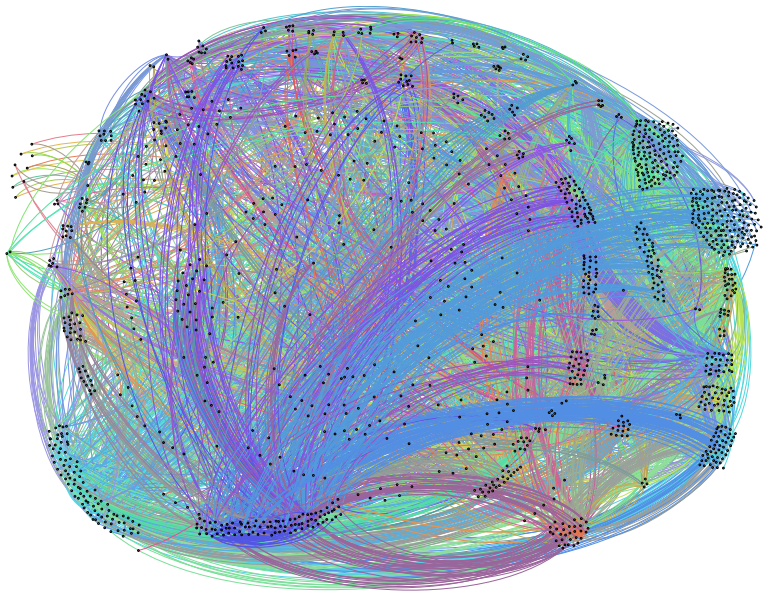
REFERENCE ARCHITECTURES FOR HIGHLY AUTOMATED DRIVING

SAGAR BEHERE



DOCTORAL THESIS IN MACHINE DESIGN
KTH ROYAL INSTITUTE OF TECHNOLOGY
STOCKHOLM, SWEDEN, 2016

Cover illustration: The complexity of architecture

The image shows a subset of data flows in the system architecture of a heavy commercial truck made by Scania CV AB, Sweden. Nodes represent logical functions, edges represent inter-function communication. There are ca. 1500 nodes and 5000 edges. Nodes are clustered by ECUs.

Image courtesy: Viktor Kaznov, Systems Architect, Scania CV AB

TRITA-MMK 2015:09
ISSN 1400-1179
ISRN KTH/MMK/R-15/09-SE
ISBN 978-91-7595-757-9

KTH School of Industrial
Engineering and Management
100 44 Stockholm
Sweden

Academic thesis, which with the approval of KTH Royal Institute of Technology, will be presented for public review in fulfillment of the requirements for a Doctorate of Technology in Machine Design. The public review is held in room Kollegiesalen, Brinellvägen 8, 100 44 Stockholm, Sweden on 2016-01-22 at 09:00.

Typeset with LaTeXusing the classicthesis template by André Miede.
Print: Universitetsservice US AB

I see God in
the instruments and the mechanisms that
work reliably,
more reliably than the limited sensory departments of
the human mechanism.

— R. Buckminster Fuller, *Whole Earth Catalog*

ABSTRACT

Highly automated driving systems promise increased road traffic safety, as well as positive impacts on sustainable transportation by means of increased traffic efficiency and environmental friendliness. The design and development of such systems require scientific advances in a number of areas. One area is the vehicle's electrical/electronic (E/E) architecture. The E/E architecture can be presented using a number of *views*, of which an important one is the *functional* view. The functional view describes the decomposition of the system into its main logical components, along with the hierarchical structure, the component inter-connections, and requirements. When this view captures the principal ideas and patterns that constitute the foundation of a variety of specific architectures, it may be termed as a *reference* architecture. Two reference architectures for highly automated driving form the principal contribution of this thesis. The first reference architecture is for cooperative driving. In a cooperative driving situation, vehicles and road infrastructure in the vicinity of a vehicle continuously exchange wireless information and this information is then used to control the motion of the vehicle. The second reference architecture is for autonomous driving, wherein the vehicle is capable of driver-less operation even without direct communication with external entities. The description of both reference architectures includes their main components and the rationale for how these components should be distributed across the architecture and its layers. These architectures have been validated via multiple real-world instantiations, and the guidelines for instantiation also form part of the architecture description. A comparison with similar architectures is also provided, in order to highlight the similarities and differences. The comparisons show that in the context of automated driving, the explicit recognition of components for semantic understanding, world modeling, and vehicle platform abstraction are unique to the proposed architecture. These components are not unusual in architectures within the Artificial Intelligence/robotics domains; the proposed architecture shows how they can be applied within the automotive domain. A secondary contribution of this thesis is a description of a lightweight, four step approach for model based systems engineering of highly automated driving systems, along with supporting model classes. The model classes cover the concept of operations, logical architecture, application software components, and the implementation platforms. The thesis also provides an overview of current implementation technologies for cognitive driving intelligence and vehicle platform control, and recommends a specific setup

for development and accelerated testing of highly automated driving systems, that includes model- and hardware-in-the-loop techniques in conjunction with a publish/subscribe bus. Beyond the more "traditional" engineering concepts, the thesis also investigates the domain of machine consciousness and computational self-awareness. The exploration indicates that current engineering methods are likely to hit a *complexity ceiling*, breaking through which may require advances in how safety-critical systems can self-organize, construct, and evaluate internal models to reflect their perception of the world. Finally, the thesis also presents a functional architecture for the brake system of an autonomous truck. This architecture proposes a reconfiguration of the existing brake systems of the truck in a way that provides dynamic, diversified redundancy, and an increase in the system reliability and availability, while meeting safety requirements.

SAMMANFATTNING

Högautomatiserade fordonssystem ger förhoppningar om ökad trafiksäkerhet samt positiva effekter för hållbara transporter genom ökad trafikeffektivitet och miljövänlighet. Design och utveckling av systemen kräver vetenskapliga framsteg i flera områden. Ett område är fordonets elektriska/elektroniska (E/E) arkitektur. E/E arkitekturen kan presenteras med hjälp ett antal vyer; denna avhandling fokuserar på den funktionella vyn. Den funktionella vyn beskriver nedbrytningen av systemet i dess huvudsakliga logiska komponenter, tillsammans med den hierarkiska strukturen, komponentinternaanslutningar, gränssnitt och krav. När denna vy sammanlänkas med principer och mönster som utgör grunden för en mängd olika specifika arkitekturer, kan den betecknas som en referensarkitektur. Två referensarkitekturer för högautomatiserad körning utgör det huvudsakliga bidraget i denna avhandling. Den första referensarkitekturen är för kooperativ körning. I en kooperativ körsituation sker kontinuerligt utbyte av trådlös information mellan fordon och väginfrastruktur i närheten av fordonet, denna information används sedan för att styra fordonets rörelse. Den andra referensarkitekturen är för autonom körning, varvid fordonet kan köras förarlöst även utan direkt kommunikation med externa enheter. Beskrivningen av de båda referensarkitekturerna inkluderar deras huvudkomponenter och motiven för hur dessa komponenter bör fördelas över arkitekturen och dess skikt. Arkitekturerna har validerats via flera verklighetsbaserade instansieringar och riktlinjerna för instansiering ingår också som en del av arkitekturens beskrivning. Referensarkitekturerna har jämförts med liknande arkitekturer i syfte att lyfta fram likheter och skillnader. Jämförelserna visar att explicit inkludering av komponenter för semantisk förståelse, världsmodellering och fordonsplattformsabstraktion är unika för de föreslagna arkitekturerna i kontexten automatiserad körning. Dessa komponenter är inte ovanliga i arkitekturer inom artificiell intelligens-/robotik; de föreslagna arkitekturerna visar hur de kan tillämpas inom fordonsdomänen. Ytterligare ett bidrag i avhandlingen är en beskrivning av en lättviktsmetod i fyra steg för modellbaserad systemkonstruktion av högautomatiserade körsystem, tillsammans med stödjande modellklasser. Modellklasserna omfattar begreppen aktiviteter, logisk arkitektur, applikationsprogramvarukomponenter och plattformar för implementering. Avhandlingen ger också en översikt av aktuell teknik för realisering av artificiell intelligens och fordonsplattform, och rekommenderar specifika metoder för utveckling och accelererad testning av högautomatiserade körsystem. Dessa metoder inkluderar modell- och hårdvaru återkopplingsteknik (eng. ”hardware-in-the loop”) tillsammans

med publicerings/prenumerations kommunikation. Bortom de mer "traditionella" tekniska koncepten, utforskar avhandlingen även området för maskinmedvetande och medvetenhet. Undersökningen visar att nuvarande tekniska metoder sannolikt kommer att nå ett komplexitetstak. För ett genombrott krävs framsteg inom hur säkerhetskritiska system kan självorganisera sig samt konstruera och utvärdera interna modeller för att spegla sin uppfattning om världen. Slutligen presenterar avhandlingen en funktionell arkitektur för bromssystemet för en autonom lastbil. Arkitekturen föreslår en omkonfigurering av de befintliga bromssystem hos lastbilen på ett sätt som tillhandahåller dynamisk, diversifierad redundans och en ökning av systemtillförlitlighet och -tillgänglighet.

APPENDED PUBLICATIONS

PUBLICATION A

Architecture challenges for intelligent autonomous machines: An industrial perspective

Sagar Behere, Fredrik Asplund, Andreas Söderberg, Martin Törngren

Proceedings of the 13th International Conference on Intelligent Autonomous Systems (IAS-13). Advances in Intelligent Systems and Computing, Springer International Publishing, 2015. ISBN: 9783319083384

Sagar wrote the text. Fredrik and Andreas edited and provided insights on safety related topics. Martin assisted in organizing the paper structure, reviewed and provided feedback.

PUBLICATION B

The Development of a Cooperative Heavy-Duty Vehicle for the GCDC 2011: Team Scoop

Mårtensson, J. ; Behere, S. et al

IEEE Transactions on Intelligent Transportation Systems, vol. 13, no. 3, pp.1033-1049, September 2012 doi: 10.1109/TITS.2012.2204876

Sagar wrote section II on Architecture. Other authors wrote sections about their individual work related to control, communications, etc.

PUBLICATION C

A reference architecture for cooperative driving

Sagar Behere, Martin Törngren, De-Jiu Chen

Journal of Systems Architecture, Vol. 59, Issue 10, Part C, November 2013, Pages 1095-1112 DOI: 10.1016/j.sysarc.2013.05.014

Sagar wrote the text. Martin and Chen reviewed and provided feedback.

PUBLICATION D

A functional reference architecture for autonomous driving

Sagar Behere, Martin Törngren

[ACCEPTED] Journal of Information and Software Technology
Elsevier BV. Extended version of publication L.

Sagar wrote the text. Martin reviewed and provided feedback.

PUBLICATION E

Systems engineering and architecting for autonomous driving

Sagar Behere, Martin Törngren

[ACCEPTED] Chapter in book "Automated Driving - Safer and more efficient future driving", Springer International Publishing AG (to be published in early 2016)

Sagar wrote the text. Martin edited, reviewed and provided feedback.

PUBLICATION F

A functional brake architecture for autonomous heavy commercial vehicles

Sagar Behere, Viacheslav Izosimov, Xinhai Zhang, Martin Törngren

[SUBMITTED] SAE World Congress , 2016

Sagar wrote most of the text. Viacheslav edited content on hazard analysis and ISO26262. Xinhai helped with simulations. Martin reviewed and provided feedback.

PUBLICATION G

Architecture support for automobile autonomy: A state of the art survey

Sagar Behere

Technical report, 2015, Dept. of Machine Design
KTH TRITA - MMK 2015:08, ISSN 1400-1179
ISRN/KTH/MMK/R15/08-SE

OTHER PUBLICATIONS

PUBLICATION H

Educating embedded system hackers: A practitioner's perspective

Sagar Behere, Martin Törngren

Proceedings of the Workshop on Embedded Systems Education (WESE) 2014, ACM Publications

Sagar wrote the text. Martin reviewed and provided feedback.

PUBLICATION I

Educating embedded system hackers: A practitioner's perspective

Sagar Behere, Martin Törngren

[ACCEPTED] SIGBED Review, special edition on education. Extended version of publication H.

Sagar wrote the text. Martin reviewed and provided feedback.

PUBLICATION J

Towards autonomous embedded systems

Sagar Behere, Martin Törngren, Jad El-khoury, De-Jiu Chen

First Open EIT ICT Labs Workshop on Cyber-Physical Systems Engineering (EIT CPSE) 2013, Trento, Italy

Sagar wrote the text. Others reviewed and provided feedback.

PUBLICATION K

Cooperative driving according to Scoop

Alam A., Behere S. et al

Real-Time in Sweden (RTiS) 2011, Västerås, Sweden

Sagar wrote the text. Martin reviewed and provided feedback.

PUBLICATION L

A functional architecture for autonomous driving

Sagar Behere, Martin Törngren

First international Workshop on Automotive Software Architectures (WASA) 2015, Montréal, Canada

Sagar wrote the text. Martin reviewed and provided feedback.

PUBLICATION M

Scoop Technical Report: Year 2011

Sagar Behere

Technical report, 2011, Dept. of Machine Design
KTH TRITA - MMK2012:12, ISSN 1400-1179
ISRN/KTH/MMK/R-12/12-SE

ACKNOWLEDGMENTS

On a cold winter morning during my early PhD efforts, I was skating over the frozen Brunnsviken lake together with my supervisor, Martin Törngren. Martin set forth at a tremendous speed, while I clung to his outstretched hand and let myself be dragged along behind, going, "*Wheeeeeee*". That has pretty much been the metaphor for this PhD work. Under Martin's nurturing guidance and encouragement, I learned to skate, and sometimes overstretched and ended up on thin ice. The metaphor with the PhD work still holds.

