is design to be executable, perform base functions and function as a reference for feasibility analysis [24, 23].

8.2.2. Test Design Studio and Enterprise Architect

Siemens uses the self-developed software Test Design Studio (TeDeSo) for model-driven development and testing. It is designed to enable and support a work-flow-driven and service-oriented process. The focus is on extensibility of modeling and generation through the use of services (fig. 8.1) [128, 129].

TeDeSo uses an extended Unified Modelling Language (UML) notation that keeps compatibility with baseline UML and offers a wide range of import and export options with other diagram formats. The built-in model editor offers many model types and performs syntax and semantic checks. Reports can be used to evaluate aspects of models for verification and indexing [128, 129].

One main function of TeDeSo is the automatic generation of test cases and subsequently executable test code. The produced code is annotated and linked to the original test cases. This allows for regression testing and test report generation with descriptions of manual steps to perform to reach critical state [128, 129].

The software that Siemens imports models from is Sparx Systems Enterprise Architect (EA). They have built an importer for EA models in TeDeSo and extended EA to include a tool bar for direct creation of TeDeSo annotations [128, 129].

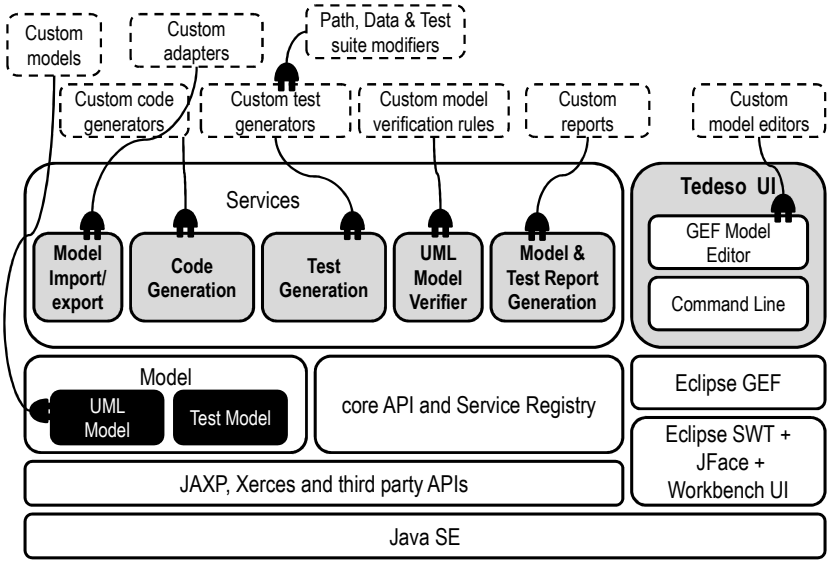

Figure 8.1.: TeDeSo Architecture [129]

8.2.3. Automated Test Infrastructure

Siemens uses a testing framework called Automatic Testing Infrastructure (ATI) for automatic test environment set-up, configuration and provision of software interfaces for hardware devices. Together with TeDeSo this constitutes a full test case creation, execution and analysis suite called Integrated Model Based Testing (iMBT) (fig. 8.2) [129, 128].

ATI is not a test engine itself. It is used together with a test engine. One such test engine that is used at Siemens is National Instruments TestStand (TestStand) [129, 128].

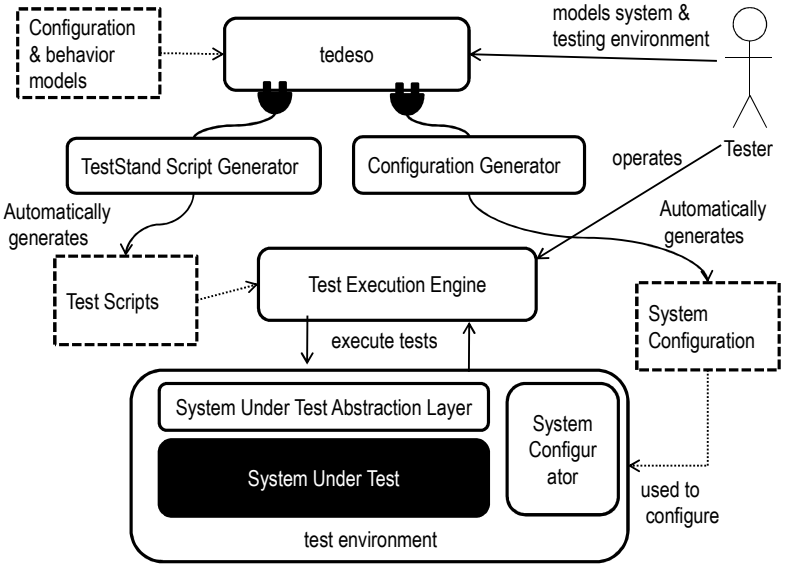

Figure 8.2.: Integrated Model Based Testing (iMBT) [128, 129]

8.3. Future Directions and Challenges

8.3.1. Unified Requirements Modeling Language

The frustration with certain weaknesses of UML and Systems Modelling Language (SysML) in regard to modeling of requirements (e.g. inability to connect product lines, features) led to the development of a new visual modeling language by Siemens and Technische Universität München (TUM) called Unified Requirements Modeling Language (URML) [14, 13].

8.3.2. SystemC with Hybrid Channels

Employees of Siemens AŞ in Istanbul (Turkey) have developed an extension for The SystemC Event Library (SystemC) that enables integration of virtual models and physical devices. The proposed solution deals with real-time

constraints and syncronization issues. It is used in industrial automation systems [48].

8.3.3. Event-B, ProB and Rodin Platform

The Industrial deployment of advanced system engineering methods for high productivity and dependability (DEPLOY) project has performed case studies in intensive cooperation with Siemens. This project is evaluating and developing Event-B, ProB and the Rigorous Open Development Environment for Complex Systems (RODIN) Platform, all of which are tools for formal verification [25].

There has been at least one industrial case study from Siemens in which ProB was used to validate large datasets in the railway domain.In another case study ProB was used to check for deadlock situations in a software [25].

8.3.4. Test Cases from Models

Siemens has researched methods to extract test cases from models and performed pilot cases on data similar to production use. They have announced plans to use the method on a large, mission-critical system in 2011. There is no indication what happened after that and if the method was deployed into production or developed further [9].

8.4. Sources at Siemens AG

- **Frank Buschmann** (Munich) [24, 23]
  Senior Principal Engineer / Principal Software Architect, Corp. Technology, Software & Engineering Division, Architecture Department
- **Roberto S. Silva Filho (Princeton, NJ)** [128]
  Corp. Research & Technology, Software Development Technologies
- **Christof J. Budnik (Princeton, NJ)** [128]
  Corp. Research & Technology, Software Development Technologies

- **Roberto S. Silva Filho (Princeton, NJ)** [129]
  Corp. Technology, Software Architecture Design
- **William M. Hasling (Princeton, NJ)** [129]
  Corp. Technology, Software Architecture Design
- **Christof J. Budnik (Princeton, NJ)** [129]
  Corp. Technology, Software Architecture Design
- **Monica McKenna (Princeton, NJ)** [129]
  Corp. Technology, Software Architecture Design
- **Abian Blome** (Germany) [17]
- **Martín Ochoa** (Germany) [17]
- **Brian Berenbach** [14, 13]
  Corp. Research & Technology
- **Alberto Avritzer** (Princeton, NJ, USA) [9]
  Corp. Research

9. Bosch GmbH

Bosch GmbH is a German company with 281k employees and 46 billion in revenues in 2013. These figures do not include the 50% joint ventures with Siemens (home appliances) and ZF Friedrichshafen AG (ZF) (automotive parts) [20]. Bosch headquarters are in Gerlingen, near Stuttgart (Germany). In 2012 Bosch was the biggest supplier of automotive parts in the world [34].

9.1. Relevant Company Structure

Similar to ZF, usually Bosch does not include employment position information with their publications. At least, there is department affiliation available for about half of the authors and locations for most of them. Of the different locations that can be distinguished, all except one are found within 15km from the company headquarters in Gerlingen (Gerlingen [122], Schwieberdingen [57, 59, 58, 49], Stuttgart [58, 59, 57, 117, 115, 116, 114]). The exception is a contribution from the Corporation Research and Technology Center in Pittsburgh (PA, USA) [1].

9.2. Recent History and Current State

Bosch uses its own interpretation of the widely accepted and used V-Model (fig. 9.1). The model is split in seven phases, which is common. The phases are called as follows:

1. System Requirements

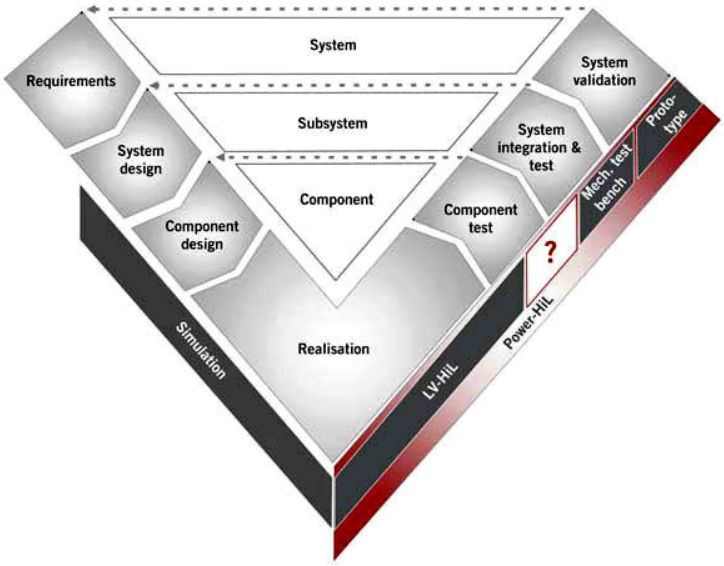

Figure 9.1.: V-Model at the Robert Bosch GmbH [96]

2. System Design
3. Component Design
4. Component Realization
5. Component Test
6. Sub- System Integration and Test
7. System Validation

Note: Gray words added for clarification in accordance to graphic (fig. 9.1).

9.3. Future Directions and Challenges

The general future direction of Bosch is characterized by a move to the most recent version of AUTomotive Open System ARchitecture (AUTOSAR) (4.0), increased application of formal methods, improvements on static code analysis and the use of Natural Language Processing (NLP) for automated

analysis of requirements. The different areas often involve multi-core architectures and might be considered an indicator that Bosch is considering this area to be of great importance.

9.3.1. New AUTOSAR 4.0 ECU Platform

Bosch has planned to publish and move to a new generation of its platform of Electronic Control Units(ECUs) for powertrain control at the end of 2014. The new platform generation (called MDG1) is based on a fully AUTOSAR compliant base software (BSW) stack and aims to move the application software into compliance gradually as well. The decision to use AUTOSAR 4.0 specifically was driven mainly by the desire to utilize multi-core chip sets. MDG1 succeeds both EDC17 (diesel engines) and ME(D)17 (gasoline engines) [122].

9.3.2. Event-B and RODIN Toolset

Gmehlich et al. have published multiple papers describing case studies in cooperation with Bosch with the goal of evaluating formal methods in practice. The technology used was the RODIN Platform and the Event-B formal method. The result was that the method is effective, but does not scale well. One important difficulty was modeling time and as a consequence real-time applications [57, 59, 58].

9.3.3. Worst Case Execution Times

Worst Case Execution Time (WCET) in multi-core systems with shared resources was the subject of research performed by Bosch in cooperation with Karlsruhe Institute of Technology (KIT). The goal is to mitigate the effect of greatly increased execution time in a small number of cases when the normal case has very low execution time. The method that was used utilizes a second core to help process those cases and proposes an improved heuristic to predict when a second core will be needed [49].

9.3.4. Formalization of Requirements

There are multiple papers (by the same researchers) reporting on case studies on the formalization and automatic analysis of natural language requirements in the domain of real-time systems. A tool chain was developed and tested on multiple automotive projects at Bosch. The results show that the time investment and computational cost are acceptable and benefit is significant [114, 115, 116, 117].

9.3.5. Requirements, Models, Test Cases

US based researchers working for Bosch have developed methods for analysis of test cases in form of vectors representing input and output values to automatically construct invariants that represent relations between those vectors. The base data is obtained trough data mining [1].

The approach was tested in a pilot project on production C source code supplied by Bosch. Experimental data indicates that full-coverage test data applied iteratively produces the best results [1].

This research uses a tool called Reactis, which is supplied with a model and generated test cases, which it then refines to extract invariants [1].

9.3.6. Static Analysis of Race Conditions

Bosch supported research into a static analysis tool that improves the off-the-shelf software The Bauhaus Toolkit (Bauhaus) that is commercially distributed by Axivion GmbH. The goal was to improve false positives in the static analysis performed by Bauhaus and additionally produce true positives by itself. The goal of the research was reached, with significant improvement in both areas. The method used was *Lockset Analysis* [77].

9.4. Sources at Bosch GmbH

- **DI (FH) Alexander Moiszi** (Tamm, Germany) [96]
  Project Manager for Testing Technology, Advanced Development and Technology Supply
- **Johannes-Joerg Rueger** (Gerlingen) [122]
  Diesel Gasoline Systems, Electronic Controls
- **Alexander Wernet** (Gerlingen) [122]
  Diesel Gasoline Systems, Electronic Controls
- **Hasan-Ferit Kececi** (Gerlingen) [122]
  Diesel Gasoline Systems, Electronic Controls
- **Thomas Thiel** (Gerlingen) [122]
  Diesel Gasoline Systems, Electronic Controls
- **Rainer Gmehlich** (Schwieberdingen, Stuttgart) [57, 59, 58]
- **Felix Loesch** (Stuttgart) [59]
- **Katrin Grau** (Stuttgart) [58]
- **Matthias Freier** (Schwieberdingen) [49]
  Corporate Sector Research
- **Amalinda Post** (Stuttgart) [114, 115, 116]
  Research and Advance Engineering
- **Igor Menzel** (Stuttgart) [117]
  Corporate Research
- **Charles Shelton** (Pittsburgh) [1]
  Robert Bosch Corporation Research and Technology Center
- **Elizabeth Latronico** (Pittsburgh) [1]
  Robert Bosch Corporation Research and Technology Center

10. Volkswagen

Volkswagen (VW) AG is a major car manufacturer and the name giving brand of the Volkswagen Group. They have sold 2.5 million vehicles in 2013 and employed 107k people while generation 65.5 billion in revenue [63].

10.1. Recent History and Current State

Volkswagen uses its own interpretation of the widely accepted and used V-Model (fig. 10.1). The model is split in ten phases, which is a remarkably high number [5]. These phases are called at Volkswagen as follows:

1. System Requirements
2. System Architecture
3. Software Requirements
4. Software Design
5. Software Creation / Code Generation
6. Module Test
7. Integration Test
8. Software Test - Functional
9. System Integration Test
10. System Test Approval

Additionally, the illustration suggests, that the process is additionally using the circular or spiral model and produces a new *sample* every cycle until series maturity is reached [5].