De-Jiu Chen could always help me to find the right words to express my thinking. Discussing research ideas with Chen is like drinking from a firehose.

Jad El-khoury guided me through my licentiate thesis with refreshing bluntness. His approach to planning, organization, and allocation of responsibility has since become a part of my own programming.

Martin Grimheden and Jan Wikander are supporters of dreams. Their positive influence has undoubtedly exceeded the awareness I have of their efforts.

Frédéric Loiret is the kind of guy with whom you can discuss anything and everything. Conversations with him spiced up my world, and his vigor and enthusiasm pulled me through many a dark period.

Xinhai Zhang is a quiet, reticent guy until you need his help. Then he grows into a giant whom you can solidly rely on. Be it brake system simulations or moving heavy boxes around, Xinhai unhesitatingly came to my aid.

Katja Gradin's company is superior even to coffee and cake, imagine that. Realizing that she genuinely cares about me, left me feeling overwhelmed more often than I admitted.

Didem Gürdür attempted to educate me on feminism, veganism, activism, and other -isms. She made a difference, while tolerating my teasing. Daniel Frede, Mohammad Khodabakhshian, Fredrik Asplund, and Daniel Malmquist tolerated my gratuitous interruptions to their work with grace. I gleefully disturbed them whenever I felt the itch to survey the social state of the art.

At Scania, Per Roos, Viktor Kaznov, and Johan Svahn, better known as *the three musketeers*, provided a "home away from home" for my research. The launch of the ARCHER project is an eloquent testimony to the synergy of our work. As well, the colleagues at the Volvo Car Corporation and others in the DFEA2020 and FUSE projects enriched this research with industrial insights.

In a league of their own, are my parents and sister — for their relentless love and support. Without you, life itself would be pointless.

CONTENTS

1 INTRODUCTION 1
- 1.1 Background 1
- 1.2 Thesis scope and relevance 5
- 1.3 Methodology 6
- 1.4 Hypothesis and research question 10
- 1.5 Overview of contributions 12
- 1.6 Impact 14

2 READING GUIDE AND CONTRIBUTIONS 17
- 2.1 Reading guide — Publications 17
- 2.2 Frame of reference 18
- 2.3 Executive summary of contributions 21
  - 2.3.1 Architecture challenges for intelligent autonomous machines ([A]) 22
  - 2.3.2 Platooning and cooperative driving ([B] and [C]) 25
  - 2.3.3 Autonomous driving ([D] and [E]) 30
  - 2.3.4 Model based systems engineering ([E]) 34
  - 2.3.5 The context of machine consciousness ([E]) 36
  - 2.3.6 A brake architecture for autonomous trucks ([F]) 38

3 DISCUSSION 43
- 3.1 Future work 46

BIBLIOGRAPHY 49

LIST OF FIGURES

Figure 1 SAE levels of automated driving [4] 2
Figure 2 Some areas contributing to highly automated driving 3
Figure 3 Thesis context — main contributions are in functional architecture 6
Figure 4 Prototypes created during the thesis work 8
Figure 5 Categories of thesis contributions 13
Figure 6 Frame of reference for automated driving architectures 18
Figure 7 Scoop architecture: Logical functions 26
Figure 8 Scoop architecture: Hardware view 27
Figure 9 Scoop architecture: Implementation view 27
Figure 10 A reference architecture for cooperative driving 29
Figure 11 Main functional components for autonomous driving 30
Figure 12 A functional reference architecture for autonomous driving 33
Figure 13 Systems engineering model classes 34
Figure 14 Partial view of modeling artifacts and allocation links 35
Figure 15 General pattern for brake architecture 39
Figure 16 Mapping from existing brake architecture to general pattern 40
Figure 17 Simulation of 'Omission' condition in primary brake (at 2.6s mark, note no reduction in speed) 41

GLOSSARY

ADL Architecture Description Language. A computer language used to create a description of a system architecture.

ARCHITECTURE A system's blueprint as reflected in the key building blocks of the system, their composition, their interplay, the resulting extra-functional properties, and so on. More formally, it is defined within a systems engineering context by ISO 42010 as, "..fundamental concepts or properties of a system in its environment, embodied in its elements, relationships, and in the principles of its design and evolution." [3]

ARTIFICIAL INTELLIGENCE The science and engineering of making intelligent machines, especially intelligent computer programs [12].

AUTOMATED DRIVING SYSTEM The hardware and software that is collectively capable of performing all aspects of the dynamic driving task for a vehicle (whether part time or full time) [4].

AUTOMOTIVE E/E ARCHITECTURE The architecture of an automobile's Electrical/Electronic systems. It includes computing hardware, software, communication links as well as functional hierarchies and their distribution across the architecture.

AUTONOMOUS MACHINES Machines which can perform their task(s) with no (or minimal) human intervention. Based on definition of Intelligent Autonomous Machine found in [13]

COMPLEX SYSTEM A system that can be analyzed into many components having relatively many relations among them, so that the behavior of some components may depend on the behavior of others, and the behavior of the system cannot simply be derived from the summation of individual components' behavior. Based on definition in [17]

CONSCIOUSNESS The fact of awareness by the mind of itself and the world, where awareness is further defined as knowledge or perception of a situation or fact [15].

COOPERATIVE DRIVING Driving in a situation where vehicles and road infrastructure in the vicinity of a vehicle continuously exchange wireless information, and where this information is then used to control the motion of the vehicle.

ECU Electronic Control Unit. Typically, an embedded computer controlling one or more vehicle functions.

EMBEDDED SYSTEM A computer system that is part of a larger system and performs some of the requirements of that system [2].

FUNCTIONAL ARCHITECTURE VIEW The functional architecture view describes the decomposition of the system into its main logical components, along with the hierarchical structure, the component inter-connections, interfaces, and data-flows. The description is made without prejudice to any particular technological implementation. This description is written in the flavor of ISO 42010 [3]

INTELLIGENCE The ability of a system to act appropriately in an uncertain environment, where appropriate action is that which increases the probability of success, and success is the achievement of behavioral subgoals that support the system's ultimate goal. [5]

MBSE Model Based Systems Engineering - the formalized application of modeling to support system requirements, design, analysis, verification and validation activities beginning in the conceptual design phase and continuing throughout development and later life cycle phases [9].

REFERENCE ARCHITECTURE A predefined architectural pattern, or set of patterns, possibly partially or completely instantiated, designed, and proven for use in particular business and technical contexts, together with supporting artifacts to enable their use [11].

SYSTEM A set of elements in interaction [18]. More specifically, an *engineered system* is defined as an interacting combination of elements to accomplish a defined objective. These include hardware, software, firmware, people, information, techniques, facilities, services, and other support elements [10].

SYSTEMS ENGINEERING Systems Engineering is an interdisciplinary approach and means to enable the realization of successful systems. It focuses on defining customer needs and required functionality early in the development cycle, documenting requirements, then proceeding with design synthesis and system validation while considering the complete problem [8].

TVVQ Testing, Verification, Validation, Qualification

1 INTRODUCTION

> "Begin at the beginning," the King said gravely, "and go on till you come to the end: then stop."
>
> — Lewis Carroll, Alice in Wonderland

1.1 BACKGROUND

In recent years, technologies that reduce (or eliminate) the need of human intervention in the control of machines, have steadily gained momentum. The application of such technologies in the automotive domain has resulted in Advanced Driver Assistance Systems (ADAS) such as Adaptive Cruise Control (ACC), Lane Keeping Assist (LKA), Automatic Emergency Braking (AEB), and Traffic Jam Assist (TJA). Such systems control (parts of) the vehicle motion in scenarios where human control performance can be improved upon. Technology trends in ADAS systems point to a future where road vehicles which may no longer require human drivers. The elimination of human drivers has the potential to improve safety of road traffic, because human behavior and limitations are the cause of almost 94% [14] of incidents that compromise road safety. Beyond safety, improvements in comfort, traffic throughput, and eco-friendliness are also expected. Towards these ends, numerous vehicle prototypes have been built, which require no human driver intervention when operating in specific conditions. Such vehicles are commonly referred to as "self-driving", "driverless", "autonomous", "unmanned", or "robotic" in vernacular English. This thesis is concerned with the design and development of such vehicles.

In the lexicon of industry experts and government agencies, a carefully created set of definitions is gaining acceptance. These definitions were created in 2014 by SAE International[1] and published in the standard SAE J3016 [4]. The standard defines the terms *dynamic driving task* and *driving mode* and uses these definitions, along with others, to describe six progressive levels of *driving automation*. These levels are summarized in Figure 1. At one extreme, level zero refers to no

1 SAE International, formerly established as the Society of Automotive Engineers, is a globally active professional association and standards organization for engineering professionals in transportation industries. See http://www.sae.org

SAE level	Name	Narrative Definition	Execution of Steering and Acceleration/ Deceleration	*Monitoring* of Driving Environment	Fallback Performance of *Dynamic Driving Task*	System Capability *(Driving Modes)*
***Human driver* monitors the driving environment**						
0	No Automation	the full-time performance by the *human driver* of all aspects of the *dynamic driving task*, even when enhanced by warning or intervention systems	Human driver	Human driver	Human driver	n/a
1	Driver Assistance	the *driving mode*-specific execution by a driver assistance system of either steering or acceleration/deceleration using information about the driving environment and with the expectation that the *human driver* perform all remaining aspects of the *dynamic driving task*	Human driver and system	Human driver	Human driver	Some driving modes
2	Partial Automation	the *driving mode*-specific execution by one or more driver assistance systems of both steering and acceleration/ deceleration using information about the driving environment and with the expectation that the *human driver* perform all remaining aspects of the *dynamic driving task*	**System**	Human driver	Human driver	Some driving modes
***Automated driving system* ("system") monitors the driving environment**						
3	Conditional Automation	the *driving mode*-specific performance by an *automated driving system* of all aspects of the dynamic driving task with the expectation that the *human driver* will respond appropriately to a *request to intervene*	System	**System**	Human driver	Some driving modes
4	High Automation	the *driving mode*-specific performance by an automated driving system of all aspects of the *dynamic driving task*, even if a *human driver* does not respond appropriately to a *request to intervene*	System	System	**System**	Some driving modes
5	Full Automation	the full-time performance by an *automated driving system* of all aspects of the *dynamic driving task* under all roadway and environmental conditions that can be managed by a *human driver*	System	System	System	**All driving modes**

Figure 1: SAE levels of automated driving [4]

automation, where all aspects of the driving task are performed by a human driver. At the other extreme, level five refers to full automation, where the automated driving system is capable of performing all aspects of the driving task under all those road and environmental conditions which can be managed by a human driver. SAE standards are not authoritative, and there remains plenty of dispute regarding whether such levels accurately and/or comprehensively capture various levels of (self-) driving functionality. Despite the dispute, the fact remains that SAE International has powerful influence in the automotive domain and SAE J3016 is (at the time of this writing) the most comprehensively thought out attempt in the industry to harmonize various definitions and classification systems. Therefore, we will refer back to the SAE automation levels in subsequent parts of this text.

Regarding terminology

There exists a vocal minority who insists that terms like "self-driving", "autonomous" etc. are inappropriate and misleading, and that their wanton usage contributes to mass confusion about the topic of road vehicle *automation*. The research covered in this thesis has been conducted over a period of five years, when the terminological war had not yet started. As such, the term "autonomous driving" is used often, but always with an accompanying description to clarify its meaning in context. The usage roughly corresponds to SAE levels four and five.

The development of highly automated driving systems and their introduction into mass markets in the form of consumer products, require progress in broadly four different areas, as shown in Figure 2.

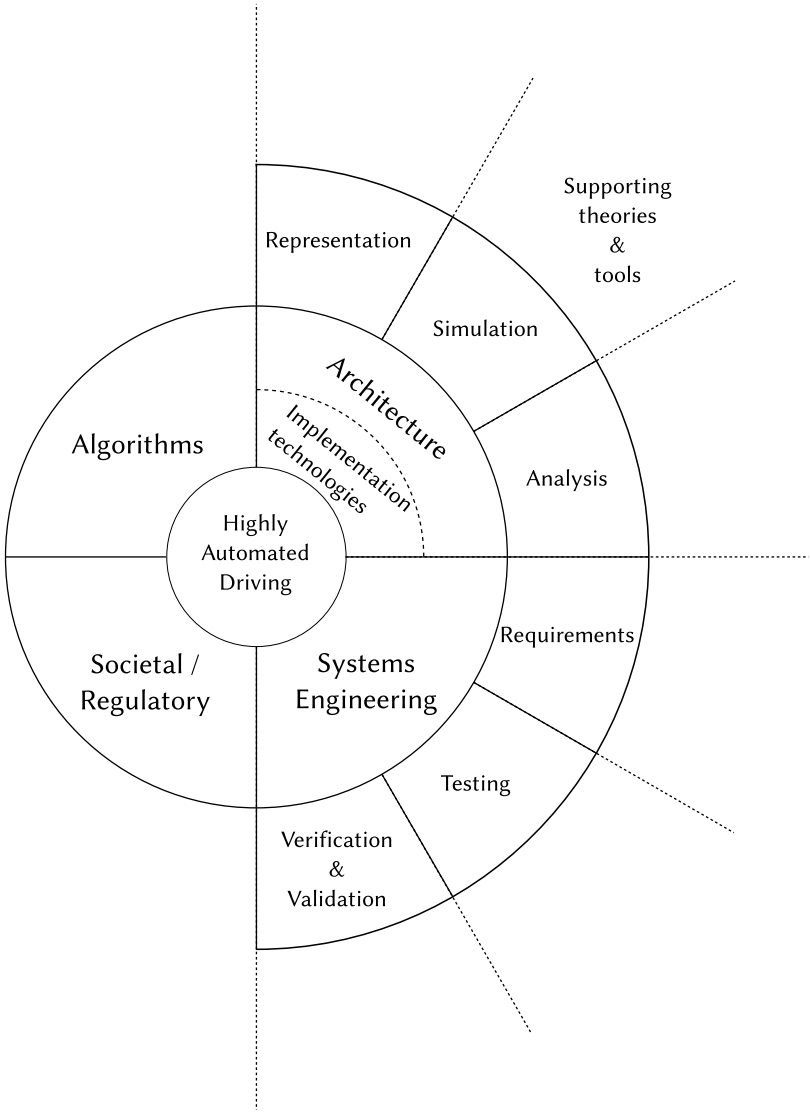


Figure 2: Some areas contributing to highly automated driving

ALGORITHMS The formal mathematical knowledge for solving specific problems in a sequence of computation steps

ARCHITECTURE AND IMPLEMENTATION TECHNOLOGIES The design and realization of systems which utilize the algorithms to perform desired functional tasks, while meeting other requirements such as modularity and reliability.

SYSTEMS ENGINEERING The systematic methods and processes used to construct systems

SOCIETAL/REGULATORY Regulatory and legal frameworks to permit system operation, including topics related to liability, insurance etc.

Within each of these areas lies a practically endless scope for research. The most relevant areas for this thesis are architecture and systems engineering, whose sub-areas are shown in the outer half-circle of Figure 2 (Representation, Simulation, ... , Verification & Validation). The development of architectures is supported by research for representation of those architectures. The representation is usually in the form of a set of models. The models may capture different aspects of the system and facilitate different types of simulations and analyses, for example, safety. Similarly, development process are supported by work tasks that create and maintain various types of artifacts, for example requirements, and further sub-processes for activities like testing, verification and validation. The system artifacts as well as the related work tasks are often highly interdependent. For example, can an architecture model be linked with a set of requirements, such that a change in either the model or the requirements can be propagated from the one to the other? Or, can the architecture be represented in such a way that it is possible to automatically generate software code from that representation? And what development process would facilitate automated testing of the generated code? Would such an architecture representation lend itself to the type of safety analysis that can be performed by some specific tool? The various influences, interactions, and inter-dependence between the work tasks not only drive the state of the art in research, they also impose very concrete constraints and limitations on the technologies, tools, and development processes used within a specific development project. And of course, there exist supporting technologies for the supporting technologies for up to many levels[2].

The state of the art in highly automated driving has reached a stage where the basic algorithms and implementation technologies (including sensors, actuators, computing, and communications) have matured to the point that it is feasible to imagine a functional system that can replace a human driver, in many different driving situations. Such systems have seen many proof-of-concept implementations, but significant challenges still exist in creating mature systems that provably retain performance and safety when faced with operational situations that are overwhelmingly complex and/or unforeseen by their designers. These challenges need to be addressed by research in the core disciplines (automatic control, artificial intelligence, computer science, etc.) as well as in architecture, development processes, and their many supporting areas. The latter research is also needed to support the rapid utilization and deployment-into-products of new technical knowledge that emerges in the field.

2 It's turtles all the way down!

1.2 THESIS SCOPE AND RELEVANCE

This thesis work has been built upon a foundation of several practical projects, most of which resulted in concrete, physical demonstrators and prototypes. An example demonstrator is a partially self-driving R730 truck developed in cooperation with Scania CV AB of Sweden. The massive R730 is one of the most powerful production trucks in the world, featuring a 16.4 liter, 16 turbo diesel V8 engine that generates 730 horsepower and 3500 Nm of torque. This truck was equipped with an autonomous driving system whose technical design and development was led by the author during the early part of the thesis work. Given the commitment to creating fully functioning prototypes, this thesis work can be approached from four different perspectives (i) Systems implemented (ii) Development processes adopted (iii) Lessons learned and (iv) Scientific contributions. Of these, the fourth perspective (contributions made) is emphasized in this thesis text.

Referring back to Figure 2, the entirety of this thesis work lies in the areas of (i) architecture and implementation technologies and (ii) development processes. Since advanced prototypes were developed, it is natural that the work involved implementation technologies and architecture. The development necessarily required forms of architecture representation, system modeling, and analysis. Since the development was done in teams of industrial and academic researchers, some form of development process was necessarily utilized, which spanned requirements, testing, verification and validation. Over repeated prototype development, the lessons learned and best practices were carried forward. Minutiae of technical implementations are captured in referred technical reports, while the architectures themselves are prominently covered in this thesis work, via the appended publications.

Figure 3 shows the main architectural viewpoints considered: (i) Service taxonomy (ii) Functional architecture (iii) Software and (iv) Hardware. These viewpoints are described in publication A. The main focus of this thesis is on part (ii) Functional architecture. The functional architecture describes the decomposition of the system into its main logical components, along with the hierarchical structure, the component inter-connections, interfaces, and data-flows. The description is made without prejudice to any particular technological implementation.

The main thesis contributions are two so called *reference architectures* which can be instantiated in the context of particular projects. The reference architectures provide a description of all relevant functionalities that are needed to solve the problem, and indicate how the functionalities should be combined. This is accompanied by rationale, decision criteria, and the pros and cons of making particular choices.

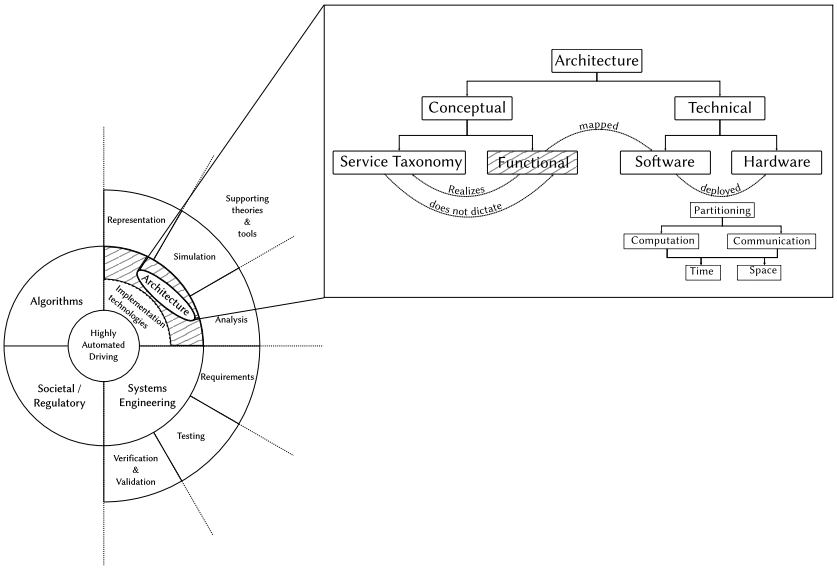


Figure 3: Thesis context — main contributions are in functional architecture

Examples and guidelines of how the reference solutions should be concretely implemented are also provided.

The architectures developed and validated during this thesis work are constrained in scope to the electrical/electronic (E/E) systems found on board a particular vehicle. Only the motion related systems are considered. That said, the architectures recognize and support the possibility that future vehicles are likely to communicate with other vehicles and the infrastructure and to function as autonomous units in a highly connected, intelligent transport ecosystem. Detailed analyses of functional safety is excluded from the scope, because the emphasis is on how the desired functionality can be achieved. However, we believe that a later inclusion of safety specific concerns should not result in disruptive changes to the proposed architectures.

The relevance of this thesis work lies in the fact that it provides proven solution patterns for the problem of architecting highly automated vehicles. The thesis work therefore forms a concise, first point of reference for creating specific technical solutions in the area.

1.3 METHODOLOGY

This thesis is based on a work of engineering. The engineer's goal is to create a sufficient solution to a need. The solution may be narrowly specific to the need, and it is reasonable to expect that there exist other solutions to satisfy the same need. The solution lays claim, not to being universally applicable, nor to be the best possible solution, but to be effective, satisfactory, and sometimes, optimal. Engineering re-

search is therefore characterized by an analysis of what constitutes a sufficient solution with the given constraints, followed by the design needed to achieve such a solution. Engineering research also emphasizes the systematic approach used for creating the solutions and the methodological and technical basis (knowledge, methods, tools) to facilitate the creation of the solutions. These statements of the nature of engineering, and the research methods described in the two upcoming paragraphs are based on content from [6], which contains an insightful analysis of the topic.

In the natural sciences, the goal of the scientist is to create knowledge which is objective, generalizable, universally true, and which can be used to describe an already existing state of affairs. Given that engineering is more concerned with creating apposite, hitherto non-existing solutions which may be narrowly specific, rather than a generally valid theory to explain observations, there is naturally some tension when engineering research methods are judged from perspectives grounded in the natural sciences. This has led to numerous research methods especially applicable to engineering, but there still remains a tendency to force-fit engineering research into a scientific conceptualization, especially with regards to generalizability.

The research conducted during this thesis closely aligns with an established research methodology known as *engineering design* [6]. This is one of the methodologies that is especially applicable to engineering, and acknowledges the purpose of engineering as creating sufficient solutions to particular needs. The principal thrust of the methodology is that research is an activity to create new knowledge, and using engineering design to create particular solutions results in new *situated knowledge*. The situated knowledge is specific to the circumstance, choice of technology, and the system architecture selected and provides "proof by construction" that the design idea is valid and effective [7]. The activity of design integrates fragments of existing knowledge to achieve a useful outcome. This includes, in the context of engineering design methodology, the application of scientific criteria like repeatability, reproduceability, separation of own versus others' contributions, along with a sound theoretical understanding of why the developed solutions work[3]. Such a design activity usually also involves advances beyond the state of art/practice. Furthermore, if discussions on potential generalization and validity are included, the overall result is a theory which links product requirements with the solution developed, and a method for developing the solution [6]. Following this methodology, four prototypes were created as shown in Figure 4. The design and implementation of each prototype yielded valuable insights which were used to improve the design and implementation of the next prototype.

3 The understanding of *why* a solution works in a certain way is a key difference between engineering design and merely building something that works.

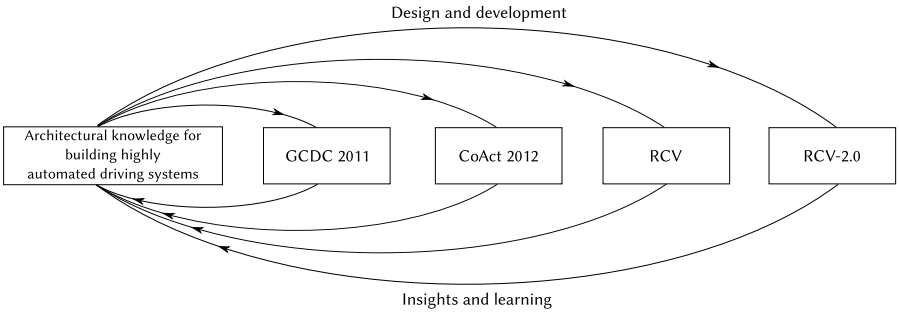


Figure 4: Prototypes created during the thesis work

To complement the design and prototyping efforts, the thesis work has also included state of the art surveys in the different research areas that are relevant to automated driving. This includes the "conventional" technological areas (embedded systems, component based software engineering ...), as well as reaches into realms of the cognitive theory of mind. Additionally, structured interaction with industrial practitioners has been organized via several workshops. Each of these workshops has been attended by well over fifty engineers from the domains of transport, industrial automation, telecom, aviation, and defense. The cross-domain perspectives have provided valuable learning opportunities and a few of the reports and publications described later in this text have come about directly as a result of these workshops.

The engineering solutions developed during this thesis work have been consciously created to be generalizable to a more representative set of problems. The generalizations are shown to be effective, via successful application to other similar problems. Specific technical implementations are documented in technical reports, while the generalized theory and identified solution patterns are described in the appended scientific publications. The scientific artifacts are so called "reference architectures" and the appended publications include useful commentary on how the reference architectures should be *instantiated* to create specific solutions. All the work is repeatable, and can be freely verified by other engineers. The rationale behind specific choices in the reference architectures is documented, and the publications also provide precise descriptions of the system boundaries and constraints within which the architectures are applicable. An evaluation of how well the architectures solve the intended problems is also included. Each publication describes the activities leading up to its content, but the overall progression of projects conducted during the PhD efforts is shown in Table 1.