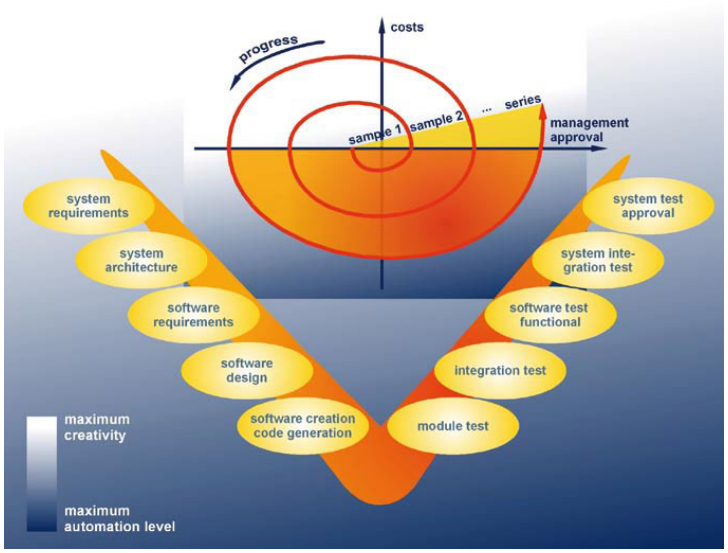

Figure 10.1.: V-Model at the Volkswagen Group [5]

10.1.1. Functional Software Development

The development of functional software is according to [5] also divided into four principal components:

1. Requirements Analysis
2. Modeling
3. Tool Chain (= Realization)
4. Strategies for Testing

VW develops the functional software in-house but delegates the development of base software (BSW) to suppliers [5].

10.1.2. Modeling

VW uses MatLab/SimuLink (ML/SL) for modeling. Implemented functions in the model are deployed into a test vehicle as soon as possible in an combined effort between the software development and test-drive engineers.

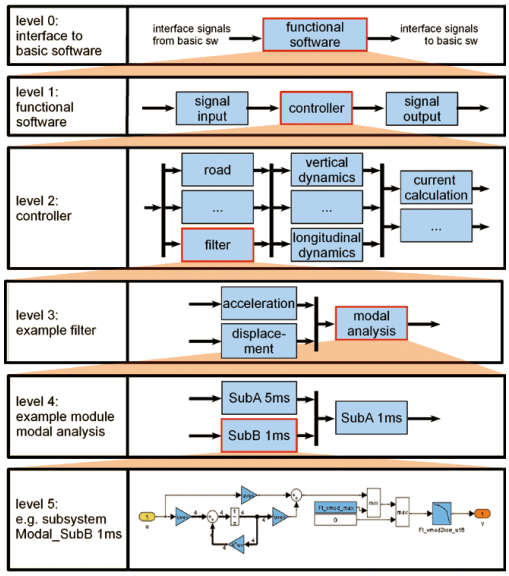

Figure 10.2.: Model hierarchy at VW [5]

VW reports highly positive effects of this approach on time efficiency, internal expertise transfer and avoidance of interface problems. The model is hierarchical and supports traceability of changes [5].

10.1.3. Tool Chain

The different parts of the tool chain are abstracted into their functions. The actual software products are not named (fig. 10.3). The different parts according to [5] are as follows:

- **Rule Checker:** Verifies adherence to modeling guidelines
- **Docu-Gen:** Generates the Documentation
- **Code-Gen:** Generates source code in .c and .h files
- **Calibration-Gen:** Extracts predefined variables relevant for calibration and provides them in .c, .h and .a2l files.
- **Interface-Gen:** Generates source files defining the interface between functional software and BSW.

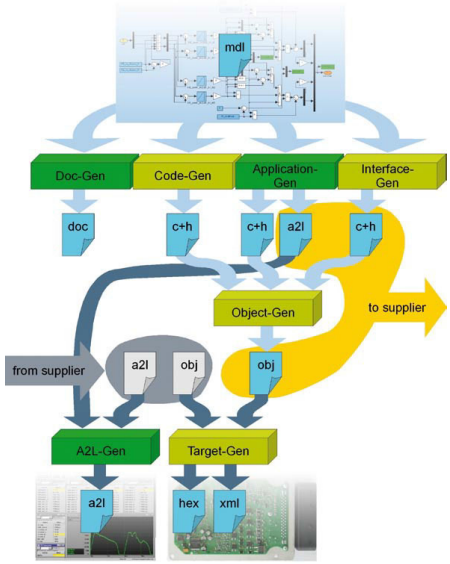

Figure 10.3.: Development Tool Chain at VW [5]

- **Object-Gen:** Compiles the .obj file from sources.

The .obj file, .a2l file and the original source files for the interface are sent to the supplier. Similarly, the .a2l and .obj file for the BSW are received from the supplier in return [5].

10.1.4. Testing

Almost all tests can be done without the supplier-provided base software stack. Manual checks are still performed for evaluating resource utilization (memory, runtime) and for errata. The hardware used for testing is of production grade [5].

Module and integration tests check Electronic Control Unit (ECU) against model behavior (back-to-back testing). Reviews are used to check the source (written in The C Programming Language (C)) against the ML/SL model.

Functional module tests are automated and take place on the Hardware-in-the-Loop (HiL) but the majority of system tests are done as vehicle tests in a test vehicle [5].

10.2. Process Similarity within Volkswagen Group

The amount of papers about VW that are available and deal with the topic of software development is not very large. Despite the limited direct sources it seems reasonable to assume that the development process at VW is very similar to other companies within the VW Group. Engineers of AUDI have cited the plan to roll out the process used at AUDI to the whole VW Group as early as 2009 [78] but have also more recently mentioned its active use at both AUDI and VW in 2010 [140] and in 2011 [134].

One method that was co-developed with Audi was the Integrated Testing Framework (ITF) (sec. 4.2.8), another one was the Virtual Test Drive (VTD) system for test drive replay and simulation of traffic and environment (sec. 4.2.7) [95].

10.3. Sources at Volkswagen AG

- **Ing. Dr. Andrea Arenz** (Wolfsburg, Germany) [5]
  Team Leader Functional Software Damper Control in the Car Chassis Development (until May 2008)
  Team Leader Quality Assurance Launch Support Electrics

11. Ford Motor Company

Ford Motor Company (Ford) is one of the largest automobile manufacturer in the world with 180,000 Employees worldwide and revenues of almost 150 billion. It is based in Dearborn (Michigan, USA) but has representations internationally [30].

11.1. Recent History and Current State

Ford seems to have been far ahead of the curve of software development practices in the automotive field in its infancy. As early as 1998 Ford Research was already releasing papers on Model Based Development (MBD), model-based code generation and its validation. It is even explicitly stated that models are being used in a development model called *System V* (fig. 11.1) that seems very similar to the well known V-Model used broadly today [135].

This early research does not seem to have translated into a discernible advantage compared to other top-tier manufacturers of today. As an example, it has adopted 24/7 automated testing on Hardware-in-the-Loop (HiL) testbenches in 2010 (sec. 11.1.3) and test generation from models (sec. 11.1.1) in 2012, which does not put them particularly far ahead of the curve.

11.1.1. Model-Based Design and Code Generation

The team that was responsible for the creation of the battery control system for the 2010 Ford Fusion Hybrid reports that they were the first at ford to use MBD in connection with automatic code generation (called *autocode* in the report) for a production program. About 80% of the final product

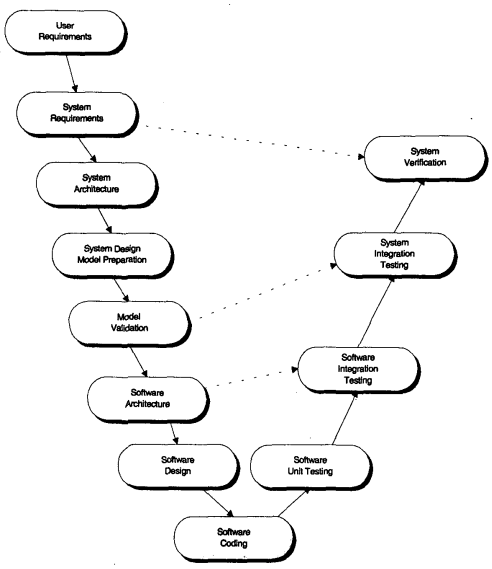

Figure 11.1.: Ford System V Development Model (1999) [135]

code was auto-generated (exception: low-level routines). The development team used dSPACE TargetLink (TargetLink) for the complete process from *function design to software implementation* [55]. TargetLink works off of MatLab/SimuLink (ML/SL) models which is a strong indication for ML/SL as a modeling tool. Model-in-the-Loop (MiL) and Software-in-the-Loop (SiL) simulation was used for frontloading of verification using the built in simulation environment of TargetLink which enabled fast switching between the two modes for comparison [55].

Challenges they faced include the software version control that was not equipped for models but for text file merging/branching and also the build automation that had to be created from scratch. They overcame these and report a complete build cycle time (model to hex binary) of about 30 minutes. They have had no issues in the field that can be traced to the automatic code generation at the time of the report [55].

This report gains weight as good representation of the status quo at ford in 2010 because the team responsible was awarded the *Henry Ford Excellence Award* for the software [55].

11.1.2. Model-Based Testing with Messina

A report from *Ford Werke* in Cologne (Germany) reports the use of Berner & Mattner MESSINA (MESSINA) software platform for testing of Ford's Lane Keeping System (LKS) created with MBD. They use virtual integration, environment simulation and systematic automated testing, all based on MESSINA. The functional models were already available. They were simulated in Dyna4 Suite by Tesis and were integrated into MESSINA in a Model-in-the-Loop (MiL) set-up. Test scenarios were generated (not constructed) by using variation in parameters. Sensible parameters and ranges for variation were defined using Classification Tree Editor (CTE). Tests were recorded for re-use in back-to-back testing testing on the HiL simulator in later stages, but back-to-back testing of software and models was performed on the spot [65].

11.1.3. Hardware-in-the-Loop Lights Out Testing

Ford has increased the HiL capacity from 5 to 13 simulators in its testing lab and undisclosed numbers in other locations (Cologne, Dunton). At the same time they moved to *minimal supervision automated testing* that can be performed 24/7 with dSPACE AutomationDesk (AutomationDesk) software. They also introduced Automotive Simulation Models (ASM) for vehicle and environment simulation [56].

Ford had been using HiL for many years before, but did not have the automation in place on this scale. The set-up also performs an automated diagnostics process with Electronic Control Unit (ECU) state diagnosis with trouble codes programmed into the firmware [56].

11.1.4. Standardization of Controller Interfaces

In 2011 Ford reports on efforts of standardizing, consolidating and simplifying the interfaces of on-board controllers. They recognized a need for a rework of the *status quo* in 2005 when a international deployment of a battery pack turned up unexpected difficulties. The resolution of the problem failed

to be executed in time because of mismatches between controllers and the whole project was canceled. This led to the establishment of an architectural framework for control software to alleviate that problem [137].

The effort is ongoing and was in 2011 expected to be used in 2013 when the first production use of largely standardized signals is expected to happen. The result of this effort is a reduction of variation in interface signals but no clear indicators on the actual level of benefit were available yet at the time of the report [137].

11.1.5. Software Life-cycle Management

Siemens Teamcenter (Teamcenter) software had been used at Ford for Product Life-cycle Management (PLM) of mechanical products when it was decided to expand that cooperation to software artifacts in 2010 with the Teamcenter digital life-cycle solution [71] (Note: Article released in 2014 but wording suggests 2009/2010 at the latest as likely implementation year).

The goal of this move was to enable repair of ECUs without replacing the ECU or even dismantling parts of the vehicle to allow physical access. Hardware replacement is slow, expensive and limited by supply and availability of replacement parts. Software updates have none of those problems. Ford equips their ECUs with flash memory to enable deployed hardware flashing [71].

Teamcenter transfers traditional properties like options and variants to the software domain properties like configuration and platform. It also tracks important attributes (memory size, memory address space, programming protocol,...) and can validate these. It can further employ requirements validation for issues (format, file part number, test cases,...) [71].

Another important function is monitoring and tracking software dependencies between the ECUs and ensuring these are satisfied. Because of tracking on the basis of the Vehicle Identification Number (VIN) all of these checks can be done on the level of each individual vehicle separately. Teamcenter also manages the global roll-out of patches and updates to service centers with global distribution broadcasts [71].

11.2. Future Directions and Challenges

Ford seems to be interested in improving their model based testing and test case generation methods.

11.2.1. Model-Based Testing with Mogentes

Ford took part in the European Union (EU) funded research project Model-based generation of tests for embedded systems (Mogentes) that was lead by the Austrian Institute of Technology (AIT) (ETH Zurich, TU Graz,...) and sought to automate the discovery of test cases and simultaneously enable reasoning about the coverage of requirements and the absence of certain kinds of errors. Ford contributed a theft prevention system and simplified software in the form of a state machine. The company Prover Technology AB (based in Stockholm) extended their software with new methods for that study [139].

Ford was not especially interested in the *white-box* testing or fault injection with available source code but concentrated instead on *black-box* testing because that is typically the given situation in the automotive industry according to Ford. The most interesting seems to have been error modeling on the basis of Unified Modelling Language (UML) requirements models. The approach is to check which inputs produce different outputs when the model is changed. Those relations between inputs and outputs constitute the error models which are the result of this approach [139].

The Mogentes project produced a number of tools that support the generation of efficient test case collections. Ford plans to move to activity diagrams in the future because state diagrams are too limited for modeling of complex systems. Another future goal is to add the use of fault injection [139].

11.2.2. Formal Validation Suite

Ford has developed a suite of tools to perform validation and other tasks. This suite consists of three tools:

1. **Cyclops**: A model preprocessor that performs static checking of models to discover common errors of UML models specified in Extensible Markup Language (XML) Metadata Interchange (XMI)
2. **Hydra**: A tool for model conversion from UML to Process/Protocol Meta Language (PROMELA), which is used as input for the Simple PROMELA Interpreter (SPIN) model checker.
3. **Marple**: A tool for automatically generating properties that may be latent in the model and should be analysed.

In combination these tools can be used to iterate over a model, improve it in every iteration and ensure its proper functioning without unwanted properties. Ford has used the suite in pilot projects and plans to improve on it and add functionality [74].

11.3. Sources at Ford

- **Florian Frischmuth** (Dearborn) [56]
  E/E Systems Engineering (EESE) Global Embedded Software Manager
- **Wajiha Chahine** (Dearborn) [56]
  E/E Validation Group Supervisor
- **Jim Swoish** (Dearborn) [55]
  HEV HV Battery Controls & Software Supervisor
- **Edward C. Nelson** (Dearborn) [74]
  Ford Research and Advanced Engineering
- **DI Dirk Gunia** (Cologne) [65]
  Group Leader
  Responsible for devel. of camera-based driver assistance systems
- **Dipl.Phys. Ekkehard Pofahl** (Cologne) [139]
  Electrical Integration
- **DI Johannes Wiessalla** (Aachen) [139]
  Vehicle Dynamics
- **DI Otto Hofmann** (Aachen) [139]
  Vehicle Dynamics
- **DI (FH) Thomas Lenzen** (Cologne) [139]
  Electrical Integration

- **Patrick Milligan** [71]
  Senior Manager for Vehicle Solutions
- **Martin Baker** [71]
  Global Manager Software, CAE and Process, Methods and Tools
- **Chris Davey** [71]
  Technical Leader, Software and Control Systems Engineering
- **Anthony Tsakiris, MSc** [137]
  Technical Expert

12. Toyota Motor Company

Toyota Motor Company is an international car manufacturing company based in Toyota, Japan. For the fiscal year 2013 Toyota reported revenues of 234 billion dollars and 333,000 employees [32]. When comparing revenue that puts Toyota as the number two worldwide between Volkswagen Group (€197b, $270b [63]) and Daimler AG (€118b, $161b [37]).

12.1. Recent History and Current State

Toyota has had a rough year in 2013 when courts ruled against it in the case of Unintended Acceleration (UA) in one of its vehicles that caused the death of the driver and injury to the passenger in a 2007 crash. The injured woman and the family of the killed woman were both awarded 1.5 million in damages by the court [118].