In 2010, an architecture for autonomous longitudinal motion control was designed and implemented on an R730 commercial truck from Scania CV AB. The truck participated in the Grand Cooperative Driving Challenge (GCDC) 2011, wherein vehicles operated au-

YEAR(S)	PROJECTS	VEHICLE	PARTNERS	OUTCOME
2010-11	GCDC 2011	Heavy duty commercial truck	Scania CV AB	1. Autonomous longitudinal motion in platooning scenario [Pub. B] 2. A reference architecture for cooperative driving [Pub. C]
2011-12	CoAct 2012	Heavy duty commercial truck	Scania CV AB	Second, different instantiation of above mentioned reference architecture for cooperative driving
2013-14	DFEA2020 + FUSE + ARCHER	Passenger cars	Volvo Car Corporation + Scania CV AB	Problem analysis, methods, and a reference architecture for autonomous driving [Pubs. A,L]
2014-	RCV	Novel research vehicle prototype	Departments within KTH	Novel electric vehicle prototype with -by-wire control of steering and propulsion
2015-	RCV-2.0	Novel research vehicle prototype	KTH + A private company	Novel vehicle prototype with full perception stack and urban autonomous driving capabilities (under development)

Table 1: Projects contributing to thesis content

tonomously on a public highway in a platooning scenario, with constant wireless communication between participating vehicles and the environment. The architecture was refined and re-applied a year later, on a different truck (an R430 model), for the CoAct 2012 project. This project also involved a platooning scenario similar to the GCDC 2011 event, but it included more demanding operational situations like splitting and merging lanes, and overtaking. The accumulated architecture underwent further evaluation and analysis in the course of three different projects with various industrial partners, including a potential application to passenger cars. One of these projects was DFEA2020 — a large Swedish consortium project aimed at development of green, safe, and connected vehicles. Another project is FUSE — also a Swedish project, with a tighter focus on functional safety and architectures for autonomous driving. The third project is ARCHER — which investigates safety, reference architectures, and testing and verification techniques applicable to commercial trucks. The FUSE and ARCHER projects are still in progress. Starting from 2014, the architecture was then applied to a novel research concept vehicle (RCV) at KTH, with a view to endow it with autonomous driving capabilities. The RCV has an all-electric, drive-by-wire powertrain with a propulsion motor embedded inside each wheel, and active steering and camber control of all four wheels. Control of these components is included in a vehicle platform abstraction, which can receive maneuvering commands from a perception and planning layer. This was used to demonstrate basic self-parking capabilities with the aid of ultrasonic sensors. The architecture was then adapted to a second variant of the RCV (RCV-2.0), where it serves as the foundation for autonomous urban driving capabilities in situations where a human driver is not expected to be available (or capable) of taking over vehicle control.

1.4 HYPOTHESIS AND RESEARCH QUESTION

The engineering design research methodology described previously in Section 1.3 often deviates from a more traditional scientific research process, with respect to how hypotheses are constructed. This is because the traditional scientific research process consists of contemplating on observations, formulating a hypothesis, making predictions based on the hypothesis, conducting experiments and gathering data, investigating the success of the hypothetical predictions, and declaring the hypothesis as valid under specific conditions. In contrast, the question, "What hypothesis fits the observed facts?" may justifiably never be raised prior to the start of an engineering project. However, when differentiating between engineering and engineering design research, it is worth indulging this question *post hoc* in some form or the other because, as mentioned in [16], presenting a rational process

has the benefit of making the product (research) and design flow understandable, maintainable, and reusable.

So how can a hypothesis be constructed for engineering design research? Well, it can be trivially asserted that engineering answers the hypothesis that it is possible to build a solution to the particular need. To improve upon the hypothesis, we can look at the implicit assumption in creating a particular engineering solution. The hypothesis can then be generically rephrased as, "If the system is built «*this way*» then it might fulfill its intended purpose. It may even work in more generalized cases, or could be more easily modified for those cases." This is better, but still generic. Applying this to the particular engineering problems considered in this thesis, the resulting (implicit) hypothesis that underlies the thesis research can be stated as

> HYPOTHESIS: A reference architecture for highly automated driving exists, which is general enough to be repeatedly instantiated to provide sufficient solutions for a variety of automated driving tasks in a variety of scenarios. Such a reference architecture will contain solution patterns and other value additions to help architects build apposite solutions for automated driving, especially with regards to performance and safety.

The above hypothesis leads to several questions like

- What are the requirements, principles, and patterns for architectures of highly automated driving systems?
- What are idealized architectures for distributed, embedded systems that are expressly designed for high levels of automation?
- How does the potential absence of a human driver affect the architecture and associated safety analysis techniques?
- What are the technologies and development processes to implement highly automated driving systems?
- What is the impact of high levels of automation on testing, verification and validation methods?

Answers to these questions can be aggregated within reference architectures. The reference architectures need to be expressed at a level of abstraction where they contain sufficient information to provide breadth and depth of solutions i.e. they are non-trivial, but at the same time, are generic enough to be useful for more than one specific use case. Therefore, the overarching research question investigated by this thesis is

RESEARCH QUESTION: What is a suitable reference architecture for highly automated driving?

- What are the main obstacles to high levels of automation?
- What are the principal architectural components and their key interactions?
- How should the architecture be organized, and how should the components be distributed across the architecture?
- What are the key decisions, and the range of choices available in making those decisions?
- What is a suitable methododolgy to design and instantiate the architecture?

To answer the research question, it is *crucial* to have strong and recurring hands-on experience in repeatedly designing and implementing different instances of highly automated driving systems. This has indeed been one of the highlights of this thesis work, as described in Section 1.3.

1.5 OVERVIEW OF CONTRIBUTIONS

The thesis contributions can be organized into four categories: (i) Scientific theories (ii) Engineering (iii) State of the Art and Industrial challenges and (iv) Education. Figure 5 shows the category for each publication. The main scientific contribution of this thesis is a set of functional reference architectures for highly automated driving.

This section provides a quick overview of the contributions made by the appended publications. Later, Chapter 2 provides an executive summary and a reading guide through the main scientific results.

PUBLICATION A: A key aim of this thesis is to create technical solutions for highly automated driving which can be adopted by practicing engineers in the industry. Therefore, it is essential to gain an understanding of the obstacles to implementing such systems, from an industrial perspective. Some of the main obstacles are reported in this publication, which resulted from structured interaction (and subsequent analysis) with about 65 practicing engineers at a workshop on autonomous systems organized at KTH.

PUBLICATION B: The technical development of the cooperative driving architecture used in the GCDC 2011 event is reported in this publication. It covers architecture, communication, state estimation and control.

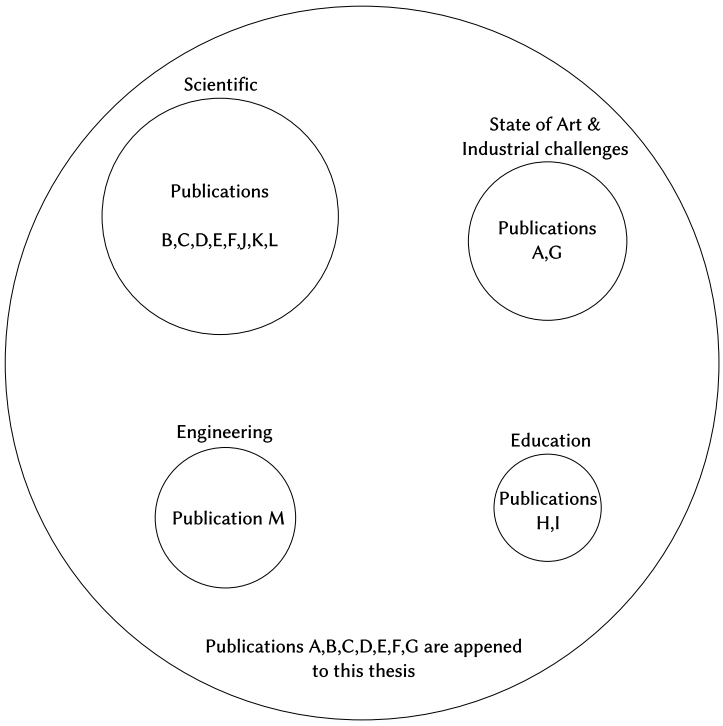


Figure 5: Categories of thesis contributions

PUBLICATION C: This publication develops upon and generalizes the architectural solution reported in publication B, resulting in a functional reference architecture for cooperative driving.

PUBLICATION D: This publication describes a functional reference architecture whose scope goes beyond cooperative driving towards highly automated driving where human drivers may be absent.

PUBLICATION E: This is a book chapter, summarizing a few key takeaways in functional architecture, model based systems engineering, and implementation technologies for autonomous driving. It also relates the shorter term technical developments of autonomous driving systems with longer term perspectives from the theory of mind and associated cognitive sciences.

These five publications thus report on a continuous generalization of automated driving architectures, with progressively wider application scope and validation.

PUBLICATION F: This is an exploration into safety criticality and reliability aspects of architecting, and presents a functional architecture for a braking system that can be applied to autonomous, commercial, heavy trucks.

PUBLICATION G: This is an early state of the art and literature survey of the research areas contributing to the creation of automated driving systems. It covers automotive E/E architectures, intelligent control and robotics architectures, and general embedded systems and software development. A synthetic discussion highlighting differences between automotive and other domains is also included. More focused, topic specific state of art sections are also provided in publications B, C and D.

During the course of the thesis work, especially during the intensive hands-on prototyping phases, it became clear that a specific set of technical skills is needed by engineers developing such systems. A summary of these skills, along with how they can be incorporated into traditional embedded systems education, including sets of exercises, is provided in PUBLICATIONS H-I. Interim (but peer reviewed) scientific results that contribute to the the main publications mentioned above are presented in PUBLICATIONS J-L. Many of the specific technology choices and other details used for practical implementations are described in PUBLICATION M.

1.6 IMPACT

It is expected that the reference architectures and their descriptions published during this thesis work will be one of the starting points when initiating a new architecture for highly automated driving. As of October 2015, the early publications B and C have been cited 20 and 12 times respectively, according to Google Scholar. Publication D is an invited paper based on an extension of publication L, which was considered to be among the best papers published at the venue. Publication E is an invited book chapter, presumably an acknowledgement of the contributions already made. Publication I is an extended journal paper based on publication H.

The engineering implementations of the autonomous truck architectures used for the GCDC 2011 and CoAct 2012 are likely to be the basis for the architectures used in the upcoming GCDC 2016 challenge. The engineering implementations of the autonomous architecture for the research concept vehicle will be used to further develop a commercial product by a private company. It will also be the basis for upcoming autonomous driving capabilities of the prototype at KTH.

The following outcomes came about as a result of work done during the thesis, which are also an indirect indicator of its impact

- 3+1 journal papers, 1 book chapter, 1 conference paper, 3+1 workshop papers, 2 technical reports, 1 licentiate thesis
- 4 significant vehicle prototypes created

- The KTH team finished fourth in the Grand Cooperative Driving Challenge, The Netherlands, 2011
- A follow-up 3 year project named ARCHER, driven by Scania CV AB, funded by VINNOVA and engaging 3 PhD candidates
- 3 invited talks to large industrial conferences on autonomous driving: Elektronik i Fordon (2012)/Sweden, Automotive Tech.AD (2015)/Berlin, and Automotive Tech.AD (2015)/Detroit
- 3 large workshops on autonomous systems and safety conducted at KTH, with participation of 50+ industrial researchers and practitioners
- Invited member of Swedish delegation, to represent autonomous systems research in Sweden, as part of Swedish-Brazilian governmental cooperation on research and technology, São Paulo, Brazil 2014
- Invited lectures on systems engineering and systems prototyping at 2 European summer schools on Cyber-Physical Systems (Trento, Italy 2014 and Stockholm, Sweden 2015)
- 2 private companies founded in Sweden. Significant technology transfer made to an autonomous driving related company in "Silicon Valley", California, USA
- 14 Master thesis students supervised
- Laboratory exercises for 3 courses developed for Embedded Systems education

2 READING GUIDE AND CONTRIBUTIONS

> "A method of solution is perfect if we can foresee from the start, and even prove, that following that method we shall attain our aim."
> — Gottfried Wilhelm von Leibniz

This chapter provides a reading guide to the appended papers, as well as an executive summary of their main content.

A quick word on notation: [A] means publication A. [A.2] means Section 2 of publication A. [B.3.1] means Section 3.1 of publication B, and so on.

2.1 READING GUIDE — PUBLICATIONS

Each appended publication is complete in itself, and therefore can be read independently of the others.

- For a thorough reading, it is recommended to read the [A,B,C,D] in the order in which they are appended. This way, the reader will first gain an understanding of the main obstacles to autonomy, as seen from an industrial perspective [A]. This is followed by a description of the specific solution, including architecture, used during the Grand Cooperative Driving Challenge, 2011 [B]. A generalization of the architectural solution in the form of a reference architecture for cooperative driving is given in [C]. A further development into an architecture for general autonomous driving is given in [D].
- If there is time to read only one publication, that should be [E]. It presents the main concepts from [D] and also adds other non-architectural perspectives.
- [F] is specific to a braking system for heavy trucks and is not generalized to the overall vehicle architecture level. Therefore, it is recommended reading for those interested in this specific area.
- Those interested in technological detail related to a specific instantiation of concepts in [B,C] should look up [M]

2.2 FRAME OF REFERENCE

This section briefly explains the relevance of key research areas contributing to autonomous driving. This is then followed by a concise reading guide to the various state of the art surveys included in the thesis work.

The systems architecture for automated driving needs to consider the characteristics of a multitude of domains with regards to technical requirements, the kind of software and other tools utilized, development processes, and even the typical ways in which the domain practitioners think. It is quite rare that the statement, "Oh that.. yeah, that has no architectural impact whatsoever." can be made regarding any stakeholder concern.