The interesting part of this incident for this thesis is the insight that was provided by the expert witnesses on the state of Toyota's software development practices and the resulting artifacts (source code and binaries). Michael Barr, a expert witness, gave a testimony in court [138] and has published his slides that go along with it [11]. In the interest of brevity the findings will be limited to excerpts and important points. Interested parties are advised to read Mr. Barr's testimony [138] and presentation [11].

In his testimony Mr. Barr characterizes Toyota's safety critical software as a *"house of cards" safety architecture* that was compiled from spaghetti code. He reports the excessive use of global variables (over 11000) and 67 functions scoring *over 50* (untestable) on cyclomatic complexity with the throttle angle function scoring *100* (unmaintainable). He further found that

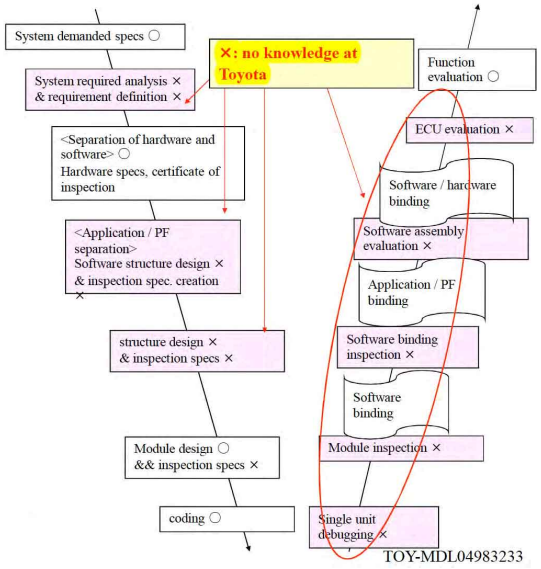

Figure 12.1.: Toyota development process (Camry L4 2005) [11]

Toyota's coding standard only contains 11 MISRA-C rules out of over 100 and 5 of those 11 were violated in the actual source code [11].

Mr. Barr criticizes Toyota's software process and presents an illustration that shows it's inadequacy (fig. 12.1). He also cites internal e-mails from 2007 that show that key Toyota employees were aware of the fact that, though it was recognized as one of the major strengths of Toyota, the fail-safe technology was in bad shape. The e-mail describes the cultural problem that *failsafe is not part of the Toyota Engineering Division DNA* [11]. The suggestion of internal experts towards superiors was to *benchmark Bosch to gauge strengths and weaknesses* [11].

In conclusion, the state of Toyota software development was very bad up until only a few years ago, which also partly explains the lack of documents that could be found for this thesis that were authored before 2009 and could be sourced to Toyota.

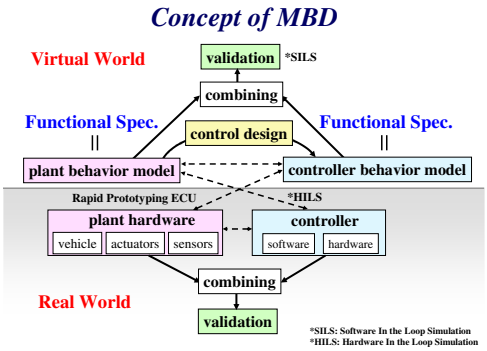

Figure 12.2.: Model Based Development, Ohata [107]

12.1.1. Signs of Recovery

A joint committee of the Society of Automotive Engineers of Japan (JSAE) and Society of Instrument and Control Engineers (SICE) was founded in 2010 to facilitate the cooperation between the automotive industry and the academia of Japan. The committee is called Joint Research Committee - Automotive Control And Modelling (JRC-ACM) and Mr. Ohata, an employee of Toyota who has published about Model Based Development (MBD) since 2009 seems to be a member of it [106]. The publications typically revolve around plant models and overall modeling process (e.g. fig. 12.2). The effect of this involvement on Toyota's MBD efforts could not be judged from the documents available.

12.2. Future Directions and Challenges

Toyota seems to have recognized the failings of its software development process and has began the move towards a modern, model-based safety-critical software development. They have token steps to reinforce current systems with model-based checks and use them as Software-in-the-Loop

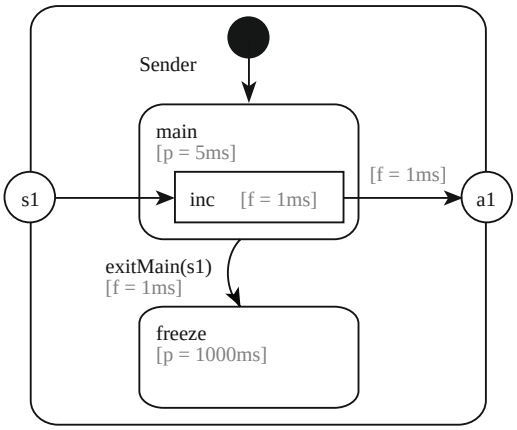

Figure 12.3.: Timing Model Visual Representation [120]

(SiL) components while they build the next generation of software that is entirely state-of-the-art and verifiable from requirement to binary.

12.2.1. Timing Models and Legacy Software Migration

An effort that seems promising comes from US based researchers of Toyota working with the University of Salzburg. In 2011 they published a paper titled *Migration of Legacy Software Towards Correct-by-Construction Timing Behavior* [120]. Besides the very telling name the paper has the description of a migration towards explicit specification in the form of Timing Definition Language (TDL) models (fig. 12.3). The created models were used to validate the timing behavior of a small part of the legacy software component (SWC) at a time and decide it's *schedulability*. If the part did not satisfy the model constraints then it could be modified to correct that. After satisfying compliance is achieved another part can be added. This allows for incremental verification of complex legacy software [120].

12.2.2. Hierarchical Accumulative Validation (HAV)

In 2013 Toyota published the details of its new model-based software-development process in the SAE International Journal of Passenger Cars [47]. The new process is called Hierarchical Accumulative Validation (HAV) and derives its name from the multi-tiered model composition (hierarchical, fig. 12.5) and its layered, incremental application of Modified Condition/Decision Coverage (MC/DC) validation (accumulative, fig. 12.6). However, the authors of the publication do seem to be very careful not to confirm that the presented model is actually in use already or that it definitely will be in use in this form by using wording like "Toyota confirmed that HAV **can be** implemented to successfully validate executable control specifications for production control system development." [47] (Note: emphasis added) consistently throughout the text. This is also one of the reasons that the model is ruled to be a *future direction* rather than *current state*, the others being the short time period available for the transition and the large amount of legacy code that needs to be transitioned.

The HAV has many similarities to the commonly used V-Model and the modeling practices at other manufacturers. The basic shape of the *V* and the tiered iteration cycles can be seen the illustration (fig. 12.4). Still, the overall impression is that Toyota has *not* copied any outside system but has in fact created every aspect to fit its requirements (e.g. rejection of mainstream tools, sec. 12.2.2).

Development Process Overview

The basic process starts with the graphical design of executable models (called *control specifications*) by control designers. These are validated against requirement specifications during the design process. Once all models required are created, the source code is automatically generated. The model architecture is hierarchical [47]. There are three tiers of models:

1. **Modules** (lowest): These do not have functionality of their own but serve as components for higher levels

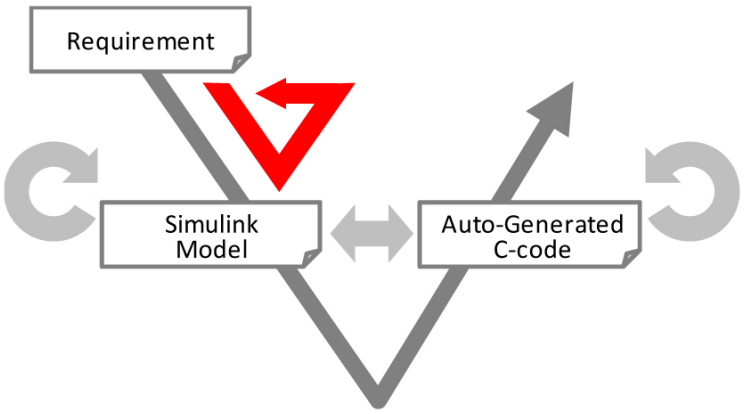

Figure 12.4.: Toyota new development process [47]

2. **Sub-Functions**: Groups of modules with functionality that can be tied to requirements
3. **Features**: A directed (execution flow) combination of sub-functions (called *functional hierarchy*) that performs a high-level functionality.

After all necessary models are available, it is attempted to achieve MC/DC coverage by incrementaly validating bottom-up and back-to-front. This means that modules are validated before sub-functions and subfunctions are validated in reverse direction of execution flow. This continues until complexity is too high for successful MC/DC coverage. After that the validation switches to Model Based Testing (MBT) [47].

Testing utilizes a combined Model-in-the-Loop (MiL) and SiL set-up in which legacy software components can be used in a SiL simulation to complete the system for integration testing of models and generated code [47]. The software for this simulation seems to be provided by MathWorks in the form of MatLab/SimuLink (ML/SL), Stateflow and MathWorks Simscape (Simscape) (engine model). Source Code is generated with Simulink Coder (formerly called Real-Time Workshop). Debugging is done is Microsoft Visual Studio (VS) which is connected to ML/SL, which enables source-level debugging of control code. Breakpoints in VS pause the execution of the model in ML/SL to allow examination of the state of variables before resuming code execution [72].

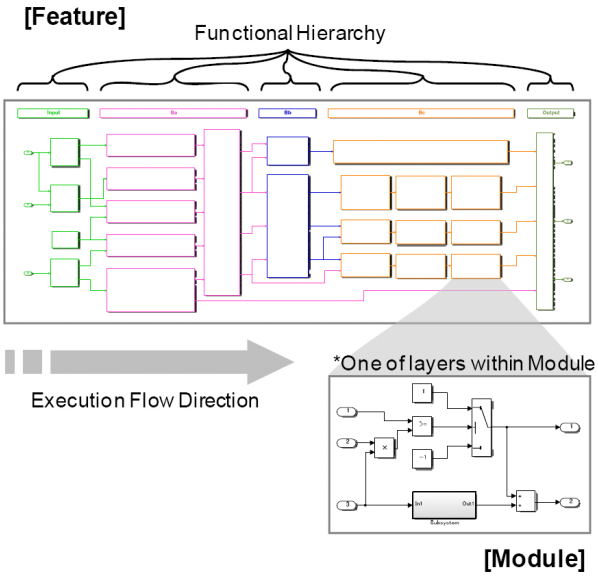

Figure 12.5.: Toyota model hierarchy [47]

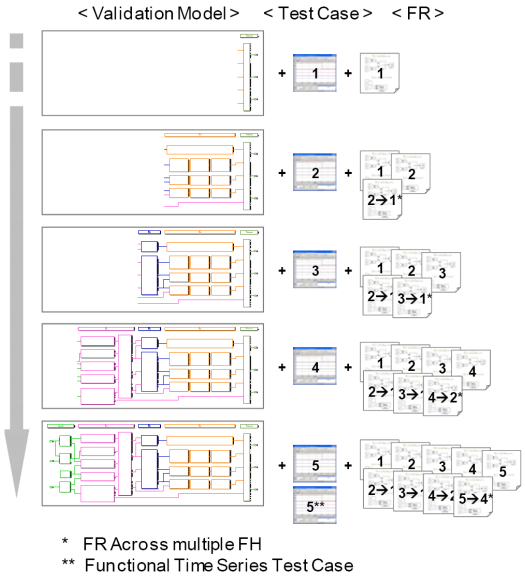

Figure 12.6.: Toyota accumulative coverage [47]

Problems and Temporary Solutions

There have been some problems with Toyota's new process as well. One major problem that slows the adoption of the process is the apparent lack of commercially available tools that satisfy Toyota's requirements. As an example, Toyota requires the connection of models into tiers for retrievals with queries like "at model level or higher" [47] or "with these desired outputs" [47]. There were also problems with test case generation tools that required constant manual adjustment or did not support generation of code for groups of models (sub-functions, features). Rather than compromise on their development process, Toyota has decided to work with temporary solutions like simple (but strictly adhered to) folder structure until commercial tools adapt to their needs or in-house tools are completed [47].

12.3. Sources at Toyota

- **Akira Ohata** (Susono, Shizuoka, Japan) [107, 106]
  Senior General Manager, Engine Control System Development
- **Dr. Hisahiro Ito** [72] Assistant Manager
- **Jared Farnsworth** (Erlanger, Kentucky, USA) [47]
- **Koichi Ueda** (Erlanger, Kentucky, USA) [47]
- **Hideaki Mizuno** [47]
- **Michio Yoshida** [47]
- **Kenneth Butts** (Ann Arbor, Michigan, USA) [120]

13. Other Companies

There are a number of techniques and methods that are often mentioned but either can not be easily attributed to one company or there was not enough information on the company to fill its own chapter.

13.1. MARTE, SysML and EAST-ADL/2

MARTE is a Unified Modelling Language (UML) profile for Modeling and Analysis of Real-Time and Embedded Systems [61] which is interesting for its ability to model timing constraints and is seen by some companies (e.g. Continental [4]) as possible competitor of AUTomotive Open System ARchitecture (AUTOSAR) in that respect (AUTOSAR does not have that ability).

Systems Modelling Language (SysML) is a UML2 profile that defines a "general-purpose graphical modeling language for specifying, analyzing, designing, and verifying complex systems that may include hardware, software, information, personnel, procedures, and facilities" [62].

EAST-ADL is a Architecture Description Language (ADL) that is aligned with AUTOSAR and meant for describing the architecture of embedded systems on multiple abstraction levels which map to *ISO26262* [6]. All three languages mentioned above are mainly found in publications by Volvo, Continental and AVL List [28, 113, 36, 29]. Between the three languages, SysML is the only one that has documents from outside the mentioned companies (Bosch [85], Siemens [141], Ford [98]), albeit only a few.

Besides the mentioned extensions there is definitely a trend towards proprietary company-specific UML profiles and extensions (Audi [140], Ford

[139], Siemens [129], Daimler [112]). UML alternatives are in use in at least one major company, with BMW using Advanced Simulation and Control Engineering Tool (ASCET) models [97] that can be converted to and from UML on demand [124].

13.2. EAST-ADL, UPPAAL-PORT and Volvo

Volvo engineers have consistently published about Electronics Architecture and Software Technology - Architecture Description Language (EAST-ADL) since its inception in 2006 and most recently in 2013. In 2011 there were reports about efforts to combine EAST-ADL with UPPAAL Partial Order Reduction Techniques (UPPAAL-PORT) and create a tool chain for "Verifying Functional Behaviors of Automotive Products" [76]. These efforts presumably contributed to the creation of **V**erification **T**ool for EAST-**A**D**L** Models using UPPAAL-PORT (ViTAL), that bridges the gap between those two [45].

Part III.

Evaluation

14. Introduction

The goal of this part is to evaluate the information gathered in part II and coalesce it into an overall picture. This should serve to allow the reader an insight into a certain phase or the overall process of the V-Model across the industry as a whole, instead of focusing on the process at any particular company.

In this part the author also aims to offer a rating system that enables the reader to judge characteristics of a particular method at a glance. A method is a process, technique or technology of interest that is being used or developed and can be identified as separate entity.