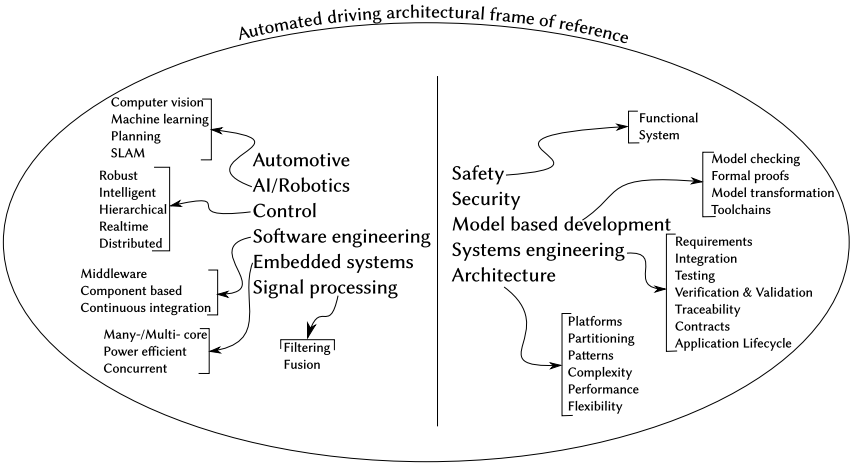


Figure 6: Frame of reference for automated driving architectures

Figure 6 shows just a few of the research areas (and sub-areas) that contribute to architectures for highly automated driving. The areas to the left of the vertical line are the more "fundamental" functional and technical areas, while those to the right of the vertical line apply more to the cross-cutting aspects of the system and its development. The architecture serves as the unifying framework and therefore a competent systems architect needs to have more than a passing familiarity with each domain.

Automotive development is of course the foundation which the rest of automated driving is built upon. This is because, regardless of what systems exist to perceive the environment and make motion decisions, there always needs to be an underlying vehicle platform responsible for executing the desired vehicle motion. The successful

development of core motion systems like steering, propulsion, and braking requires significant know-how from the automotive domain.

Functions for environmental perception, localization, and path planning are based on techniques evolved in the Robotics and Artificial Intelligence areas. This especially applies to the topics of computer vision, extraction of features from sensor data, classification and learning of objects, and planning collision-free trajectories in realtime. Associated algorithms are often probabilistic and may not guarantee results within a given time window. This has an impact on the architecture and analyses related to safety and certification, which are conventionally targeted towards heavily deterministic behavior. Similarly, the computation silicon and typical technological platforms used for robotics and AI research differ from the conventional automotive ECU hardware and software, in terms of available computational capacity and software tools used. This impacts the technical architecture of the automated driving system.

The ever increasing usage of sophisticated electronic control for basic functions like braking, as well as relatively higher level trajectory tracking requires strong contributions from the control engineering domain. There are a large number of modes and states of various systems and their subsystems, the hierarchical and distributed control structures need to respond with correct and safe behavior if and when faults occur, signal uncertainty increases, degraded system modes are entered etc.

Most automated driving functionality is realized through software, and therefore software engineering plays an important role in the system design and development. From the design perspective, topics related to middleware, platform abstraction, and component based software engineering are becoming increasingly relevant, as evidenced by the growing adoption of AUTOSAR. Even for the prototyping code used in the "higher level" functionality related to environmental perception and decision making, technologies like the Robot Operating System (ROS) are commonly utilized, which facilitate reusable software components and inter-component communication. On the development side, topics like agile development and continuous integration are important, because of requirements of faster time-to-market, and delivery of updates and bug fixes to deployed software code.

The inclusion of safety and systems engineering hopefully needs no justification. However, Model Based System Engineering (MBSE) is an important, and sometimes overlooked, part of systems architecting. In MBSE, the different artifacts generated during the systems development process are represented in the form of models, many of which are inter-linked. The architectures can be modeled, analyzed, and successively enriched and it may be possible to generate working system implementations purely via model transformation methods. This is sometimes the case for purely control related tasks, where

TOPIC	PUBLICATION
Automotive EE architecture	G.2
Robotics and intelligent control	G.3
General Embedded Systems	G.4
Discussion: Automotive vs. other domains	G.5
Vehicle platooning and its benefits	B.1.a
Cooperative driving	C.1.3
Architectures for autonomous driving	D.4.3
MBSE methods and tools	E.1.7, E.1.7
Machine consciousness	E.1.4

Table 2: State of the art surveys

the tasks can be modeled and simulated in a tool like Simulink, and the models can be executed using rapid control prototyping tools like the dSpace MicroAutoBox. The models may be annotated with requirements like safety, and specific system properties can be analyzed within Simulink itself, or via the use of add-on tools. Often, specific tools work only with a given representation, and so MBSE has the concept of "round trip" flows, wherein (a part of) the system can be modeled in one tool, the model undergoes transformation to another representation which is analyzed and/or enriched via some other tool, and then subsequently transformed back to the original representation. When such models and tools are also used for generating implementation code, it can impose very strong constraints on the implementation technologies. This has a direct bearing on the technical architecture.

The architectural decisions need to consider not just the technical characteristics of the various domains involved, but also the mindsets of experts within these domains. It is sometimes the case that individuals with deep expertise with one set of tools and concepts in a domain are unable to appreciate why experts in other domains prefer a different set of tools.

In addition to the areas shown in Figure 6, an architect also needs to keep regulatory and certification concerns in mind. MBSE techniques can facilitate dealing with such concerns, by generating documentation trails, proving safety properties, and analyzing the impact of system changes.

A mapping of the state of the art surveys conducted during the thesis work and the publications which contain them is shown in Table 2. The general state of the art on the technological areas contributing to autonomous driving is provided in [G]. It considers autonomous driving as the intersection of the three main areas: (i) Automotive

(ii) Robotics and Intelligent control and (iii) General Embedded Systems. The individual coverage of each of these three areas [G.2, G.3, G.4] is followed by a brief discussion highlighting their relevance and importance to autonomous driving. This is followed by a longer discussion [G.5] comparing architectural considerations for automobiles and robotics, since autonomy in the former is heavily influenced by the latter. This publication was prepared mid-way during the thesis work. The areas considered remain key to technical development of autonomous driving functionality. However, during latter parts of the thesis work, it became clear that complementary results are also needed from other areas. These are covered in subsequent publications, where relevant to the context.

The concept of vehicle platooning and its benefits are presented [B.1.a]. This includes references to the performance and stability of platooning vehicles, fuel saving potential, and the effects on traffic throughput. Also mentioned are some of the European Union projects contributing to these areas. A more generalized survey of cooperative driving is given in [C.1.3]. This survey covers three areas (i) Impact of intelligent cooperative driving on traffic flow (ii) Selected results from individual technological areas of automatic control, wireless communications, and smart transport infrastructures and (iii) Systems integration efforts relevant for whole systems.

Three representative architectures for autonomous driving are mentioned in [D.4.3], for the purpose of comparison with a proposed reference architecture. The architectures are for Stanford's Junior vehicle which participated in the 2007 DARPA Urban Driving Challenge, the HAVE-IT project architecture, and the architecture of the Mercedes Benz S-class car, Bertha. The architectures were selected because they are representative of the evolution in autonomous driving architectures.

Finally, a brief overview of systems engineering methodologies and tools is provided in [E.1.6] and [E.1.7], while a [E.1.4] presents an overview of concepts and relevant takeaways from the area of computational self-awareness and machine consciousness.

2.3 EXECUTIVE SUMMARY OF CONTRIBUTIONS

This section summarizes the key content and takeaways from the appended publications. It is intended to be a self-contained write up for those readers who may not want to read all the publications. Note that for the sake of convenience, citations present in the actual publications are removed from the summary.

2.3.1 *Architecture challenges for intelligent autonomous machines ([A])*

[A] identifies the main bottlenecks for autonomous systems development, from an industrial perspective. The identified bottlenecks are not exclusive for autonomous systems, but they are especially relevant to this category[1]. The main bottlenecks are

CONSTRUCTING AND MAINTAINING THE WORLD MODEL Intelligent autonomous systems need a model of their internal and external environments (worlds) to correctly reason about action choices and make decisions. At all times, the model needs to reflect the real environment, typically via strong statistical correlations. There are several architectural issues related to the construction, implementation, and maintenance of a world model. Subsystems may require partial world models containing only the information necessary for the subsystem operation. The same information may be needed by several subsystems, but in differing representations. Simultaneously, it may be desirable to have a global, common world model to prevent unnecessary, replicated model building activities within individual subsystems and to have a single source of data. But the subsystems may require the model information with differing Quality of Service (QoS) requirements. Some subsystems may require historical information. Should the subsystem maintain history itself or should it be part of a common global world model? Should the partial world models within individual subsystem be gathered together to form a common global model? Or should relevant parts of the global world model be "projected" into individual subsystems? Should a global world model be *implemented* in a distributed, redundant way? What about security and access control to the model information? Who are the readers, who are the writers, how to manage simultaneous conflicting writes — these are all questions relevant to the system architecture.

USER INTERACTION An autonomous system must necessarily reduce the user engagement required for system operation. If the required user engagement is maintained or increased (with respect to frequency, level of control, scope, etc.), autonomy provides no immediate benefit. At the same time, the system needs to conduct its operations with maximum transparency, so that the user can be aware of the operational context and system behavior. Unfortunately, it is not always clear what 'transparency' implies, nor do established norms exist to achieve it. Automa-

1 This is actually a recurring theme for many research areas. There are few results that are solely and exclusively for autonomous systems alone. However, given the complexity and safety critical nature of the autonomous systems, some results have a magnified importance in context.

tion and autonomy, though intended to enhance human capabilities, often degrade them. Worse still, lack of consensus on user interaction leads to functionally similar, yet subtly different autonomous systems. This makes it problematic to rely on past experience when human operators migrate between similar systems. Relying on faulty automation might easily be framed as over-reliance and misuse, while relying on correct automation that still allows for mistakes might be framed as under-reliance or disuse. From the perspective of autonomous systems, it therefore becomes relevant to explore flexible architectures that allow for different user interaction paradigms while maintaining maximum transparency and a meaningful exposure of error handling strategies such as graceful degradation.

COMPLEXITY AND FEATURE INTERACTION Federated and integrated architectures are meant to decentralize the logical architecture and allow for treating different functionality in isolation. However, autonomy seems to be pushing in the other direction, requiring an increased communication between different subsystems. The result is an increased architectural complexity, where the architecture is required to simultaneously isolate and bring together different parts of the system. This, coupled with the increasing number of (often conflicting) goals within an autonomous system, significantly raises the *cognitive complexity* of the system. Increasing complexity has a direct and negative impact on test case coverage, verification and validation of the system. One of the potential consequences of complexity is *feature interaction*. A feature interaction is said to occur when the operation of a subsystem/feature interferes with the operation of another subsystem/feature leading to unexpected and undesirable system level behavior, for example, simultaneous actuation of the brake and throttle during driving. From an architectural perspective, feature interaction can be addressed by two complementary approaches. The first approach is 'correctness by construction' which refers to the principles and mechanisms of composing systems out of subsystems in a way that there are no unexpected side effects or emergent behavior. The complementing approach is to formally represent both the architecture and a feature interaction and use model checking and verification methods to search for the feature interaction in the architecture. Once found, the interaction can be eliminated by a variety of problem specific approaches. Both the approaches above seek to detect and eliminate feature interactions during the design phase. However, all feature interactions may not be detected or eliminated and therefore intelligent autonomous systems need mechanisms to resolve feature interactions during system operation (a.k.a 'runtime').

EXTRA-FUNCTIONAL PROPERTIES This relates to those characteristics of the system which are not directly related to the functionality offered by the system to its user. They are often related to the quality attributes of the system. The extra-functional properties on which autonomy has a significant impact are (i) Safety (ii) Redundancy (iii) Determinism/Predictability. Safety engineers frequently rely on the availability of a human operator to take control, in case the system is unable to cope with its operational environment. The user interaction problems associated with autonomy, along with the fact that a human may simply not be available to take control, means that autonomous systems need to be designed to a greater degree of robustness. Such robustness is often added by means of providing redundancy for critical sensors, actuators as well as communication and computation facilities. Adding redundancy to existing systems brings up sensitive issues related to cost. Also, existing machinery often simply lacks the physical space to accommodate redundant units (geometry constraints). Therefore, application of traditional redundancy measures is often not feasible for evolving autonomous architectures. New thinking is needed to add redundancy to the architecture while avoiding mere replication of the subsystems under question. Safety critical systems are usually required to demonstrate deterministic operation. For autonomous systems, it is a matter of debate and semantics whether autonomy involves a tradeoff with determinism. The overall behavior of an autonomous system may remain predictable within certain bounds, but the system may not be absolutely deterministic within those bounds. For example, it could be predictably shown that the system will generate collision free trajectories on a driveable area, but the precise trajectories generated may not be predictable. If the behavior of an individual machine can not be deterministically predicted, it is unlikely that the interactions of that machine with a heterogeneous mix of other autonomous, manual and remotely operated machines will be deterministic. This leaves designers and architects with an open question: What is the extent of permissible unpredictability within the system? This question is not explicitly answered by standards and certification requirements.

It is heavily debated whether the existing safety standards (like IEC 61508, ISO 26262, or ISO 13849-1) are immediately applicable to autonomous systems. The main reason for this is that they rely on techniques for hazard and risk analysis in which human involvement is an important factor in the Risk Reduction Level (RRL) estimation. ISO2626 for instance directly uses the anticipated capability of a driver to control unexpected vehicle behavior caused by major system failure as a major input to calculating the required RRL (ASIL) on the

system. This reasoning is not immediately applicable for autonomous and intelligent functions because of the lack of human involvement. However, one way by which a standard like ISO26262 can be applied to autonomous driving, is to reduce the controllability to the worst case value for a given hazard. This then leads to higher ASIL levels for the hazard under consideration.

2.3.2 *Platooning and cooperative driving ([B] and [C])*

This section covers the specific architecture used during the Grand Cooperative Driving Challenge (GCDC) 2011 [B], and the subsequent generalization to a reference architecture for cooperative driving [C]. The team that participated in the GCDC was named "Scoop" and hence, the architecture used during the GCDC is referred to as the "Scoop architecture" in [B].

The Scoop architecture comprises of a distinct set of sensors, computation hardware, software, and communication components installed on top of a factory standard truck. These additional components communicate with the factory standard motion subsystems in the truck, such as the engine, brakes, and the transmission. The main requirements that the architecture fulfilled were

- Separation of the functionality into self-contained logical units, which could be designed, developed, and tested independently of each other by different teams
- Enabling of run-time reconfiguration of inter-component communication (change sources/sinks), in order to generate different system behaviors. For example, it should be possible to route an existing data-flow between components $A \rightarrow B \rightarrow C$ to $A \rightarrow C$, in case component B malfunctions.
- Permitting different algorithms to be swapped in and out of the different architectural components
- Enabling diagnostics and self-monitoring services for each architectural component
- Allowing changes to the running system. Changes include reconfiguration of component properties and calibration parameters within executing components
- Being minimally intrusive in the existing vehicle architecture

The top level logical functions in the architecture are shown in Figure 7. They include *Information gathering*, *Estimation*, *Control*, *Information broadcast*, and *Supervision*. A description of each function can be found in [B.II.B.2]. The system was implemented using two distinct

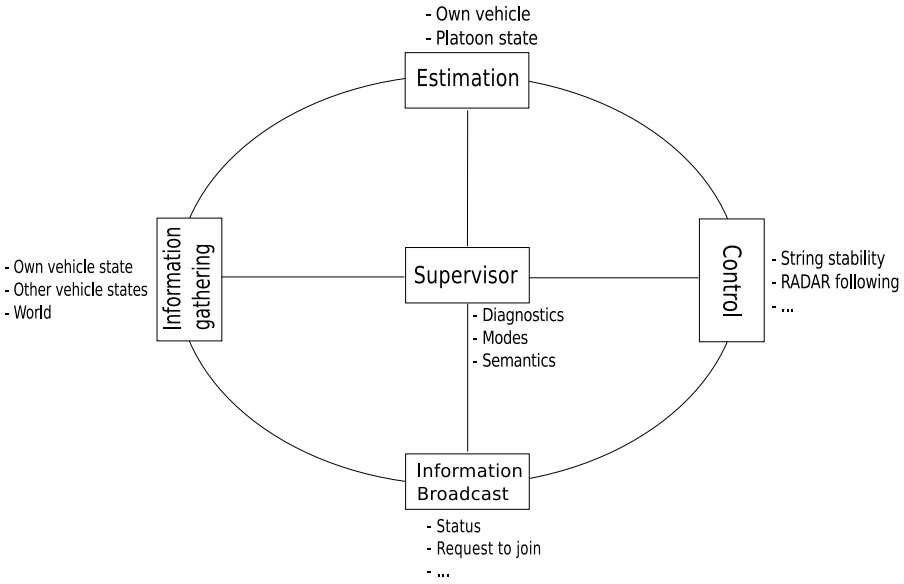


Figure 7: Scoop architecture: Logical functions

computers (i) A custom computer designed using off-the-shelf components with no moving parts and (ii) A re-purposed ECU from Scania. This hardware view is shown in Figure 8 and further described in [B.II.B.3]. The custom computer was based on a dual-core Intel x86 processor architecture (Atom) with a GNU/Linux operating system. The OROCOS realtime middleware was used for constructing software components on this computer. The Scania ECU executed entire Simulink models "at the press of a button" using code generation and a proprietary toolchain. The two computers communicated via a dedicated CAN bus, and each computer was additionally connected to a CAN bus in the truck. The GPS and wireless communication hardware was connected to the custom computer.

The ORCOCS middleware enables creation of components and specification of the components' characteristics such as periodicity of execution, number of input/output ports, datatypes flowing through ports, etc. It also has native support for buffered and unbuffered inter-component communication, as well as for raising and reacting to asynchronous events. Various OROCOS components were created, with close correspondence to the logical functions depicted in Figure 7. The mappings between these software components and the logical functions are shown in the implementation view depicted in Figure 9, where the arrows indicate an "implements" relationship. So for example, the 'Wireless' component in the software layer implements 'Information gathering' and 'Information broadcast' from the functional layer. This is further described in [B.II.B.4]

The Scoop architecture was very successful according to a number of evaluation criteria, the key points of which were

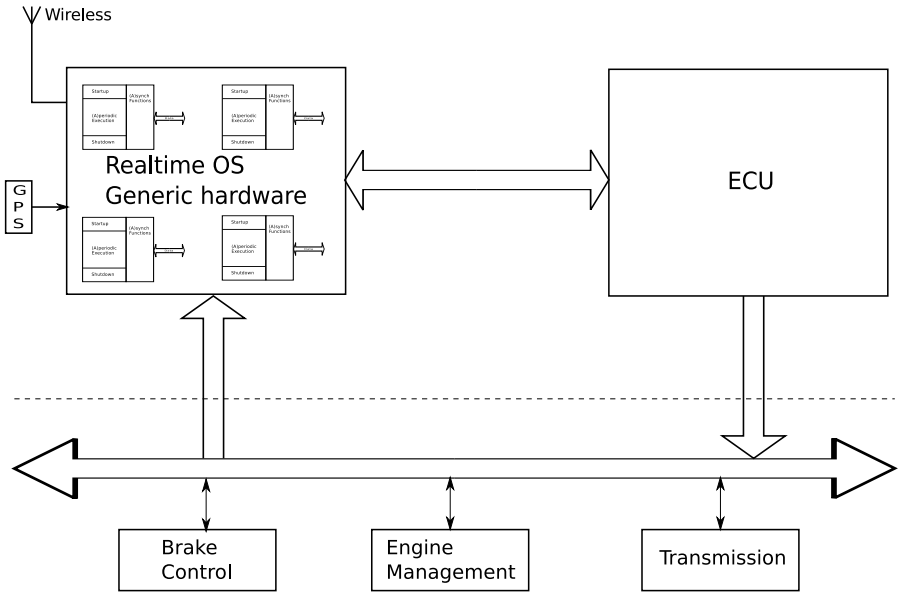


Figure 8: Scoop architecture: Hardware view

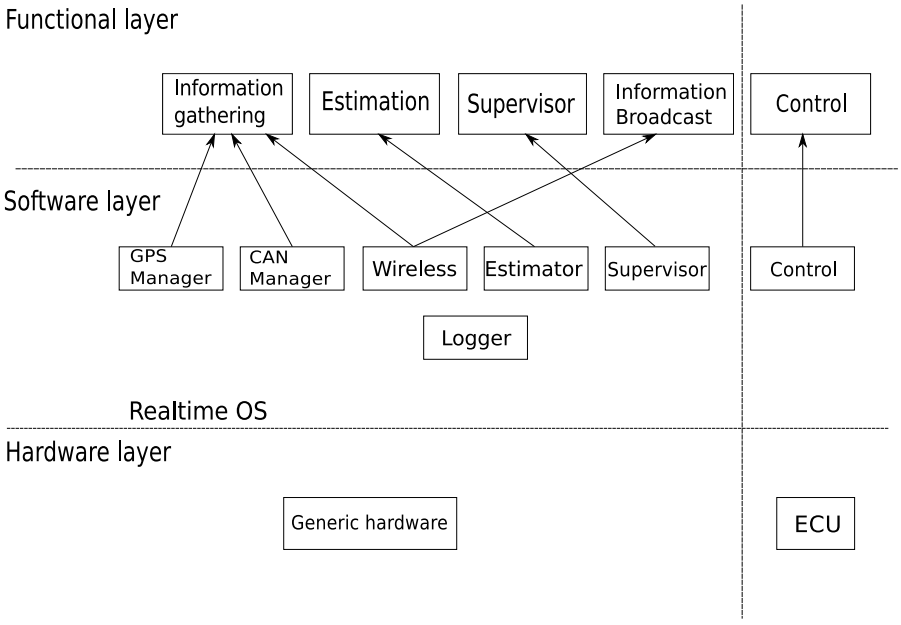


Figure 9: Scoop architecture: Implementation view

- The implementation *worked* at the technical level. No unexpected crashes, segmentation faults, mysterious behavior. The system startup and shutdown was clean, with state being saved, good datalogging and easy possibilities for runtime system calibration.
- The architecture mostly stayed in the background, avoiding intrusion in the function developers' mindspace. They could get their job done without thinking too much about the architecture or using 'quick-and-dirty' hacks to bypass architectural limitations.
- The architecture scored well on aspects of robustness, extensibility, and safety. Unexpected inputs rarely caused the system to get unstable, and behavior modifications were being made to acknowledge changing competition rules up until the day of the event.
- The architecture guided thinking and showed possibilities, rather than constraining the options available to the developers

[B.II.B.5] discusses other topics related to the architecture, such as how specific behavior emerges from the interaction between the architectural components.

Generalizing beyond the Scoop architecture, [C.2.1] describes characteristics of cooperative driving systems, classifying them in functional, extra-functional, and miscellaneous categories. The functional characteristics note the distributed, hierarchical nature of the control task, that requires constructing a world model from limited sensor input (compared to the sensory inputs of a human driver). The extra-functional characteristics emphasize reliability and accuracy of data, as well as the mixture of safety critical and non-critical components within the same platforms. As a precursor to the reference architecture, [C.2.1] briefly discusses the differences between software execution environments in dedicated microcontrollers and general purpose PC hardware, and the associated impact on the architecture.

A reference architecture for cooperative driving is introduced in [C.3], within the context of the ECU network found in existing vehicle E/E architectures. The services needed in a cooperative driving system are identified, and they include positioning, communication, vehicle interfaces, Human Machine Interface (HMI), among others. The reference architecture shown in Figure 10 is then described. First, the key properties common to all architectural elements are mentioned, followed by a more detailed description of the individual functional elements.

[C.4] provides guidelines for instantiation of the reference architecture. The guidelines cover the minimum data sets needed for an instantiation to function as well as considerations for implementing

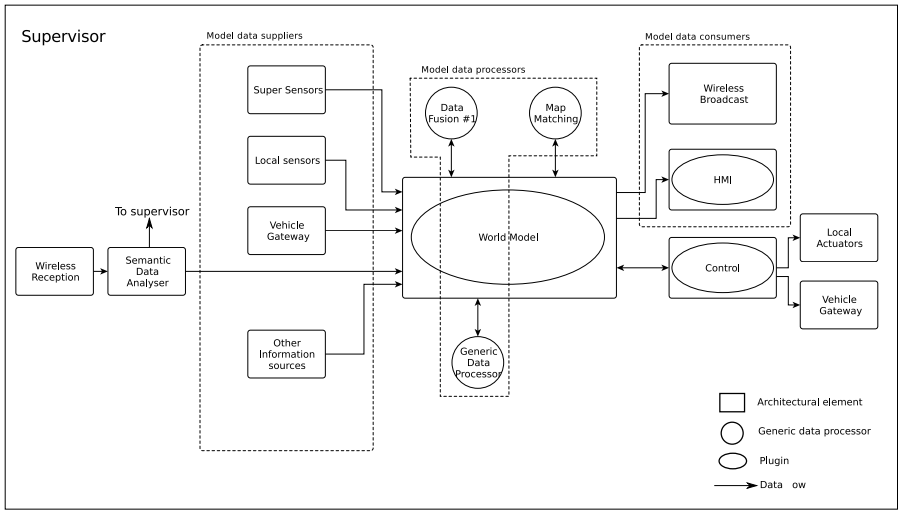


Figure 10: A reference architecture for cooperative driving

each of the elements in the reference architecture. These considerations include computational capacity, execution environments, vendor solutions, and testing/calibration needs. [C.4.2.2] digs deeper into communication mechanisms and covers bandwidth requirements, Quality of Service (QoS) concerns, and communication patterns such as publish/subscribe.

[C.6.2.1] gives some examples of how the reference architecture works by considering four different scenarios. These scenarios are (i) Controlling vehicle motion in a platoon (ii) Approaching a platoon from behind and joining in (iii) GPS signal loss and (iv) An external demand for imposition of radio silence is received. For each of these scenarios, it is described how the various architecture elements interact with each other, and which data flows between them. A comparison of the reference architecture with the industry standard AUTOSAR framework is made in [C.6.3], and with other autonomous system architectures in [C.6.4]. The comparisons show that the reference architecture does not make radical departures from established principles of autonomous system architectures, but there are novel aspects related to how it adapts to existing vehicle architectures in a minimally invasive way, by becoming another "node on a bus".

The reference architecture was instantiated a second time during the CoAct 2012 project. In this instantiation, the Scania ECU shown in Figure 8 was replaced by a SpeedGoat xPC Target controller, which uses a rather different technological implementation stack, but is functionally identical in the sense that it executes a Simulink model "at the push of a button". The mapping shown in Figure 9 was also changed. Specifically, the Estimator function was moved to the SpeedGoat unit, along with the GPS Manager. Of course, the GPS hardware was then connected to the SpeedGoat. These changes were made in view of the fact that the CoAct project team did not have as many competent C++

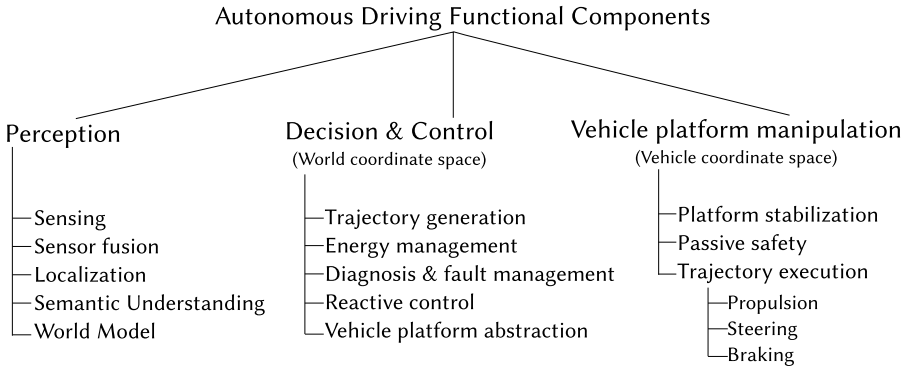


Figure 11: Main functional components for autonomous driving

programmers as the GCDC/Scoop team. Therefore, it was considered easier to design most of the functionality in Simulink, which requires less programming skills. While it would still be possible to generate C++ code from Simulink and execute it on the custom computer, it was far easier to do so on the SpeedGoat, because of all the instrumentation and tools provided by the MathWorks for their xPC product. A consequence of these changes was an increase in the required communication bandwidth between the custom computer and the SpeedGoat, compared to the Scoop architecture. This was addressed by changing the dedicated link between the computers from a CAN bus to 100Mbit/s Ethernet using the User Datagram Protocol (UDP). This second instantiation also incorporated algorithms that provided additional functionality related to lateral maneuvers and overtaking in platoons. This instantiation was also deemed to be entirely satisfactory.