14.1. Structure

Part III is organized along the structure of the V-Model and every phase of the V-Model corresponds to a chapter in this part. To enable this structure and relationship, it was necessary to generalize the V-Model implementations that are being used by different companies. The resulting consolidated V-Model structure can be seen in figure 14.1. The phases and chapters are named as follows:

- Requirements Management (chapter 15)
- Modeling and Simulation (chapter 16)
- Test Cases and Prototyping (chapter 17)
- Implementation (chapter 18)
- Component Testing (chapter 19)
- System and Integration Testing (chapter 20)
- Software Monitoring (chapter 21)
- Software Maintenance (chapter 22)

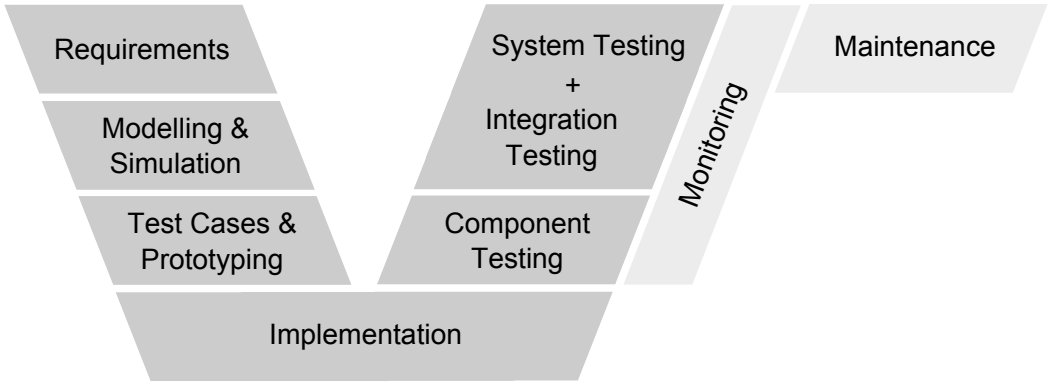

Figure 14.1.: V-Model for part III structure

Software Monitoring and *Software Maintenance* are two additional chapters to help complete the software life-cycle beyond development. To find out more about a particular phase, please refer to the appropriate chapter.

14.2. Rating System

Every method is evaluated on four different criteria (besides the assignment into one of the phases of the V-Model). Two of the criteria, *Prevention* and *Verification & Validation* (V&V), are simple binary (yes/no) values denoted by either a green check mark (✓, yes) or a red cross (X, no). The other two criteria are rated on a scale of 0..5 denoted by the appropriate number of yellow stars.

Prevention

The evaluation of prevention is based on its dictionary definition from Merriam-Webster that defines it as being *the act or practice of stopping something bad from happening* [42]. Setting this definition in the context of Model Based Development (MBD) leads to the definition used which is **prevents fault or failure of the software component or system as a whole**.

Verification & Validation

For the evaluation of V&V the definition endorsed by the IEEE is used [70]. That definition reads as follows (emphasis added by author):

> **Validation**. The assurance that a product, service, or system meets the needs of the customer and other identified stakeholders. It often involves acceptance and suitability with external customers. Contrast with verification.
>
> **Verification**. The evaluation of whether or not a product, service, or system complies with a regulation, requirement, specification, or imposed condition. It is often an internal process. Contrast with validation.

Combining the two and setting the definition in context if MBD results in the short definition used for this evaluation which is **ensures compliance with requirements or specifications**. This definition is deliberately chosen to exclude transitive relationships like *helps create requirements, therefore helps ensure compliance* because the inclusion of those would potentially evaluate to *yes* on every method and therefore be useless.

Prediction

The evaluation of prediction is based on its dictionary definition from Merriam-Webster that defines its verb as being *to declare or indicate in advance; especially: foretell on the basis of observation, experience, or scientific reason* [41]. Setting this definition in the context of MBD leads to the definition used which is **helps foretell reliability of a software or system on the basis observation, experience, or scientific reason**.

Deployment

Deployment is used to express the degree to which a method is matured and moved on from research and has been deployed into regular production use. The possible values are used to mean the following:

★★★★★	Not used, no interest in using (excluded from thesis)
★★★★★	Only used in research
★★★★★	Pilot projects interest in using
★★★★★	Plans for deployment by at least one major company
★★★★★	Already deployed by at least one major company
★★★★★	Industry standard, deployed by most companies

Source Reliability

Source Reliability is used to express the degree to which the documents used to explain a method and judge its **Deployment** rating are from reliable and knowledgeable sources. To classify any particular source on its reliability it is evaluated on:

1. the affiliation of the authors with the relevant companies
2. their position within that companies hierarchy
3. the relevance of their department and/or their job description to the methods domain of use

This evaluation leads to assignment into one of the following rating categories:

★★★★★	External people, unaffiliated (excluded from thesis)
★★★★★	Researcher/s working with one company
★★★★★	Researcher/s working at one of the companies
★★★★★	Engineer/s employed at one of the companies
★★★★★	Engineer/s in relevant department at one company
★★★★★	Engineers in relevant departments at multiple companies

In general, multiple people, multiple companies and relevant positions are viewed as indicators of reliability.

15. Requirements Management

The first phase in the typical V-Model is about the analysis and formalization of requirements (see sections 4.2.1, 5.2.1, 6.2, 7.1.1, 9.2, 10.1).

Correct execution in this phase is very important. The part of the V-Model that is relevant for software development could easily be compared to the Waterfall Model but with additional iterative characteristics. In both models there is Big Design Up-Front. Although the V-Model iteratively re-visits the requirement analysis phase it is still costly to correct mistakes later on. Errors propagate through all other phases and corrections also have to do so which is time and resource intensive.

Project relevant documents usually exist before the project itself is even started. These pre-existing documents contain but are not limited to functional and technical specifications, relevant standards and regulatory requirements. Domain experts are employed to condense all of that information into a single collection of architectural, system and component constraints (requirements) [110, 97].

15.1. Method: Natural Language Processing

The body of relevant information can be very large, in some cases over 3000 pages. This puts a large burden on a small number of experts, limiting throughput, quality and efficiency. There is a large overhead because experts have to search through multiple documents and keep track of conflicting information [110]. Audi and Daimler have developed methods to ease the burden and improve the situation for the expert and the business process.

The approach of Daimler involves the semantic analysis of documents to automatically classify text passages and group them together according to their content [110].

Audi reports the same efforts but additionally the goal of storing business rules and building an ontology. The semantics of text documents then get linked to the ontology concepts for a multitude of benefits [121, 108, 123].

Rating

Deployment	★★★★★	2+ companies, millions invested
Reliability	★★★★★	Researcher (Daimler), Engineer (Audi)
Prevention	✓	Helps with prevention
V&V	✓	Requirements are base for V&V
Prediction	X	No relation

15.1.1. Method: Extraction from Test Cases

US based researchers working for Bosch have developed methods for analysis of test cases in form of vectors representing input and output values to automatically construct invariants that represent relations between those vectors. The base data is obtained trough data mining. The approach was tested in a pilot project on production C source code supplied by Bosch. Experimental data indicates that full-coverage test data applied iteratively produces the best results [1]

Rating

Deployment	★★★★★	Pilot project
Reliability	★★★★★	Researchers(?) at Bosch
Prevention	✓	Helps with prevention
V&V	✓	Requirements are base for V&V
Prediction	X	No relation

15.1.2. Method: Formalization of Restricted Grammars

The companies in the scope of research typically use off-the-shelf software for the storage and management of individual requirements in the form of natural language (potentially in simplified/restricted form). BMW [97], AUDI [78], Daimler [110, 111, 22, 81] and ZF Friedrichshafen AG (ZF) [102, 12] use IBM Rational Dynamic Object Oriented Requirements System (DOORS) while Siemens [129, 14, 35] uses Sparx Systems Enterprise Architect (EA) for this purpose. There is no clear information of the practice in the remaining companies (VW, Bosch).

The stored artifacts have a Unique ID (UID) to enable tracking the connection of requirements to parts of models and source code functions (BMW [97], ZF [102]).

There are multiple papers (by the same researchers) reporting on case studies on the formalization and automatic analysis of natural language requirements in the domain of real-time systems. A tool chain was developed and tested on multiple automotive projects at Bosch. The results show that the time investment and computational cost are acceptable and benefit is significant [114, 115, 116, 117]

Rating

Deployment	★★★★★	Multiple pilot projects
Reliability	★★★★★	Researchers at Bosch
Prevention	✓	Helps with prevention
V&V	✓	Requirements are base for V&V
Prediction	X	No relation

16. Modeling and Simulation

The available materials indicate that Model Based Development (MBD) with Big Design Up-Front is the unanimous consensus for best practices in the automotive industry (see chapter II).

16.1. Big Design Up-Front

The typical design strategy is top-down design which follows from the structure given by the V-Model which many automotive companies are obliged to follow. This process usually follows steps identical or similar to the following:

1. System / Architecture Design
2. System / Architecture Modeling
3. Software Design & Modeling
4. Component Design & Modeling

16.2. Method: Collaborative Design

One very important step in the early phases of a project is the collaborative development of the overall architecture, system layout, technology choices and design decisions that greatly affect the end result. Even though this process is not limited to the early stages it is very dominant at the start of a project.

Despite this importance there is not a large amount of sources dealing with it originating in the automotive industry. Presumably this is because of the

impression that this topic is not interesting enough for scientific papers. It was necessary to research the personal reports of everyday work by software architects to secure sources that confirm the facts in this matter. Only Siemens [24, 23] and AVL List [80] are explicit in their papers about reviews, walkthroughs and inspections taking place and having an important role in the development of the design and later artifacts.

Rating

Deployment	★★★★★	Already used at Siemens
Reliability	★★★★★	Senior Engineer (Siemens)
Prevention	✓	Helps with prevention
V&V	X	No relation
Prediction	X	No relation

16.3. Modeling

Models are the natural first development target as they are the basis for MBD. The design language is usually UML with varying profiles but can be EAST-ADL when modeling architectures.

There are different kinds of models:

1. Architecture Models: Model the hierarchy, dependency and composition of the system and parts of it.
2. Component Models: Model a single part or *function* (e.g. MBFS [52]).
3. Behavior Models: Explicitly model and specify the behavior of a part.

For implementing those, some of the following diagram types can be used:

- Class Diagrams
- State Diagrams
- Activity/Sequence Diagrams
- Control/Data Flow Graphs
- Circuit Diagrams
- etc.

16.4. Method: Invariant Specification

There are some aspects of the system under test (SuT) that are either difficult to model because of their continuous temporal nature, or difficult to assign to a specific model in the system. One example could be the charge level of the battery or the overall system temperature. These can be modeled by using invariants, sometimes called *watchers* or *watcher instruments* [102].

These *invariants* are not models in the conventional sense, but grouped together may be seen as an extra model parallel to the overall system model.

While the explanation of the *Watcher Instruments* comes from ZF [102] the same software is used at Daimler [54], which makes it probable that Daimler is also taking advantage of them.

Rating

Deployment	★★★★☆	Already used at ZF
Reliability	★★★★☆	Head of Engineering Dept. (ZF)
Prevention	✓	Helps with prevention
V&V	✓	Serves V&V
Prediction	✓	Prediction trough V&V

16.5. Simulation

Models are often built to be executable (simulation) and able to be integrated into a testing process. This can be done both for checking the model but also to validate later stages of production. More on simulation can be found in the appropriate chapters dealing with prototyping and testing (e.g. chapter 17, 19, 20).

17. Test Cases and Prototyping

Test cases are at the center of many of the efforts that seek to further software reliability. General correctness can not yet be proven automatically and maybe never will. Even manual proving is only possible for small, select parts of source code.

Using test cases to "reasonably ensure" the correctness of a software component seems to be the most viable option at the moment, though it has pitfalls as well. The coverage of all possible inputs and outcomes is non-trivial and adequate coverage can not be proven easily either.

17.1. Method: Test Cases from Models

The current approach of the automotive industry is tool-assisted test case generation from models combined with extensive simulation early in the development process and rigorous iteration.

Audi (sec 4.2.2), Siemens [9], Bosch [1], Ford [65] and GM [119] are some of the companies that have stated that automatic generation of test cases is being researched and developed. Audi are the only company that indicates that usage is more than research efforts.

Rating

Deployment	★★★★☆	Deployed (Audi), Research, Pilots
Reliability	★★★★☆	Mixture of Engineers and Researchers
Prevention	✓	Testing is prevention
V&V	✓	Testing is V&V
Prediction	✓	Prediction trough V&V

17.2. Method: Rapid Prototyping / Front-loading

The specification of the V-Model strictly follows the modeling (descent), implementation (bottom) and testing (ascent) of the v-shape the model is named after. Artifacts have to be completed before they can be tested. This delays the evaluation of models until after implementation, which makes changes based on that evaluation costly. That high cost compelled companies to find ways of validating models and testing integration before the appropriate software and hardware are available.

This can be done in different ways. One possible approach is using *walking skeletons* (Siemens, sec. 8.2.1) for prototyping architecture. Another is using executable models for validating their behavior in a Model-in-the-Loop (MiL) set-up, which most companies included in the research are using.

Rating

Deployment	★★★★★	Deployed by multiple companies
Reliability	★★★★★	Engineers of multiple companies
Prevention	✓	Testing is prevention
V&V	✓	Testing is V&V
Prediction	✓	Prediction trough V&V

18. Implementation

The source code is written and compiled to a binary in the implementation phase. All checks and tests that rely on the source code are also performed here. Although the name of this phase comes from the act of manually writing source code, this process is often automatically performed by code generators when using MBD.

18.1. Method: Code Generation

This method describes the generation of software component (SWC) code, not code generation for the execution of test cases. There are three basic approaches to implementation in Model Based Development (MBD), two of which are being actively used.

1. Generation of stubs, manual implementation (e.g. ZF)
2. Automatic generation, additional manual editing
3. Automatic generation, no editing (e.g. AUDI, BMW, VW)

Option 2 does not seem to be used. This could be because the large effort of creating detailed models for the generators is inefficient if code can not be re-generated without losing some or all of the work done in editing.

Naturally, more automation is always preferable, but is not trivial to achieve. A high automation level at the code generation step has to come at the cost of more effort that has to be put into the detail level of models used for generation.

Still, automatic generation offers large advantages over manual coding, especially since models are created anyway for the generation of test cases and validation of different artifacts.

Rating

Deployment	★★★★★	Deployed (Audi, BMW, VW)
Reliability	★★★★★	Engineers (Audi, BMW, ...)
Prevention	✓	Prevents coding errors
V&V	✓	Ensures valid code
Prediction	X	No relation

18.2. Method: Static Code Checking

Whenever source code is generated it has the potential of containing errors. Even automatically generated code can be imperfect at times, even if code generators have shown to produce very good quality code in practice [5].

There are multiple kinds of static code checking that offer different benefits. Any modern Integrated Development Environment (IDE) offers built-in checking for syntax errors and other criteria that can be performed in near-real-time. Additionally, off-the-shelf software (MathWorks Polyspace (PS), Bauhaus, Astrée) potentially offers more advanced techniques that may not work in real time but need time to compute.

Rating

Deployment	★★★★★	In use: Eclipse, Visual Studio
Reliability	★★★★★	Engineers
Prevention	✓	Prevents faults
V&V	✓	Ensures correct function
Prediction	X	No relation

18.3. Method: Formal Verification & Validation

Formally verifying source code is a special type of static checking. One approach to this kind of checking is the Memory Interval Constraint Solving (MEMICS) project (Daimler, sec. 6.3.3) which compiles the source code to an alternative target which can be checked by constrained solvers. This allows for a range of checks to be performed. One rather interesting one is the option to check for deadlocks and race conditions - two categories that are very hard to automatically check for with conventional tools like static checkers.

An industrial case study was performed at Siemens by the DEPLOY project [60] which used the RODIN platform [136] for formal validation in the railway domain (sec.8.3.3). The same tool set was also used at Bosch with similar results (9.3.2). There is also research at Audi that uses an approach based on Petri nets to formally verify the causal ordering of test steps (sec. 4.3.4).