2.3.3 *Autonomous driving ([D] and [E])*

A three layer functional reference architecture for autonomous driving is described in [D]. First, the functional components are described, followed by reasoning on how the components can be distributed across the architectural layers. The architecture description itself consists of identifying the principal stakeholder concerns, the connections among the components and a comparison with similar architectures to highlight similarities and differences.

The main functional components are shown in Figure 11 and described in [D.2]. The components are grouped into three categories relating to (i) Perception (ii) Decision and control and (iii) Vehicle platform manipulation. The Perception components include sensing, fusion, localization, semantic understanding and the world model. These components are responsible for gathering data about the immediate operational environment, and interpreting this data to form a consistent view of the world. The world view includes knowing the position of the vehicle with respect to a map, static road features

(lane markings, guard rails etc.), the location, size, and motion of moving objects and their classification (car, truck, bicycle etc.) The decision and control components are responsible for generating a desired, collision-free motion trajectory among obstacles, keeping in mind concerns of energy management, vehicle state with respect to faults and errors, as well as requirements of good road usage and legal limits. The generated trajectory is usually in 'world coordinate space' i.e. it defines how the vehicle should move in the physical world with respect to the map coordinate system. The trajectory may be represented as a set of accelerations and velocities which is passed on to the vehicle platform. The vehicle platform includes various actuation components responsible for actual vehicle motion (propulsion, braking, steering), and passive safety of the platform. These components generate the actuator specific commands that will realize the desired trajectory.

This architecture is an evolution of the cooperative driving architecture which is discussed in the previous section, and shown in Figure 10. Components like the vehicle gateway, semantic understanding, and the world model have been retained. Other functions in the cooperative driving architecture, like localization and control have been expanded upon in the autonomous driving architecture. A notable difference between these architectures is the role of the world model in the overall data-flow. In the cooperative driving architecture, the world model is at the center, and it can not be bypassed. All data flows into and out of the world model, which becomes the central repository of all information in the system. This imposes high performance and reliability requirements on the world model. Given the scope of cooperative driving, these were technically achievable, especially since the latency introduced by the world model was considered to be acceptable for the application. This has not been the case for the autonomous driving architecture, which sees considerably larger data throughput to/from the world model. Therefore, this architecture supports direct data transfer between various components, as well as via the world model. During instantiation, the extent to which the world model intervenes in the data flows can be freely determined. This would range between two extremes: one where the world model only passively records data flowing between the different components, and the other where all data flows through the world model, during the journey from producer to consumer.

The autonomous driving architecture is initially split up into two layers[2]: the vehicle platform and a cognitive driving intelligence. [D.3] discusses how the functional components can be distributed across these layers, by considering two extremes and their pros/cons. The

2 Later, a third layer related to teleoperation is added, but this layer is not directly responsible for generating and executing motion requests during normal autonomous driving

recommended functionality distribution is based on achieving as clean a split as possible, between the driving intelligence and the vehicle platform. This lowers the cognitive complexity of the architecture, as well as reduces the potential for feature interaction and other undesirable emergent behavior, for example by clearly delineating the tasks of trajectory generation and its execution. It also enables better reuse of the driving intelligence and the vehicle platform in other projects. Thus, the autonomous vehicle is really considered to be a close cooperation between two distinct autonomous entities, which are the vehicle platform and the cognitive driving intelligence. Of these, the former is relatively less intelligent, and responsible for maintaining its own safety to the extent of not violating laws of physics and keeping the vehicle within its safe performance envelope, regardless of the motion requested. The latter is more intelligent and capable of taking decisions under uncertainty and reasoning on its perception of the operational situation. This split is not unlike that of a human being (the cognitive driving intelligence) riding a horse (which is arguably autonomous, with significantly reduced intelligence compared to the human rider).

An architecture balances stakeholder concerns. Therefore, the architecture description first identifies the main stakeholder concerns for a functional architecture. These concerns are grouped into two categories: Business and Engineering. The business concerns relate to considerations of economics, upgrade paths from existing product architectures, unified architectures for autonomous and non-autonomous product variants, and methods to reduce and accelerate testing, verification and validation. Business concerns also reflect regulatory, standards, and certification aspects as well as the possibilities of post-sale modification to vehicle software. The engineering concerns relate more to the functionality of the individual components, and the ability to develop and test them independently. They cover the technologies for virtualized and simulated vehicle testing and the constrains and characteristics of the actual tools and technologies used to implement the vehicle.

The proposed reference architecture is shown in Figure 12 and described in [D.4.2]. Some of the more interesting data links are described, such as that between the localization and sensor fusion, which can be used for modifying Bayesian priors[3] of how the sensor data is characterized at specific geographical locations (e.g. tunnels). The interface between the cognitive driving intelligence and the vehicle platform consists of at least two trajectories: one which takes the vehicle to its desired destination and the other which takes it to a safe(er) state via open-loop control, in case of a sudden and total failure of the cognitive driving intelligence layer. The reactive control is allocated to

3 The 'expected value' of lidar inputs can be modified, for example, if it is known that a particular location always gives noisy returns.

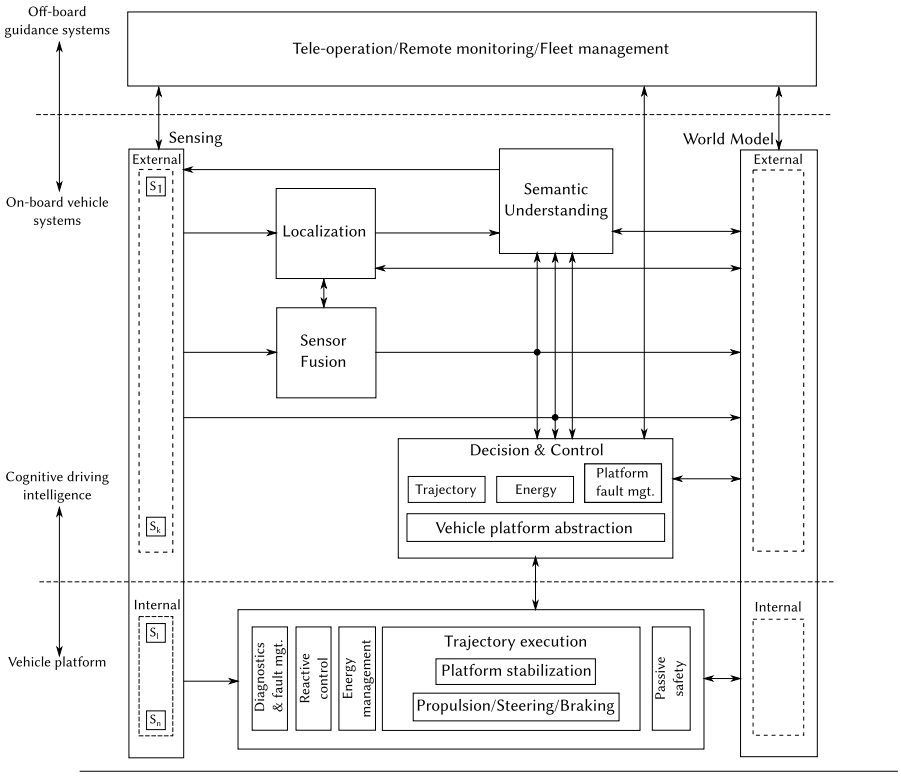


Figure 12: A functional reference architecture for autonomous driving

the vehicle platform, because in our experience, existing technical implementations of the perception and decision and control components have not been fast enough to always deal with unexpected events as part of deliberative control. The presence of the tele-operation layer is also elaborated upon. This layer is chiefly used for gathering telematics data to assess vehicle performance, as well as to manually and remotely guide the vehicle in case it gets stuck in its decision making processes.

A comparison of the proposed reference architecture with other published architectures for (partially) autonomous driving is made in [D.4.3]. The compared architectures are for Junior, Stanford's winning entry in the 2007 DARPA Urban Driving Challenge, the HAVE-IT project's architecture for advanced driver assistance systems, and the Mercedes Benz S-class vehicle that completed a 103 mile autonomous drive from Mannheim to Pforzheim in 2014. These architectures represent a steady improvement of functionality and implementation over the past decade, as well as involvement of academic, Tier 1 and OEM stakeholders. The comparison shows that explicit recognition of semantic understanding, world model and vehicle platform abstraction components are unique to the proposed architecture.

The discussion in [D.5.2] looks at how the functional architecture can be affected by the eventual technical architecture and the characteristics of technologies used in the domains of control engineer-

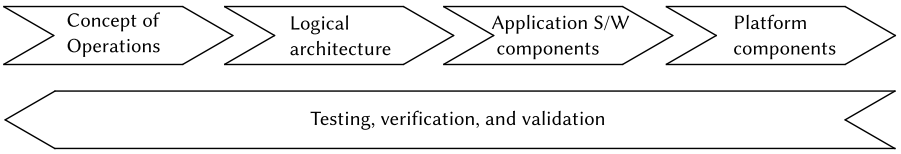


Figure 13: Systems engineering model classes

ing, embedded systems programming, and computer science. This is done by considering the most common tools used in these fields. It is highlighted that the components of the vehicle platform are typically implemented by control engineers and embedded systems programmers, whereas the components of the cognitive driving intelligence are implemented by computer scientists who use general purpose computers during the prototyping phases, and presumably more mature platforms as these systems get closer to series production.

2.3.4 *Model based systems engineering ([E])*

The complex and safety-critical nature of highly automated driving systems increases the need to use systematic systems engineering processes. A model based systems engineering process for autonomous driving systems is outlined in [E.1.4]. The process consists of four steps, each of which is associated with a variety of model classes. The model classes are depicted in Figure 13 and the steps are

1. Describing the system concept of operation
2. Creating a logical system architecture, independent of implementation technologies
3. Mapping the logical system architecture to application software components
4. Implementing the application software components on candidate platforms

Associated with all the four modeling steps is a continuous refinement of requirements, test cases, safety viewpoints, and documentation artifacts. Each step must introduce additional models representing requirements and test cases relevant to that step. Requirements must be allocated to models that assure them, and both requirements and test cases need to be assigned to unique members of the engineering team. Safety considerations are usually incorporated by following processes established by safety standards like ISO26262. This implies that additional models/views like functional safety architecture will be introduced, along with their refinements to technical safety architectures and associated redundancies and switching modes for application software and platform components. The number of models

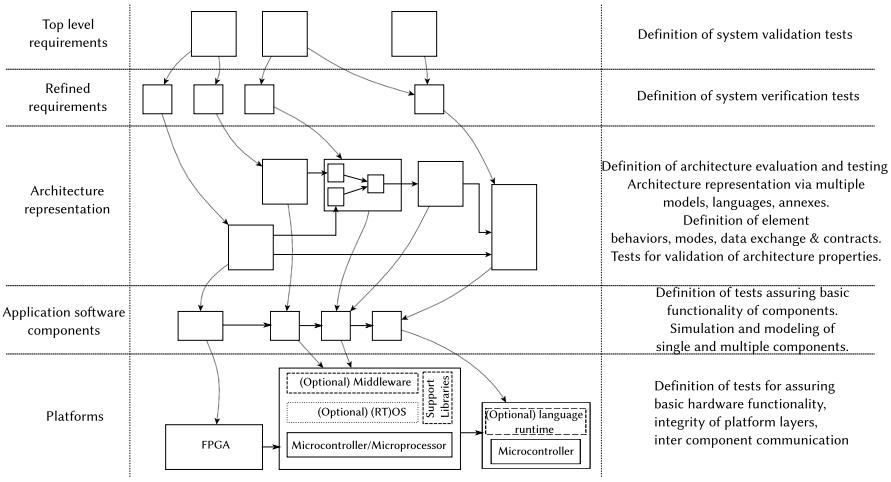


Figure 14: Partial view of modeling artifacts and allocation links

may grow and shrink at each step as the systems engineering process is iteratively applied during system development. Ultimately, the artifacts ideally exist as a web of interconnected models at each step and across the steps. Preserving the links between the models, and keeping the models up to date with the implementation is a significant challenge. One way to do this is via increasing toolchain automation, but given the relative lack of production ready tools, it becomes the responsibility of the architecting and systems engineering team to select and minimize the number of models used to represent the system. This in turn, depends on a variety of technical and non-technical factors like the nature of the project and its maturity level, available tools, the importance attached to systems engineering by project management, the skills and qualifications of the people involved etc. One recommendation based on our experience is to always synchronize the software and platform models with the actual technologies being utilized in the project. These will change as the project moves through stages of proofs-of-concept, prototyping, to certifiable implementations. The models need to be updated correspondingly.