The engineers at Ford use a combination of three in-house tools for formal validation (sec. 11.2.2).

Plans for deployment of these formal methods into production is hindered by the same problem independent of company or field. None of the formal verification approaches scales very well with complexity, which can hardly be avoided.

Rating

Deployment	★★☆☆☆	Problems stop deployment plans
Reliability	★★★★☆	Multitude of sources boosts score
Prevention	✓	Prevents faults
V&V	✓	Ensures correct function
Prediction	✓	Proves reliability

19. Component Testing

As soon as a software component can be compiled, it can also be verified by checking it against its specification, models and test cases. This phase in the classical V-Model is the first one that is meant to trigger an iteration if the testing reveals any flaws or shortcomings. If an iteration is triggered this means tracing the part responsible to the artifact that induced its creation and adjusting that artifact. Then the implementation is adjusted accordingly and component testing is re-started.

19.1. Method: Back-to-back Testing

Back-to-back testing is - simply stated - the comparison of the behavior of an artifact with the behavior of its derivative or counterpart. In the case of Electronic Control Unit (ECU) development that is commonly used to indicate the comparison of models to binaries created according to those models.

An example of back-to-back testing is the co-simulation of software and executable models. Every company that is included in this thesis either does go the extra step to make their models able to be executed and simulated or has no information on it available at all.

Rating

Deployment	★★★★★	Part of every known development process
Reliability	★★★★★	May be best documented method
Prevention	✓	Prevents faults
V&V	✓	Ensures correct function
Prediction	✓	Prediction through V&V

19.2. Method: Virtual Integration

Virtual integration is a frontloading technique that aims to perform integration testing before the target hardware is available. To make this possible, that hardware is simulated for the tests. For software component testing this might take the form of actual binary execution on a simulated micro controller (e.g. sec. 6.3.1 (Daimler), 5.2.5 (BMW).

Rating

Deployment	★★★★☆	Deployed by at least 2 major companies
Reliability	★★★★★	Engineering sources in very relevant position
Prevention	✓	Prevents faults
V&V	✓	Ensures correct function
Prediction	✓	Prediction through V&V

19.3. Method: State Space Exploration

State Space Exploration is a testing method that does not rely on test cases or pre-defined procedures. It relies on interface definitions in the form of input (possible) and output (acceptable) signals. The signals are defined with the possible values or ranges and a off-the-shelf software explores combinations and sequences of signals. This method does not only try to

produce error states but also valid states that are close to being critical. This is an advantage and unique value because it allows to detect possible future problems that do not produce failures at the moment of testing.

This method is explicitly used by ZF (sec. 7.1.3) and probably used by Daimler (use same software as ZF, sec. 6.2.5).

Rating

Deployment	★★★★☆	Deployed by at least 1 major company
Reliability	★★★★★	Engineering sources in very relevant position
Prevention	✓	Prevents faults
V&V	✓	Helps validate the state space
Prediction	✓	Prediction through V&V

20. System/Integration Testing

20.1. Method: Multiple Functions on Same ECU

In the future, multiple functions will have to be put on the same ECU out of necessity to control the excessive growth in number of ECUs, their power use and the complexity of the resulting system. The problem with this is that each of those functions could potentially interfere with the execution of the others. Scheduling is already delicate when all functions are of the same Automotive Safety Integrity Level (ASIL) level, but when mixed levels are introduced, it becomes even harder. Functions of different ASIL levels have different priorities [125].

While an entertainment function might want to always use all available resources (best frame rate) possible, a safety-critical function might want to keep a unused resource buffer to protect from delays in execution if resource usage spikes. This is an extreme example (entertainment + safety critical on same ECU) but it serves to illustrate the point.

Engineers from Audi report on a combination of measures to ensure satisfactory execution of all functions hosted on the same ECU even if (and especially when) they have different ASIL levels. They use AUTomotive Open System ARchitecture (AUTOSAR) timing protection with Criticality Aware Priority Assignment (CAPA) scheduling (sec. 4.3.6).

Rating

Deployment	★★★★★	Case studies, deployment plans assumed
Reliability	★★★★★	Engineers in one company
Prevention	✓	Part of testing
V&V	✓	Part of testing
Prediction	✓	Prediction through V&V

20.2. Method: Networked Hardware-in-the-Loop

After the software component (SWC) was tested sufficiently with Software-in-the-Loop (SiL) techniques, it is flashed onto an actual hardware controller and connected to a test bed in a Hardware-in-the-Loop (HiL) set-up. This allows for testing in an environment very close to the target environment in terms of analog disruption factors and interferences like heat, electrical instability, signal interference and others that might affect the hardware directly.

The biggest downside of Hardware-in-the-Loop (HiL) is the inefficiency of testing in real-time. Even if the system is just idling between events it can not be sped up and has to execute in constant time frames.

As far as deployment goes, HiL seems to be the industry baseline and is used in every company. Because of this ubiquity the method was moved up to *Networked* HiL as even that definition still receives the highest score. Networked means that the controller is not simulated by itself but is connected to other components of the vehicle which are either simulated or also connected as a HiL.

Rating

Deployment	☆☆☆☆☆	Used by every major company
Reliability	☆☆☆☆☆	Engineers in very relevant positions
Prevention	✓	Prevents faults from occurring
V&V	✓	Ensures correct function
Prediction	✓	Prediction through V&V

20.3. Method: Environment Simulation

When a component is connected in a Networked HiL that complete system needs to be stimulated with inputs. Those can be pre-crafted sequences or pre-recorded sequences from test drives of a vehicle, but both of these are very inflexible. There are many parameters that might influence the outcome like vehicle weight and size, outside temperature and many others.

Although recorded sequences are helpful as a reference and are in fact being used, it is unrealistic to have every possible driving sequence pre-recorded on file. This is probably the reason that many major companies have developed some kind of environment and vehicle simulator platform. It allows for configurable drive scenarios which automatically generate the appropriate signals so stimulate the HiL, SiL, MiL and any combination of those.

BMW uses their platform CarMaker for this purpose (sec. 5.2.5), Audi and Volkswagen (VW) use their Virtual Test Drive system (sec. 4.2.7, sec. 10.2), ZF uses ZF Friedrichshafen SoftCar (SoftCar) (sec) 7.1.3).

Rating

Deployment	☆☆☆☆☆	Used by at least four major companies
Reliability	☆☆☆☆☆	Engineers in very relevant positions
Prevention	✓	Helps with prevention
V&V	✓	Helps execute validation
Prediction	✓	Prediction through V&V

20.4. Method: Portable Emulation

At the end of the V-Model it is assumed that the component and its hardware were already tested exhaustively and that there is an actual controller that can be built into a test vehicle to be tested in integration tests. This assumption is not necessarily correct, because at least two major companies (BMW, Daimler) have found ways to build portable emulators for virtual integration, which enable partial frontloading of integration tests into earlier development phases.

BMW uses the Virtuelle Absicherungsplattform (VAP) (virtual validation platform) which is a real-time Linux system running on IA-32 (x86) architecture (sec. 5.2.5) and Daimler uses the Virtual Integration and Testing (VIT) platform (sec. 6.3.1) simulation software.

Both of these can be loaded on a mobile PC system and connected to the vehicle network to simulate the component during a physical test drive.

Rating

Deployment	☆☆☆☆☆	Used by at least two major companies
Reliability	☆☆☆☆☆	Engineers in relevant positions
Prevention	✓	Helps with prevention
V&V	✓	Helps execute validation
Prediction	✓	Prediction through V&V

21. Software Monitoring

Merriam-Webster describes monitoring as "to watch, observe, listen to, or check (something) for a special purpose over a period of time" [40]. In the context of software development this means to observe or check the state of the software or parts of it over a period of time. In this chapter it is also extended to include the recording of the results of monitoring - information (or even knowledge).

Monitoring approaches start when there is something to monitor. To avoid having separate chapters for methods in the descending part of the V-Model that are produced by frontloading approaches, the notes about front-loaded methods are included in the appropriate phases in the ascending (right-hand) side of the V-Model.

Useful domain-specific terms are "measurement, which is acquiring data from inside an ECU and calibration, which is writing data to the ECU" [26] (Note: Emphasis added. In German documents the word *applizieren* rather than the direct equivalent *kalibrieren* is used to mean calibration).

21.1. Method: Component Test Monitoring

In the component testing phase (chap. 19) monitoring is done either on the executable component behavior model (frontloading, MiL) or the generated binaries (SiL). Often both ways are treated as interchangeable. Because the SWC is not yet deployed into a hardware device it is possible to step through the execution or set breakpoints in a debugger, which offers a very powerful way to check the execution state. Still, even in this state, treating the artifact as a stand-in for a real hardware device is still important. Not only is it easier to gain overview through a constant stream of information provided

by monitoring, it is also important to evaluate and debug the component's ability to be monitored and react to monitoring requests and techniques appropriately.

ZF reports that their tool SoftCar offers monitoring for models (MiL) and binaries (SiL) on its own but external tools (like TestWeaver) can also be used through an Universal Measurement and Calibration Protocol (XCP) interface (sec. 7.1.3). Audi has commissioned the creation of a connector called INCA SimuLink® Integration Package (INCA-SIP) that allows ETAS GmbH INCA (INCA) to directly integrate with MatLab/SimuLink (ML/SL) and provide the same calibration experience in every phase (sec. 4.2.3). Daimler reports that calibration of a simulated ECU over XCP is implemented (sec. 6.2.4). Without listing all companies it seems very clear that this is a broadly used practice. One thing to note is that all of these are frontloading techniques.

Rating

Deployment	★★★★★	Deployed (Audi, ZF, Daimler and others)
Reliability	★★★★★	Multitude of engineers in different companies
Prevention	✓	Part of testing
V&V	✓	Part of testing
Prediction	✓	Prediction through V&V

21.2. Method: Integration Test Monitoring

Monitoring in the integration-, system- or vehicle test is an established technique for a long time. Recent developments of virtual integration platforms like BMW's Virtuelle Absicherungsplattform (VAP) (sec. 5.2.5) or Daimler's Virtual Integration and Testing (VIT) (sec. 6.3.1) extend the possibilities of component test monitoring (sec. 21.1) to later test phases as well.

Rating

Deployment	★★★★★	Standard technology
Reliability	★★★★★	Multitude of engineers in different companies
Prevention	✓	Part of testing
V&V	✓	Part of testing
Prediction	✓	Prediction through V&V

21.3. Method: Flexible Monitoring with XCP

The Universal Measurement and Calibration Protocol (XCP) protocol offers the ability to inspect values inside the ECU over the Car Area Network (CAN) and can be configured on-the-fly during runtime. Audi has successfully used AUTOSAR with XCP to generate A2L files automatically by using AUTOSAR definitions as inputs for the generator. This allows to select the measured data for each test case dynamically from a list and greatly reduces the required bandwidth.

They have used XCP in other unusual use cases as well. Firstly, they have shown that data can be traced on-the-fly (without interrupting the execution, without significant delay). This allows e.g. for plotting continuous values at interfaces and visually analyzing results. Secondly, they have generated task traces of the ECU's task system without overhead with pre-/post-task handlers. XCP data aquisition (DAQ) events attached to buffers were used to empty and reuse them when they became full, which ran in the idle time of the CPU.

Rating

Deployment	★★★★☆	Case studies, deployment plans assumed
Reliability	★★★★☆	Engineers in one company
Prevention	✓	Part of testing
V&V	✓	Part of testing
Prediction	✓	Prediction through V&V

21.4. Method: Synchronized Data Logging

When monitoring events in a vehicle, which has over 50 ECUs scattered throughout its body, it is non-trivial to synchronize the time across the whole system and supply all log entries with accurate time. One framework that is used for this task is EB Assist Automotive Data and Time-Triggered Framework (ADTF) by Elektrobit Automotive GmbH, at least according to the producer of the software [84].

One company that has confirmed that in their own publications is Audi, who have written about it since 2009 [95] and most recently in 2011 [99]. Audi also reports using the record and replay feature of the software.

Rating

Deployment	★★★★★	Deployed at multiple companies
Reliability	★★★★☆	Engineers in one company + Producer
Prevention	✓	Part of testing
V&V	✓	Part of testing
Prediction	✓	Prediction through V&V

22. Software Maintenance

After a ECU software was released and mass-produced there is still potential to repair that product trough post-release updates or improve components and *functions* used in it for future vehicles.

22.1. Method: Field Data Analysis

There is a constant flow of data from service partners back to the Original Equipment Manufacturers(OEMs) that can be used to improve future versions and generations of the ECU software. Whenever a vehicle is brought in for a repair the collected data of the diagnostic functions read out and sent back for analysis. This results in a data flow of approximately 5 gigabyte per day according to BMW (sec. 5.3.2).

BMW uses that data to improve their software, but also to improve their diagnostic functions, which can be almost half (40%) of the complete software amount in a vehicle. ZF reports the use of operational statistics to assign a reliability value to different components of a system so that future systems built from these parts can be accurately estimated on their reliability (sec. 7.2.1).

Rating

Deployment	☆☆☆☆☆	Used by at least two major companies
Reliability	☆☆☆☆☆	Engineers in relevant departments
Prevention	✓	Helps with prevention
V&V	✓	Helps improve validation
Prediction	✓	Prediction through V&V

22.2. Method: Software Life-cycle Management

After software is released into series production and shipped with the vehicle, any defects become extremely expensive to fix. The traditional way to fix a defective ECU is to physically replace it with a new one. If this is a general defect in all ECUs of that series, you have to bring in all vehicles and change all of them. This is a very expensive way to do things.

Ford estimates that they have avoided expenses of an approximately $100 million by using Product Life-cycle Management (PLM) software Siemens Teamcenter (Teamcenter) that was reinvented/extended for the management of software. Teamcenter manages software dependency tracking, broadcast deployment to service centers over the Internet and many other functions. It has in most cases removed the need for replacement of ECU, because modern ECUs have built-in flash memory and can be flashed over the vehicle network (sec. 11.1.5).

While only Ford has documents available that lay out their process in general, third parties [73] report on over-the-air (OTA) updates at other manufacturers like Audi (Audi connect solution), GM (OnStar embedded connectivity platform), Tesla, Chrysler (firmware over-the-air (FOTA), embedded 3G data connection in the car or a Wi-Fi router), Toyota, Daimler (MBRACE2 [38]) and BMW.

Long time industry software supplier Vector Informatik GmbH (CANape) is joining in and partnering with Red Bend Software to provide OTA updates to ECU firmware. Vector supplies the boot-loader and flashing technology while Red Bend supplies the OTA technology [130].

Rating

Deployment	★★★★☆	Used by at least one company
Reliability	★★★★☆	Engineer at one company, suppliers
Prevention	✓	Helps with prevention
V&V	✓	Helps improve validation
Prediction	✓	Prediction through V&V

23. Tool Overview

	Audi	BMW	Bosch	Daimler	Ford	Siemens	Toyota	VW	ZF
Requirements	DOORS	DOORS	-	DOORS	-	EA	own	-	DOORS
Environment Simulation	MODENA VTD	CarMaker	-	Silver VEOS	-	-	Simscape	-	SoftCar
Component Modeling	ML/SL	ASCET	-	ML/SL	ML/SL	-	ML/SL Stateflow	ML/SL	ML/SL
Behavior Modeling	EXAM MaTeLo	TcEd	-	-	-	EA TeDeSo	-	-	-
Test Management	EXAM	MESSINA	-	TestWeaver TESTUS	-	TestStand ATI	-	-	TestWeaver
Formal Verification	LA PetriNets	-	Event-B	-	Cyclops Hydra Marple	Event-B	MC/DC	-	-
Virtual Integration	-	VAP	-	VIT	-	-	-	-	-

Table 23.1.: Tool Overview

Table 23.1 shows the use of different software products at each company. Sources for the indication are found below:

Audi

- [78]: IBM Rational Dynamic Object Oriented Requirements System (DOORS), Extended Automation Method (EXAM) execution, Berner & Mattner Modena (MODENA); ML/SL [52]
- [94]: Markov Test Logic (MaTeLo), EXAM modeling
- [134]: Logic of Action (LA), Petri Nets
- [21, 94, 52]: MatLab/SimuLink (ML/SL)

BMW

- [97]: DOORS, Advanced Simulation and Control Engineering Tool (ASCET), Test Case Editor (TcEd), Berner & Mattner MESSINA
- [88]: CarMaker (CM)
- [87]: Virtuelle Absicherungsplattform (VAP)

Bosch

- [57, 59, 58]: Event-B, ProB

Daimler

- [110]: DOORS
- [54]: QTronic Silver (Silver), QTronic TestWeaver (TestWeaver)
- [67, 66]: Virtual Integration and Testing, dSPACE VEOS, ML/SL
- [126]: Berner & Mattner TESTUS
- [92]: ML/SL

Ford

- [55]: dSPACE TargetLink (→ ML/SL)

Siemens:

- [128, 129]: Sparx Systems Enterprise Architect (EA), Test Design Studio (TeDeSo), Automatic Testing Infrastructure (ATI), National Instruments TestStand (TestStand)

Toyota:

- [47]: self-developed
- [72]: MatLab/SimuLink (ML/SL)

VW:

- [5]: MatLab/SimuLink (ML/SL)

ZF:

- [102]: DOORS, SoftCar, TestWeaver

Part IV.

Ending Remarks

24. Conclusion

This thesis sought to explore the development of life-critical software in the automotive domain which is not easily accessible due to the sensitive nature of the information for the competitive advantage. It has laid out the overall process that is being used in a series of top companies in the industry together with many of the software tools that help perform that process. It has further identified future directions and challenges for each company and analyzed the reliability of it's sources through the people that they originate from and their affiliations. It has identified common technologies, methods and tools used across different companies and consolidated that information into a generalized structure that allows an insight into software development in the industry as a whole. The abstracted methods were analysed and rated to allow the reader their own evaluation and conclusion but also to ease the digestion of the information and the option to serve as a reference guide.

The main empirical findings are discussed in detail in part III and this section will try to condense the strongest points in short.

1. **Development Model - The V-Model**: This is unanimously the only model that is used, and even Toyota (who only recently re-evaluated their practices) have found it to be the best model to use. There is not even an alternative in any company included that could be mentioned as an alternative (except maybe variations on the V-Model itself).
2. **Requirements Management with DOORS**: Every company seems to agree that the importance of requirements management can not be overstated. The most used tool for this is overwhelmingly DOORS with notable exceptions of Siemens (uses EA) and Toyota (own tool, in development).

3. **Frontloading**: The clear step for the future is the integration of testing efforts directly into the development of models at every step starting at the earliest possible point. Executable models in multiple abstraction levels, visualization and abstraction of hardware and seamless transition between models and code are only some of the tools that support this development.
4. **Automatic Generation of Everything**: Pure C source code is being reduced to a kind of transfer medium between models and binaries. Most companies are either already (completely) automatically generating the source code or are on their way. Executable tests (test cases) are seemingly moving in the same direction.
5. **Automatic Execution of Tests**: With formal definition of everything it is a logical step to seek the automation of testing at every step. With availability of models for every step of the way this becomes a reasonable goal to pursue.

Taking all of the findings into account, it seems that the automotive industry (and presumably other life-critical domains) are actually moving away from traditional software *development* altogether and into something more of a software *modeling* way of doing things. This different approach might not seem very familiar to the average software developer of today and may very well be looked at separately from the rest of software development efforts. Traditional software development can not reasonably be divorced from source code altogether, but the automotive industry is trying to do exactly that. The traditional effort is just not reliable enough.

Future work in this area might focus on furthering the insight into the actual practice in everyday development, assessing the objective degree of success that the described methods can offer and evaluate quantifiable improvements that MBD can offer above traditional methods and its pitfalls. There is also a potential to compare the different software tools used on a detailed scale and give recommendations on methods and tooling for companies that aspire to replicate or even top the success of the biggest companies in the field.

The findings in this work should still be taken with a grain of salt, for all of its findings (despite the high level of consistency across companies) are based on approved publications of the companies themselves, with

supplemental information of third parties. Unfortunately, it is not possible to assert with conviction and accuracy, what the actual situation and practice is in the daily development work at the companies mentioned.

Model Based Development has its drawbacks like high initial costs, slow development speed and additional expenses for re-training of expert personnel. The high complexity, high bug count and bad manageability of a 100 million LOC code-base seems to tip the scale in favor of MBD anyway.

Overall the Model Based Development approach presents itself as a viable, even preferable method to ensure safety and reliability of life-critical systems in the automotive domain and similar fields.

List of Terms

ACM Association for Computer Machinery. 12
ADL Architecture Description Language. 86, 87, 125
ADTF Automotive Data and Time-Triggered Framework. 114
AEV Audi Electronics Venture GmbH. 18
AIT Austrian Institute of Technology. 75
ARXML AUTOSAR XML. 32, 33
ASCET Advanced Simulation and Control Engineering Tool. 32, 33, 37, 87, 118
ASIL Automotive Safety Integrity Level. 26, 27, 107
ASM Automotive Simulation Models. 73
ATI Automatic Testing Infrastructure. 57, 118, 119
AutomationDesk dSPACE AutomationDesk. 73
AUTOSAR AUTomotive Open System ARchitecture. 9, 25, 27, 32, 33, 35, 38, 40, 47, 62, 63, 86, 107, 113
AVL List Anstalt für Verbrennungskraftmaschinen List GmbH. 86, 97

back-to-back testing Comparison of the behavior of an artifact with the behavior of its derivative or counterpart. 73, 104
Bauhaus The Bauhaus Toolkit. 64
Big Design Up-Front A process in which the majority of planning and design effort happens in the first phase of the process. 93, 96
BMW Bayrische Motorenwerke AG. viii, 29–33, 36, 37
bottom-up testing A design approach in which small parts (bottom of hierarchy) is tested first, then tested together after each of the parts has passed.. 8
BRMS Business Rule Management System. 26
BSW base software. 33, 35, 40, 63, 67–69

C The C Programming Language. 32, 69, 94, 122

C++ The C++ Programming Language. 49
calibration writing data to an ECU. 111
CAN Car Area Network. 41, 51, 113
CANape Vector Informatik CANape. 42, 51
CAPA Criticality Aware Priority Assignment. 27, 107
CM CarMaker. 36, 109, 118
CTE Classification Tree Editor. 73
CTM/ES Classification Tree Method. 40

DAQ data aquisition. 26, 113
DEPLOY Industrial deployment of advanced system engineering methods for high productivity and dependability. 59, 103
DOORS IBM Rational Dynamic Object Oriented Requirements System. 19, 20, 23, 30, 39, 45, 48, 49, 95, 118, 119, 121

EA Sparx Systems Enterprise Architect. 56, 95, 118, 119, 121
EAST Electronics Architecture and Software Technology. 87, 125
EAST-ADL Electronics Architecture and Software Technology - Architecture Description Language. 86, 87, 97
ECU Electronic Control Unit. 20, 22, 25–27, 30, 33, 35–37, 40–43, 50, 51, 63, 69, 73, 74, 104, 107, 112–116, 125, 126
EE Electrical Engineering. 18
EMF Eclipse Modelling Framework. 49
ET Engineering Technology. 18
ETAS Engineering Tools, Application and Services. 32
EU European Union. 75
Event-B Formal Method. An evolution of the B-method (also known as classical B). Supported by the Rodin Tool Suite. 118, 119
EXAM Extended Automation Method. 20, 22, 23, 50, 118

Ford Ford Motor Company. 71–76
FOTA firmware over-the-air. 116
frontloading Shifting the application of methods to an earlier stage in a development process. 20, 35, 72, 105, 110–112
FSM finite state machine. 32

GENIVI Geneva In-Car Infortainment. 33
GM General Motors. 99

GRAE Group Research and Advanced Engineering. 38

HAV Hierarchical Accumulative Validation. 82
HiL Hardware-in-the-Loop. 18, 19, 22, 24, 42, 70, 71, 73, 108, 109

IABG Industrieanlagen-Betriebsgesellschaft mbH. 7
IDE Integrated Development Environment. 102
IEEE Institute of Electrical and Electronics Engineers. 91
iMBT Integrated Model Based Testing. 57, 58
INCA ETAS GmbH INCA. 20, 51, 112
INCA-SIP INCA SimuLink® Integration Package. 20, 21, 112
interface A group of signals with the description of acceptable types, ranges and effects that can be used to communicate in a predictable way. 32
IR Intermediate Representation. 46
IT Information Technology. 29
ITF Integrated Testing Framework. 23, 70

JRC-ACM Joint Research Committee - Automotive Control And Modelling. 80
JSAE Society of Automotive Engineers of Japan. 80

KIT Karlsruhe Institute of Technology. 63

LA Logic of Action. 26, 118
LKS Lane Keeping System. 73
LLVM Low Level Virtual Machine. 46
LOC lines of code. 2, 123

MARTE Modeling and Analysis of Real-Time and Embedded Systems. 86
MaTeLo Markov Test Logic. 19, 20, 22, 23, 118
MBD Model Based Development. 3, 6, 19, 20, 32, 38, 71, 73, 80, 90, 91, 96, 97, 101, 122, 123
MBFS Model-Based Function Specification. 20, 97
MBT Model Based Testing. 19, 40, 83
MC/DC Modified Condition/Decision Coverage. 82, 83, 118
measurement reading data from the an ECU. 111
MEMICS Memory Interval Constraint Solving. 46, 103
MESSINA Berner & Mattner MESSINA. 33, 73, 118
MiL Model-in-the-Loop. 19, 20, 72, 73, 83, 100, 109, 111, 112

MISRA-C Motor Industry Software Reliability Association - Guidelines for the use of the C language in critical systems. 79
ML/SL MatLab/SimuLink. 19, 20, 40, 67, 69, 72, 83, 112, 118, 119
MODENA Berner & Mattner Modena. 22, 118
Mogentes Model-based generation of tests for embedded systems. 75
MVP minimum viable product. 56

NLP Natural Language Processing. 45, 49, 62

OCR Optical character recognition. 13
OEM Original Equipment Manufacturer. 7, 40, 48, 115
OTA over-the-air. 116

PDF Portable Document Format. 13
Petri net A kind of bipartite directed graph. 26
PLM Product Life-cycle Management. 74, 116
ProB A model checker and constraint solver. 59, 118, 119
PROMELA Process/Protocol Meta Language. 76, 128
PS MathWorks Polyspace. 102

ReCaRe **Re**view with **Ca**tegorized **Re**quirements. 45
rich client An application which includes all execution logic in its own files and because of that can operate off-line. 23
RODIN Rigorous Open Development Environment for Complex Systems. 59, 103
RPC Remote Procedure Call. 32
RTE Run-Time Environment. 33, 35
RTOS Real-Time Operating System. 31, 33
RUD rough upfront design. 55, 56
runnable interface to internal behavior of a component. 32

SAE Society of Automotive Engineers. 82
SICE Society of Instrument and Control Engineers. 80
SiL Software-in-the-Loop. 19, 22, 24, 40, 42, 72, 80, 83, 108, 109, 111, 112
Silver QTronic Silver. viii, 41–44, 118, 119
Simscape MathWorks Simscape. 83, 118
SoftCar ZF Friedrichshafen SoftCar. viii, 50–52, 109, 112, 118, 119
spaghetti code Incomprehensible source code that is hard to read and/or has a high degree of unnecessary dependencies. 78

SPIN Simple Process/Protocol Meta Language (PROMELA) Interpreter. 76
SSA Static Single Assignment. 46
Stateflow MathWorks Stateflow. 83, 118
State-of-the-Art The highest level of general development achieved at a particular time. 4
SuT system under test. 34, 40, 98
SWC software component. 33, 35, 39, 81, 101, 108, 111
SWIFI Software-implemented Fault Injection. 25
SysML Systems Modelling Language. 58, 86
SystemC The SystemC Event Library. 58
SystemDesk dSPACE SystemDesk. 44

TargetLink dSPACE TargetLink. 44, 72, 119
TcEd Test Case Editor. 33, 118
TCP/IP Transport Control Protocol / Internet Protocol. 51
TDL Timing Definition Language. 81
Teamcenter Siemens Teamcenter. 74, 116
TeDeSo Test Design Studio. 56, 57, 118, 119
test bed An installation connecting multiple hardware parts that is used for testing a prototype. 22
test oracle A means of automatically deciding if a test has succeeded or failed. 20
TestStand National Instruments TestStand. 57, 118, 119
TESTUS Berner & Mattner TESTUS. 40, 118, 119
TestWeaver QTronic TestWeaver. viii, 40, 41, 43, 49, 50, 52, 118, 119
top-down design A design approach in which big structures (top of hierarchy) is designed first, then split up into components to add detail.. 8, 96
TPT Time Partition Testing. 40
TUM Technische Universität München. 58

UA Unintended Acceleration. 78
UID Unique ID. 49, 95
UML Unified Modelling Language. 56, 58, 75, 76, 86, 87, 97
UPPAAL-PORT UPPAAL Partial Order Reduction Techniques. 87
URML Unified Requirements Modeling Language. 58

VAP Virtuelle Absicherungsplattform. 9, 29, 33–37, 110, 112, 118

VEOS dSPACE VEOS. 44, 118, 119
VIN Vehicle Identification Number. 74
VIT Virtual Integration and Testing. viii, 43, 44, 110, 112, 118, 119
ViTAL **V**erification **T**ool for EAST-**A**D**L** Models using UPPAAL-PORT. 87
V-Model A software development model used by many car manufacturers, especially in Europe. viii, 3–8, 18, 19, 30, 33, 34, 38, 39, 48–50, 61, 62, 66, 67, 71, 82, 89, 90, 93, 96, 100, 104, 110, 111, 121
VS Microsoft Visual Studio. 83
VSA Vehicle Systems Architect. 32, 35
VSB Vehicle Systems Builder. 35
VTD Virtual Test Drive. 23, 70, 109, 118
VW Volkswagen. viii, 23, 66–70, 109

Waterfall Model A software development model that prescribes one-directional progress through its phases. 93
WCET Worst Case Execution Time. 63

x86 IA-32. 110
XCP Universal Measurement and Calibration Protocol. 25, 41, 51, 112, 113
XMI Extensible Markup Language (XML) Metadata Interchange. 76
XML Extensible Markup Language. 49, 76, 129

ZF ZF Friedrichshafen AG. 7, 48, 50–53, 61, 95, 98, 101, 106, 109, 112, 115

Bibliography

[1] Chris Ackermann et al. "Automatic requirement extraction from test cases." In: *Runtime Verification*. Springer. 2010, pp. 1–15 (cit. on pp. 2, 61, 64, 65, 94, 99).

[2] Dirk Ahrens et al. "Objective evaluation of software architectures in driver assistance systems." In: *Computer Science-Research and Development* 28.1 (2013), pp. 23–43 (cit. on p. 29).

[3] Hanna Amlinger. "Application of a new software tool for the automated test of automotive electronic control unit software." PhD thesis. Stockholm, 2009 (cit. on p. 18).

[4] Saoussen Anssi et al. "AUTOSAR vs. MARTE for enabling timing analysis of automotive applications." In: *SDL 2011: Integrating System and Software Modeling*. Springer, 2012, pp. 262–275 (cit. on p. 86).

[5] Ing Andrea Arenz and Dipl-Ing FH Sven Potrykus. "Model-based algorithm development." In: *ATZ worldwide* 112.1 (2010), pp. 18–22 (cit. on pp. 66–70, 102, 119).

[6] EAST-ADL Association. *EAST-ADL Website*. 2014. URL: http://www.east-adl.info/ (cit. on p. 86).

[7] AUDI AG. *Annual Report for 2013* (cit. on p. 18).

[8] AUDI AG. *AUDI Website - Audi at a glance*. 2014. URL: http://www.audi.com/content/audi_com/corporate/en/company/audi-at-a-glance.html (cit. on p. 18).

[9] Alberto Avritzer et al. "Automated generation of test cases using a performability model." In: *Software, IET* 5.2 (2011), pp. 113–119 (cit. on pp. 59, 60, 99).

[10] Dipl-Ing Alexander Banerjee, Ing Maik Würthner, and Dipl-Ing Cosmin Tudosie. "Prevision GPS—Die Schaltstrategie für das Getriebesystem Traxon." In: *ATZ-Automobiltechnische Zeitschrift* 115.6 (2013), pp. 490–493 (cit. on p. 54).

[11] Michael Barr. *BOOKOUT V. TOYOTA - 2005 Camry L4 Software Analysis*. Expert wittness presentation in Toyota case. 2013 (cit. on pp. 78, 79).

[12] Lothar Beller and Pavel Turjanica. *Standardization and Tool Chains for an Independent Test Center with Different Customer Requirements*. 2010 (cit. on pp. 18, 48–50, 53, 54, 95).

[13] Brian Berenbach. "A 25 year retrospective on model-driven requirements engineering." In: *MoDRE*. 2012, pp. 87–91 (cit. on pp. 58, 60).

[14] Brian Berenbach, Florian Schneider, and Helmut Naughton. "The use of a requirements modeling language for industrial applications." In: *Requirements Engineering Conference (RE), 2012 20th IEEE International*. IEEE. 2012, pp. 285–290 (cit. on pp. 58, 60, 95).

[15] R Bergmann and R Walesch. *HIL-Strategie Audi*. Jan. 2012 (cit. on pp. 22, 23).

[16] Berner & Mattner Systemtechnik GmbH. *Safe with TESTUS*. 2011 (cit. on p. 47).

[17] Abian Blome et al. "VERA: A flexible model-based vulnerability testing tool." In: *Software Testing, Verification and Validation (ICST), 2013 IEEE Sixth International Conference on*. IEEE. 2013, pp. 471–478 (cit. on p. 60).

[18] BMW AG. *Annual Report for 2013*. Mar. 2014 (cit. on p. 29).

[19] BMW AG. *BMW Website - Locations*. 2014. URL: http://www.bmwgroup.com/bmwgroup_prod/e/0_0_www_bmwgroup_com/unternehmen/unternehmensprofil/standorte/standorte/index.html (cit. on p. 29).

[20] Bosch GmbH. *Bosch Website - Bosch in figures*. 2014. URL: http://www.bosch.com/en/com/bosch_group/bosch_figures/bosch-figures.php (cit. on p. 61).

[21] Valerie Bouquet. *Application of the Model-Based Testing approach with MaTeLo in test industrialization at AUDI.* http://www.all4tec.net/index.php/en/news/18
use-case-audi-uses-matelo-for-test-industrialization. June 2013 (cit. on pp. 19, 20, 118).

[22] Ekaterina Boutkova. "Experience with variability management in requirement specifications." In: *Software Product Line Conference (SPLC), 2011 15th International*. IEEE. 2011, pp. 303–312 (cit. on p. 95).

[23] Frank Buschmann. "A Week in the Life of an Architect." In: *Software, IEEE* 29.3 (2012), pp. 94–96 (cit. on pp. 55, 56, 59, 97).

[24] Frank Buschmann. "Tests: The Architect's Best Friend." In: *Software, IEEE* 28.3 (2011), pp. 7–9 (cit. on pp. 55, 56, 59, 97).

[25] Michael Butler, Laurent Voisin, and Thomas Muller. "Tooling." In: *Industrial Deployment of System Engineering Methods*. Springer, 2013, pp. 157–185 (cit. on p. 59).

[26] Philipp Caliebe, Christoph Lauer, and Reinhard German. "Flexible integration testing of automotive ECUs by combining AUTOSAR and XCP." In: *Computer Applications and Industrial Electronics (ICCAIE), 2011 IEEE International Conference on*. IEEE. 2011, pp. 67–72 (cit. on pp. 25, 26, 111).

[27] Robert N. Charette. *This Car Runs on Code*. 2009. URL: http://spectrum.ieee.org/transportation/systems/this-car-runs-on-code (cit. on p. 2).

[28] DeJiu Chen et al. "An architectural approach to the analysis, verification and validation of software intensive embedded systems." In: *Computing* (2013), pp. 1–40 (cit. on p. 86).

[29] D Chen et al. "Integrated safety and architecture modeling for automotive embedded systems*." In: *e & i Elektrotechnik und Informationstechnik* 128.6 (2011), pp. 196–202 (cit. on p. 86).

[30] Ford Motor Company. *Delivering Profitable Growth for All - 2013 Annual Report*. 2014 (cit. on p. 71).

[31] Association for Computing Machinery. *The ACM Digital Library*. 2014. URL: http://dl.acm.org/ (cit. on p. 7).

[32] Toyota Motor Corporation. *Annual Report 2013 - True Competitiveness for Sustainable Growth*. 2013 (cit. on p. 78).

[33] Inc. Coverity. *Coverity Scan: 2013 Open Source Report*. 2013. URL: http://softwareintegrity.coverity.com/rs/coverity/images/2013-Coverity-Scan-Report.pdf (cit. on p. 2).

[34] Cran Communications Inc. "Supplement: Top Suppliers." In: *Automotive News* (June 2013) (cit. on pp. 48, 61).

[35] Othon Crelier, William M Hasling, Christof J Budnik, et al. "Design principles for integration of model-driven quality assurance tools." In: *Software Components, Architectures and Reuse (SBCARS), 2011 Fifth Brazilian Symposium on*. IEEE. 2011, pp. 100–109 (cit. on p. 95).

[36] Philippe Cuenot et al. "11 The EAST-ADL Architecture Description Language for Automotive Embedded Software." In: *Model-Based Engineering of Embedded Real-Time Systems*. Springer, 2011, pp. 297–307 (cit. on p. 86).

[37] Daimler AG. *Daimler at a glance - financial year 2013*. 2014 (cit. on pp. 38, 78).

[38] Daimler AG. *Mercedes MBRACE 2 FAQ*. 2012 (cit. on p. 116).

[39] Dipl-Ing Markus Deicke, Wolfram Hardt, and Ing Marcus Martinus. "Simulation Hardwarespezifischer Komponenten von ECU-Software in der Virtuellen Absicherung." In: *ATZelektronik* 7.3 (2012), pp. 226–231 (cit. on p. 37).

[40] Merriam-Webster Dictionary. *Monitoring - Definition and More*. 2014. URL: http://www.merriam-webster.com/dictionary/monitoring (cit. on p. 111).

[41] Merriam-Webster Dictionary. *Predict - Definition and More*. 2013. URL: http://www.merriam-webster.com/dictionary/predict (cit. on p. 91).

[42] Merriam-Webster Dictionary. *Prevention - Definition and More*. 2014. URL: http://www.merriam-webster.com/dictionary/prevention (cit. on p. 90).

[43] dSPACE GmbH. *dSPACE Website - Software Operating System Compatibility*. URL: https://www.dspace.com/en/pub/home/support/supvers/supverscompm/release_roadmap.cfm (cit. on p. 44).

[44] Institute of Electrical and Electronics Engineers. *IEEE Xplore Digital Library*. 2014. URL: http://ieeexplore.ieee.org (cit. on p. 7).

[45] Eduard Paul Enoiu et al. "Vital: A verification tool for east-adl models using uppaal port." In: *Engineering of Complex Computer Systems (ICECCS), 2012 17th International Conference on*. IEEE. 2012, pp. 328–337 (cit. on p. 87).

[46] ETAS GmbH. *ETAS GmbH INCA-SIP*. URL: http://www.etas.com/en/products/inca_sip.php (cit. on p. 20).

[47] Jared Farnsworth et al. "Hierarchical Accumulative Validation of Executable Control Specifications." In: *SAE International Journal of Passenger Cars-Electronic and Electrical Systems* 6.1 (2013), pp. 186–193 (cit. on pp. 82–85, 119).

[48] Dogan Fennibay, Arda Yurdakul, and Alper Sen. "A Heterogeneous Simulation and Modeling Framework for Automation Systems." In: *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on* 31.11 (2012), pp. 1642–1655 (cit. on p. 59).

[49] Matthias Freier and Jian-Jia Chen. "Prioritization for real-time embedded systems on dual-core platforms by exploiting the typical-and worst-case execution times." In: *Industrial Embedded Systems (SIES), 2018 8th IEEE International Symposium on*. IEEE. 2013, pp. 21–29 (cit. on pp. 61, 63, 65).

[50] Ing Jörn Freyer et al. "Eine Spur Aufmerksamer der Audi Active Lane Assist." In: *ATZ-Automobiltechnische Zeitschrift* 112.12 (2010), pp. 926–930 (cit. on p. 18).

[51] ZF Friedrichshafen. *ZF at a Glance*. 2013 (cit. on p. 48).

[52] Dipl-Ing FH Johann Gabler et al. "ECU function development." In: *ATZelektronik worldwide* 5.5 (2010), pp. 36–39 (cit. on pp. 18, 20, 21, 27, 97, 118).

[53] Johann Gabler et al. *Function Development Without a Vehicle*. http://www.etas.com/data/RealTimes_2010/rt_2010_2_6_en.pdf, 2010 (cit. on p. 27).

[54] Stefan Gloss, Milan Slezák, and Andreas Patzer. "Virtualisation - Validation of Over 200 Transmission Variants on PC." In: *ATZelektronik* 8.4 (2013), pp. 290–294 (cit. on pp. 40–43, 47, 98, 119).

[55] dSPACE GmbH. "Fusion Hybrid Energized." In: *dSPACE Magazine* 3 (Mar. 2010) (cit. on pp. 72, 76, 119).

[56] dSPACE GmbH. "Go for Quality." In: *dSPACE Magazine* 3 (Mar. 2010) (cit. on pp. 73, 76).

[57] Rainer Gmehlich and Cliff Jones. "Experience of Deployment in the Automotive Industry." In: *Industrial Deployment of System Engineering Methods*. Springer, 2013, pp. 13–26 (cit. on pp. 61, 63, 65, 119).

[58] Rainer Gmehlich et al. "On fitting a formal method into practice." In: *Formal Methods and Software Engineering*. Springer, 2011, pp. 195–210 (cit. on pp. 61, 63, 65, 119).

[59] Rainer Gmehlich et al. *Towards a Formalism-Based Toolkit for Automotive Applications*. Tech. rep. Computing Science, Newcastle University, 2012 (cit. on pp. 61, 63, 65, 119).

[60] *Grant agreement for Industrial deployment of advanced system engineering methods for high productivity and dependability (DEPLOY) - Annex I* (cit. on p. 103).

[61] Object Management Group. *MARTE Website*. 2014. URL: http://www.omgmarte.org/ (cit. on p. 86).

[62] Object Modelling Group. *SysML Website*. 2014. URL: http://www.omgsysml.org/ (cit. on p. 86).

[63] VolksWagen Group. *Annual Report for 2013*. 2014 (cit. on pp. 66, 78).

[64] The Guardian. *Carmaker BMW keeps luxury top spot with record 2013 sales*. Jan. 2014. URL: http://www.theguardian.com/business/2014/jan/13/bmw-luxury-top-spot-record-2013-car-sales (cit. on pp. 18, 29, 38).

[65] Dipl-Ing Dirk Gunia and Dipl-Ing FH Jürgen Schüling. "Model-based testing of Ford's lane keeping system." In: *Auto Tech Review* 1.6 (2012), pp. 48–51 (cit. on pp. 73, 76, 99).

[66] Artur Honisch et al. "Virtual Integration and Testing of Vehicle E/E Systems." In: *ATZelektronik worldwide* 8.5 (2013), pp. 22–25 (cit. on pp. 44, 47, 119).

[67] Artur Honisch and Karsten Krügel. "Virtuelle Integration und Test von E/E-Fahrzeugsystemen." In: *ATZelektronik* 8.5 (2013), pp. 350–355 (cit. on pp. 43, 44, 47, 119).

[68] IABG. *V-Modell XT*. 1.4. Industrieanlagen-Betriebsgesellschaft GmbH. May 2012 (cit. on pp. 6, 18).

[69] IABG. *V-Modell-Seiten der IABG*. Industrieanlagen-Betriebsgesellschaft GmbH. 2014. URL: http://www.v-modell.iabg.de/ (cit. on pp. 3, 6, 7).

[70] "IEEE Guide–Adoption of the Project Management Institute (PMI(R)) Standard A Guide to the Project Management Body of Knowledge (PMBOK(R) Guide)–Fourth Edition." In: *IEEE Std 1490-2011* (Nov. 2011), pp. 1–508. DOI: 10.1109/IEEESTD.2011.6086685 (cit. on p. 91).

[71] Siemens Product Lifecycle Management Software Inc. *Ford Motor Company £100+ million in warranty cost savings*. URL: http://www.plm.automation.siemens.com/en_us/about_us/success/case_study.cfm?Component=63184&ComponentTemplate=1481 (cit. on pp. 74, 77).

[72] The Mathworks Inc. *Toyota Front-Loads Development of Engine Control Systems Using Comprehensive Engine Models and SIL+M*. 2014 (cit. on pp. 83, 85, 119).

[73] Krishna Jayaraman. *Over-the-Air Updates to Slash Automobiles' Recall Rates*. 2013. URL: http://www.frost.com/prod/servlet/press-release.pag?docid=284456381 (cit. on p. 116).

[74] Adam C Jensen et al. "A toolchain for the detection of structural and behavioral latent system properties." In: *Model Driven Engineering Languages and Systems*. Springer, 2011, pp. 683–698 (cit. on p. 76).

[75] Andreas Junghanns, Jakob Mauss, and Michael Seibt. "Faster Development of AUTOSAR compliant ECUs through simulation." In: *ERTS-2014, Toulouse* () (cit. on p. 42).

[76] Eun-Young Kang, Pierre-Yves Schobbens, and Paul Pettersson. "Verifying functional behaviors of automotive products in EAST-ADL2 using UPPAAL-PORT." In: *Computer Safety, Reliability, and Security*. Springer, 2011, pp. 243–256 (cit. on p. 87).

[77] Steffen Keul. "Tuning Static Data Race Analysis for Automotive Control Software." In: *Source Code Analysis and Manipulation (SCAM), 2011 11th IEEE International Working Conference on*. IEEE. 2011, pp. 45–54 (cit. on p. 64).

[78] Gerhard Kiffe et al. "Benutzungsorientiertes und modellzentriertes Testen im HiL-Testing." In: *ATZelektronik* 4.5 (2009), pp. 56–60 (cit. on pp. 19, 22–24, 27, 70, 95, 118).

[79] Jens Kohl et al. "Using multivariate split analysis for an improved maintenance of automotive diagnosis functions." In: *Software Maintenance and Reengineering (CSMR), 2011 15th European Conference on*. IEEE. 2011, pp. 305–308 (cit. on p. 37).

[80] Martin Krammer, Eric Armengaud, and Quentin Bourrouilh. "Method library framework for safety standard compliant process tailoring." In: *Software Engineering and Advanced Applications (SEAA), 2011 37th EUROMICRO Conference on*. IEEE. 2011, pp. 302–305 (cit. on p. 97).

[81] Jorg Leuser. "Challenges for semi-automatic trace recovery in the automotive domain." In: *Proceedings of the 2009 ICSE Workshop on Traceability in Emerging Forms of Software Engineering*. IEEE Computer Society. 2009, pp. 31–35 (cit. on p. 95).

[82] Matthias Lindner and Thomas Tille. "Design of highly integrated mechatronic gear selector levers for automotive shift-by-wire systems." In: *Mechatronics, IEEE/ASME Transactions on* 15.6 (2010), pp. 961–968 (cit. on p. 29).

[83] Quantisle Ltd. *Qiqqa Website*. 2014. URL: http://www.qiqqa.com (cit. on p. 13).

[84] Jürgen Ludwig. "Elektronischer Horizont Vorausschauende systeme und deren Anbindung an NavigatioNseinheiten." In: *ATZelektronik* 7.6 (2012), pp. 434–439 (cit. on p. 114).

[85] CR Maga and N Jazdi. "Interdisciplinary modularization in product line engineering: A case study." In: *Automation Quality and Testing Robotics (AQTR), 2012 IEEE International Conference on*. IEEE. 2012, pp. 179–184 (cit. on p. 86).

[86] Managemant Circle AG. *Lean Product Development 2013 Interaktive Konferenz*. http://www.leanmagazin.de/medien/publikationen/doc_download/476-lean-product-development-2013.html. Found through a search engine. 2013 (cit. on p. 54).

[87] Ing Marcus Martinus, Zoran Cutura, and Dipl Ing FH Thomas Würz. "Virtuelle Absicherungs-Plattform Integration und Wiederverwendung von Software." In: *ATZelektronik* 7.1 (2012), pp. 56–61 (cit. on pp. 33, 35–37, 118).

[88] Ing Marcus Martinus, Dipl-Ing Markus Deicke, and Dipl-Ing Michael Folie. "Virtueller Fahrversuch Hardwareunabhängige Integration von Seriensoftware." In: *ATZelektronik* 8.5 (2013), pp. 344–349 (cit. on pp. 36, 37, 118).

[89] Jakob Mauss. "Chip simulation used to run automotive software on PC." In: *ERTS-2014, Toulouse* (2014), pp. 05–07 (cit. on p. 41).

[90] Jakob Mauss and Matthias Simons. "Steuergeräte-Simulation auf PC mittels TriCore-Emulation." In: *ATZelektronik* 7.6 (2012), pp. 460–465 (cit. on pp. 41, 47).

[91] Torben Meyer, Carsten Pöge, and Gottfried Mayer. "Integration of emulation functionality into an established simulation object library." In: *Proceedings of the Winter Simulation Conference*. Winter Simulation Conference. 2012, p. 253 (cit. on p. 29).

[92] Alexander Michailidis et al. "Test front loading in early stages of automotive software development based on AUTOSAR." In: *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2010*. IEEE. 2010, pp. 435–440 (cit. on pp. 38–40, 47, 119).

[93] Dipl-Inform Alexander Michailidis, Ing Thomas Ringler, and Ing Stefan Kowalewski. "Virtuelle Integration Modellbasierter Fahrzeugfunktionen unter Autosar." In: *ATZelektronik* 5.1 (2010), pp. 32–37 (cit. on pp. 38, 40, 47).

[94] MicroNova AG. *Exam und MaTeLo: von den Requirements zum Testfall*. 2012. URL: http://www.exam-ta.de/images/stories/exam/Flyer_EXAM_MaTeLo_1v0_digital.pdf (cit. on pp. 20, 22, 118).

[95] Ing Maximilian Miegler et al. "Hardware-in-the-Loop-Test von vorausschauenden Fahrerassistenzsystemen." In: *ATZelektronik* 4.5 (2009), pp. 14–19 (cit. on pp. 28, 70, 114).

[96] Dipl-Ing FH Alexander Moiszi and M Sc Michael Tybel. "Gap in the V-model of E-drives Closed by Parameter Identification." In: *ATZelektronik worldwide* 8.5 (2013), pp. 36–39 (cit. on pp. 62, 65).

[97] Dieter Nazareth and Robert Siwy. "Development of an AUTOSAR Software Component Based on the V-Model." In: *Proceedings of the FISITA 2012 World Automotive Congress*. Springer. Beijing, China: Springer Berlin Heidelberg, 2013, pp. 407–416 (cit. on pp. 18, 30–34, 36, 37, 87, 93, 95, 118).

[98] Edward Nelson and Henry Huang. "A Software and System Modeling Facility for Vehicle Environment Interactions." In: *Model-Driven Development of Reliable Automotive Services*. Springer, 2008, pp. 34–47 (cit. on p. 86).

[99] Dipl-Ing FH Mirko Nentwig, Dipl-Ing Reinhard Schieber, and Ing Maximilian Miegler. "Hardware-in-the-Loop-Test für Vernetzte Fahrerassistenz Systeme." In: *ATZelektronik* 6.4 (2011), pp. 20–25 (cit. on pp. 23, 28, 114).

[100] Mirko Nentwig and Marc Stamminger. "Hardware-in-the-loop testing of computer vision based driver assistance systems." In: *Intelligent Vehicles Symposium (IV), 2011 IEEE*. IEEE. 2011, pp. 339–344 (cit. on pp. 23, 28).

[101] Florian Netter, Frank Gauterin, and Bjorn Butterer. "Real-Data Validation of Simulation Models in a Function-Based Modular Framework." In: *Software Testing, Verification and Validation (ICST), 2013 IEEE Sixth International Conference on*. IEEE. 2013, pp. 41–47 (cit. on pp. 18, 24, 28).

[102] Martin Neumann et al. *Absicherung von Steuerungssoftware für Hybridsysteme*. 2011 (cit. on pp. 49–52, 54, 95, 98, 119).

[103] Martin Neumann et al. *Absicherung von Steuerungssoftware für Hybridsysteme (Slides)* (cit. on p. 54).

[104] Dirk Nowotka and Johannes Traub. "Formal Verification of Concurrent Embedded Software." In: *Embedded Systems: Design, Analysis and Verification*. Springer, 2013, pp. 218–227 (cit. on pp. 46, 47).

[105] Dirk Nowotka and Johannes Traub. "MEMICS - Memory Interval Constraint Solving of (concurrent) Machine Code." In: (2012) (cit. on pp. 46, 47).

[106] Akira Ohata. "Benchmark problem for nonlinear identification of automotive engine." In: *Intelligent Control and Automation (WCICA), 2012 10th World Congress on*. IEEE. 2012, pp. 3305–3310 (cit. on pp. 80, 85).

[107] Akira Ohata. *Systems Assurance and Behavior Modeling: Requirements for OMG*. URL: http://sysa.omg.org/docs/Sep10/OMG_Sys_assurance100922.pdf (cit. on pp. 80, 85).

[108] Nouha Omrane et al. "Lexicalized ontology for a business rules management platform: An automotive use case." In: *Rule-Based Modeling and Computing on the Semantic Web*. Springer, 2011, pp. 179–192 (cit. on pp. 26, 28, 94).

[109] Leon Osborne et al. *V-Model*. Licensed under Public domain via Wikimedia Commons. 2005. URL: http://commons.wikimedia.org/wiki/File:Systems_Engineering_Process_II.svg#mediaviewer/File:Systems_Engineering_Process_II.svg (cit. on p. 8).

[110] Daniel Ott. "Automatic requirement categorization of large natural language specifications at mercedes-benz for review improvements." In: *Requirements Engineering: Foundation for Software Quality*. Springer, 2013, pp. 50–64 (cit. on pp. 39, 44, 45, 47, 93–95, 119).

[111] Daniel Ott and Alexander Raschke. "Review improvement by requirements classification at Mercedes-Benz: Limits of empirical studies in educational environments." In: *Empirical Requirements Engineering (EmpiRE), 2012 IEEE Second International Workshop on*. IEEE. 2012, pp. 1–8 (cit. on p. 95).

[112] Jan Peleska et al. "A real-world benchmark model for testing concurrent real-time systems in the automotive domain." In: *Testing Software and Systems*. Springer, 2011, pp. 146–161 (cit. on p. 87).

[113] Marie-Agnès Peraldi-Frati et al. "Timing Modeling with AUTOSAR." In: () (cit. on p. 86).

[114] Amalinda Post and Jochen Hoenicke. "Formalization and analysis of real-time requirements: a feasibility study at BOSCH." In: *Verified Software: Theories, Tools, Experiments*. Springer, 2012, pp. 225–240 (cit. on pp. 61, 64, 65, 95).

[115] Amalinda Post, Jochen Hoenicke, and Andreas Podelski. "rt-inconsistency: a new property for real-time requirements." In: *Fundamental Approaches to Software Engineering*. Springer, 2011, pp. 34–49 (cit. on pp. 61, 64, 65, 95).

[116] Amalinda Post, Jochen Hoenicke, and Andreas Podelski. "Vacuous real-time requirements." In: *Requirements Engineering Conference (RE), 2011 19th IEEE International*. IEEE. 2011, pp. 153–162 (cit. on pp. 61, 64, 65, 95).

[117] Amalinda Post, Igor Menzel, and Andreas Podelski. "Applying restricted english grammar on automotive requirements—does it work? a case study." In: *Requirements Engineering: Foundation for Software Quality*. Springer, 2011, pp. 166–180 (cit. on pp. 61, 64, 65, 95).

[118] The Associated Press. *Jury Finds Toyota Liable in Fatal Wreck in Oklahoma*. 2013. URL: http://www.nytimes.com/2013/10/25/business/jury-finds-toyota-liable-in-fatal-wreck-in-oklahoma.html?_r=0 (cit. on p. 78).

[119] S Ramesh and A Gadkari. "Rigorous model-based design & verification flow for in-vehicle software." In: *Design Automation Conference (DAC), 2011 48th ACM/EDAC/IEEE*. IEEE. 2011, pp. 13–16 (cit. on p. 99).

[120] Stefan Resmerita et al. "Migration of legacy software towards correct-by-construction timing behavior." In: *Foundations of Computer Software. Modeling, Development, and Verification of Adaptive Systems*. Springer, 2011, pp. 55–76 (cit. on pp. 81, 85).

[121] Peter Rosina. *Kooperationsprojekt ONTORULE (Slides)*. http://www.bicc-net.de/workspace/uploads/subfeatures/downloads/7-peter-rosina-1330035757.pdf. Feb. 2012 (cit. on pp. 26, 28, 94).

[122] Johannes-Joerg Rueger et al. "MDG1: The New, Scalable, and Powerful ECU Platform from Bosch." In: *Proceedings of the FISITA 2012 World Automotive Congress*. Springer. 2013, pp. 417–425 (cit. on pp. 61, 63, 65).

[123] Christian de Sainte Marie, Miguel Iglesias Escudero, and Peter Rosina. "The ONTORULE project: Where ontology meets business rules." In: *Web Reasoning and Rule Systems*. Springer, 2011, pp. 24–29 (cit. on pp. 26, 28, 94).

[124] Holger Schmid and Robert Siwy. "Management verteilter Entwicklungen - reduzierte Entwicklungskosten und hoher Qualitätsstandard durch Einsatz einer Modellbibliothek." In: *Elektronik Automotive* 9 (2006), pp. 56–59 (cit. on pp. 37, 87).

[125] Karsten Schmidt et al. "Design patterns for highly integrated ECUs with various ASIL levels." In: *ATZelektronik worldwide* 7.1 (2012), pp. 22–27 (cit. on pp. 18, 26–28, 107).

[126] Dipl-Ing Oliver Schütze and Dipl-Ing Carsten Rustige. "Testmanagement für Antriebe im Nutzfahrzeug." In: *ATZelektronik* 6.1 (2011), pp. 40–43 (cit. on pp. 40, 42, 47, 119).

[127] Siemens AG. *Thinking for the long term Providing answers - Siemens Annual Report 2013*. http://www.siemens.com/investor/pool/en/investor_relations/siemens_ar_2013.pdf. 2013 (cit. on p. 55).

[128] Roberto S Silva Filho and Christof J Budnik. "An Integrated Model-Driven Approach for Mechatronic Systems Testing." In: *Proceedings of the 2012 IEEE Fifth International Conference on Software Testing, Verification and Validation*. IEEE Computer Society. 2012, pp. 447–456 (cit. on pp. 56–59, 119).

[129] Roberto Silveira Silva Filho et al. "Experiences using Tedeso: an extensible and interoperable model-based testing platform." In: *Automated Software Engineering* (2013), pp. 1–39 (cit. on pp. 56–58, 60, 87, 95, 119).

[130] Red Bend Software. *Update ECUs using Delta- and Over-the-Air-Technology*. Jan. 2014. URL: http://www.vector.com/portal/medien/cmc/events/Webinars/2014/Vector_RedBend_Webinar_Flashing_over_the_air_and_delta_technology_20140121_EN.pdf (cit. on p. 116).

[131] Sven Söhnlein et al. "Software reliability assessment based on the evaluation of operational experience." In: *Measurement, Modelling, and Evaluation of Computing Systems and Dependability and Fault Tolerance*. Springer, 2010, pp. 24–38 (cit. on pp. 53, 54).

[132] Springer International Publishing AG. *SpringerLink*. 2014. URL: http://link.springer.com/ (cit. on pp. 4, 7).

[133] Mugur Tatar, Jakob Mauss, and Andreas Junghanns. *Systematic Test and Validation of Automotive Systems*. http://www.qtronic.com/doc/TestWeaverIntro.pdf. 2012 (cit. on p. 52).

[134] Sebastian Thiel and Frank Derichsweiler. "Petri net based verification of causal dependencies in electronic control unit test cases." In: *Computer Software and Applications Conference Workshops (COMPSACW), 2011 IEEE 35th Annual*. IEEE. Munich, Germany: IEEE, 2011, pp. 143–148 (cit. on pp. 18–20, 22, 26, 28, 70, 118).

[135] Steve Toeppe et al. "Practical validation of model based code generation for automotive applications." In: *Digital Avionics Systems Conference, 1999. Proceedings. 18th*. Vol. 2. IEEE. 1999, 10–A (cit. on pp. 71, 72).

[136] E Troubitsyna. "Rodin Deliverable D18: Intermediate Report on Case Study Development." In: *Project IST-511599, School of Computing Science, University of Newcastle* (2006) (cit. on p. 103).

[137] Anthony Tsakiris. "Managing Software Interfaces of On-Board Automotive Controllers." In: *IEEE software* 28.1 (2011) (cit. on pp. 74, 77).

[138] Karen Twyford. *Transcript of morning trial proceedings had on the 14th day of october, 2013*. 2013 (cit. on p. 78).

[139] Dipl-Ing Johannes Wiessalla et al. "Modellbasierte Erzeugung von Testfällen mit integrierter Fehleranalyse." In: *ATZelektronik* 7.1 (2012), pp. 62–67 (cit. on pp. 75, 76, 87).

[140] Dirk Zitterell and Sebastian Thiel. "Automatisierter funktionaler Steuergerätetest mit der EXtended Automation Method (EXAM)." In: *GI Jahrestagung (2)*. 2010, pp. 351–356 (cit. on pp. 18, 22, 28, 70, 86).

[141] Albert Zündorf et al. "Using Graph Grammars for Modeling Wiring Harnesses–An Experience Report." In: *Graph transformations and model-driven engineering*. Springer, 2010, pp. 512–532 (cit. on p. 86).