Based on the four modeling steps, a partial view of modeling artifacts and their allocation links is shown in Figure 14, where the rightmost column describes some of the functionality provided by the models. The dotted, curved arrows crossing the vertical layers represent allocation links. Thus arrows between the architecture representation and application software components show the mapping between logical architecture elements and particular application software components. For the sake of clarity, the continuous refinement

and accumulation of requirements and tests are not shown in the Figure.

An overview of key technologies for modeling and implementing autonomous driving systems in given in [E.1.5].

2.3.5 *The context of machine consciousness ([E])*

When thinking about autonomous driving, it is very easy to get lost in practical minutiae of sensors, hardware, programming, modeling etc. However, staying entirely at this level can lead to "not seeing the forest due to all the trees". As the complexity of system architectures rises, it is worth taking a step back and asking, "What is it that we are really trying to do, and is this the right direction?"

The ultimate goal for autonomous driving is not just human-like driving, but to go beyond human-like driving in order to overcome human limitations and appreciably increase road safety, traffic efficiency, and environmental benefits. To achieve this goal, and to interact and coexist with human environments, machines would need a level of consciousness that approaches human (admittedly under tightly constrained notions). This is because consciousness is instrumental to reasoning, decision making, and problem solving capabilities in the face of uncertainty. Indeed, a variety of literature suggests that robots' problem solving capacities would be enhanced by the ability to introspect. This is also a recurring theme across disciplines like computer Systems-on-Chip, programming languages, robotics and even explorations for fault-tolerant on-board computing for robotic space missions. Therefore, it is worthwhile to ask, "What is consciousness?" and "What would consciousness mean in the context of autonomous driving?". This topic is explored in [E.1.6].

From a strict engineering perspective, it is considered that questions regarding the *actual* nature of consciousness are too vague or mysterious to answer. Indeed, as far back as the 1950s, researchers like Alan Turing proposed a *behaviorist* alternative, wherein if a machine can be considered intelligent (and presumably conscious) if it displays behavior that would be considered human-like, by other humans. A loose application of this behaviorist concept to autonomous driving could be: If a savvy human judge can consistently accept a computer's driving abilities as equivalent to those of a competent human driver, then we would have some measure of the computer's driving capabilities. Indeed, on 4th May 2012, one of Google's self-driving Prius vehicles was granted a "driving license" by the Nevada State Department of Motor Vehicles, after the vehicle successfully passed driving tests similar to those administered to human drivers.

In a machine's architecture, the awareness of the external world is usually explicitly represented in the internal world model of the machine, by maintenance of data structures reflecting perception of the

external world. From a philosophical point of view then, awareness of the external world is "absorbed" by the machine's internal states, to which engineering attention must be devoted, in order to achieve progressive results in machine consciousness. However, for meaningful exploitation of any internal awareness, the machine needs to be aware of the awareness i.e. it needs to be *self-aware*. One approach to understanding self-awareness is to understand how the human brain functions, and to mimic biological structures found therein. The strongest approach to understanding human consciousness (and the only one relevant to autonomous driving) is that of *computationalism*, which is the theory that many relevant aspects of the human brain can be modeled as having a computational structure. This approach is the basis of the field of *Artificial Intelligence* (AI) which explores computational models of problem solving. A synthetic summary of the principal propositions of dominant researchers in the field is that *"..consciousness is the property a computational system X has, if X models itself as experiencing things."*. Thus, central to the theory of computational consciousness is that introspection is mediated by models. Such models are explored in the research area of *cognitive architectures*, some of which are mentioned in [E.1.6] along with pointers for further study.

[E.1.6] also discusses a critical review of cognitive architectures that were found in the literature. This review is in terms of applications to real world, safety critical systems and is made along four main themes of (i) Realtime (ii) Resource management (iii) Learning and (iv) Meta-learning. It reveals critical gaps between the design of cognitive architectures and concrete operations in real-world settings. The gaps usually occur because cognitive architectures tend to ignore real-time operation and resource management aspects, which are crucial to useful technological application. It is argued that ignoring these aspects limits the usefulness of the cognitive architectures to only "toy problems" and moreover, can lead to flawed theoretical foundations.

The current approach to autonomous driving capabilities is mostly bottom-up, and based on refinement of existing vehicle functions. The approach relies on careful, manual construction of systems, where learning and decision making takes place only on the data/content or module levels. Such a hand-crafted approach is termed as *Constructionist Design* in the AI domain. It is the opinion of some researchers (see [E.1.6]) that constructionist design has limitations based on the limited ability of humans to hold the entire architecture in their heads and reason about it. These limits are captured in the term *cognitive complexity* in the domain of embedded systems architecture. As the cognitive complexity rises, it becomes increasingly difficult (and costly) to verify and validate system behavior and assure properties like safety. To overcome these problems, a newer *constructivist* approach has emerged in the AI domain, that advocates self-directed, introspective, learning and dynamically adapting conscious architec-

tures. This is considered to be a paradigm shift, whereunder the machine may proactively invoke stimulus-response cycles to continuously form and maintain self-models, and reason about their characteristics. The constructivist approach presents new challenges, especially regarding determinism and predictability of behavior, which run counter to existing requirements of provable safety. Therefore, although this approach may be the eventual way forward, it is unlikely to be adopted for autonomous driving until sufficient tools, methods, and analysis techniques are available for the creation of practical and demonstrably safe systems.

2.3.6 *A brake architecture for autonomous trucks ([F])*

Heavy commercial vehicles, usually referred to as "trucks", are the dominant means of inland freight transport. Trucks are over-represented in fatal road accidents, and hence there is a strong case for making them safer. One way to achieve safety can be by eliminating the human drivers, since human error (typically, lack of attention) is the cause of 90% of truck accidents. Also, human drivers account for approximately 30% of fleet operation costs, which are pure overhead since the task of a truck is to transport goods, not human drivers. Eliminating the human driver implies highest levels of driving automation, which affects many of the vehicle systems. One such system is the brake system, which provides required vehicle deceleration and arrests motion during the truck's operation. It is safety critical, since its malfunction can lead to unacceptable risks during vehicle operation.

The functional brake architecture described in [F] proposes a reconfiguration of the different brakes already found in existing trucks, into a type of redundancy based monitor-actuator pattern. The reconfiguration ensures that a single fault will not result in loss of braking torque, during any operational situation. No additional system components need to be introduced, nor does the physical layout of the braking system need any modification.

The principal requirements considered for the functional architecture are

1. The system *shall* provide sufficient braking torque in all operational situations, including standstill
2. The system *shall not* cause a sudden loss of braking function
3. The system *shall not* require the presence of additional braking systems, beyond those already present in a commercial truck
4. The system *shall* incorporate fail operational behavior to the extent that it is possible to reach a safe state in case of insufficient braking torque due to system failure

5. The system *shall not* increase the probability of a "Vehicle off road" condition i.e. a situation where the mission needs to be aborted, and a secondary vehicle is needed for towing to a workshop

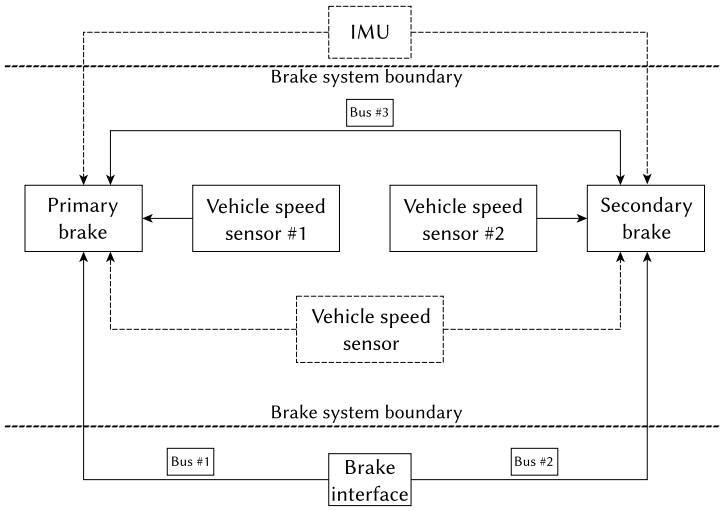


Figure 15: General pattern for brake architecture

The publication focuses on absence of braking torque, and therefore only discusses hazards in the 'Omission', 'Too little', and 'Too late' categories. The general architectural pattern is shown in Figure 15. It consists of a primary and secondary brake system with three different communication buses and distinct vehicle speed sensors for each brake system. It is shown in [F.III.B] that no single failure in such a system can lead to total loss of brake torque. The application of the pattern to the brake systems usually found in existing trucks is shown in Figure 16. The existing brakes are grouped into a category consisting exclusively of the parking brake, and another category consisting of the wheel brakes and other auxiliary retardation systems like exhaust brakes, retarder, electrical machine etc., as available on the truck. The parking brake is mapped onto the secondary brake in the general pattern, while the rest of the brakes are mapped onto the primary brake.

A preliminary reliability analysis [F.III.B] indicates that an order of magnitude improvement in reliability may be expected due to the reconfiguration advocated by the proposed architecture. A simulation study setup to investigate the effect of Omission, Tool Little, and Too Late failures indicates satisfactory braking performance in their presence. For example, the simulation of an Omission failure in the primary brakes (shown in Figure 17) indicates only a small increase in braking distance.

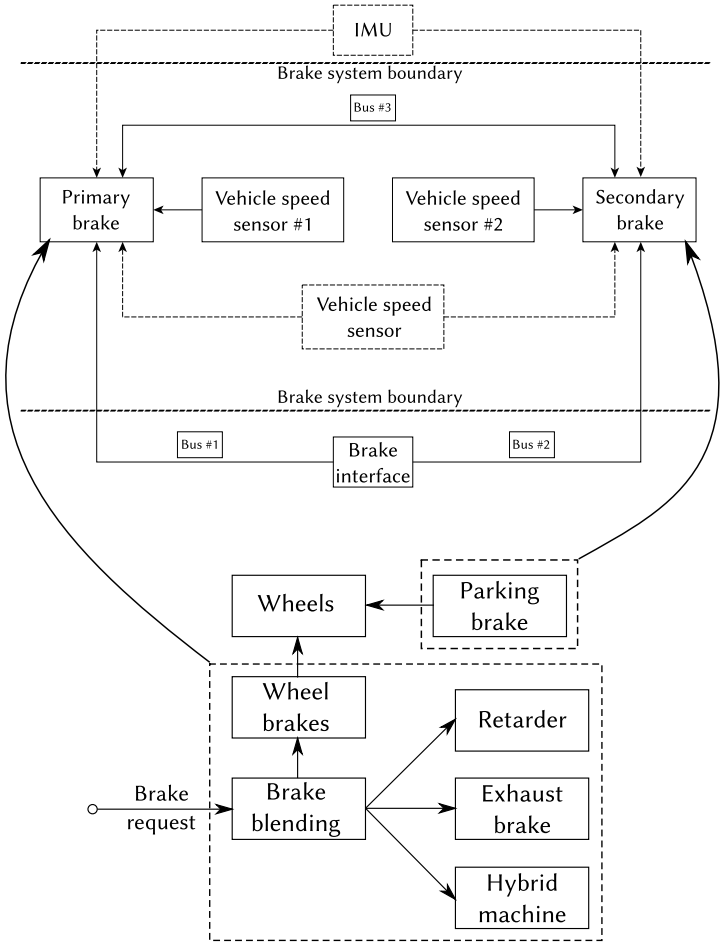


Figure 16: Mapping from existing brake architecture to general pattern

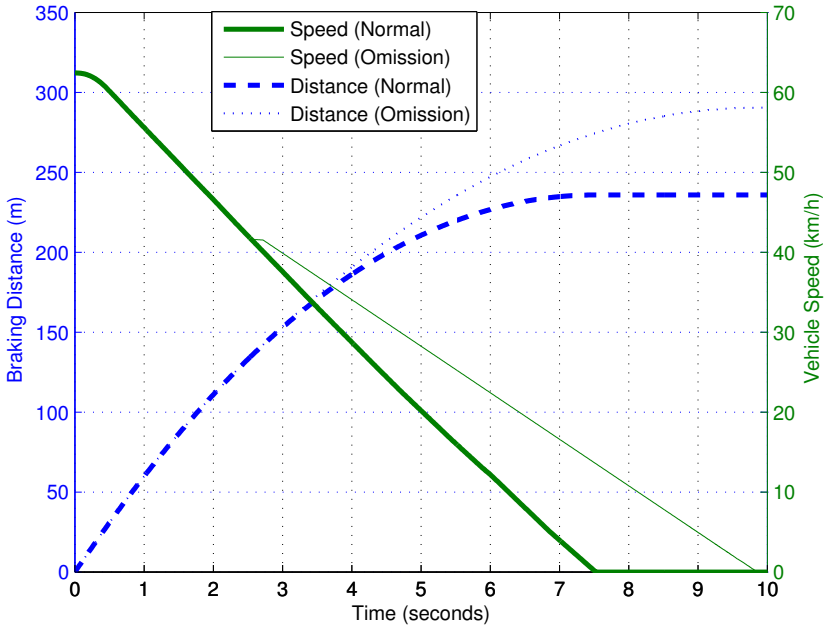


Figure 17: Simulation of 'Omission' condition in primary brake (at 2.6s mark, note no reduction in speed)

3 DISCUSSION

> "We choose to go to the Moon in this decade and do the other things, *not* because they are easy, *but because they are hard*."
>
> — John F. Kennedy, 1962

The hypothesis underlying this thesis work, introduced in Section 1.4 was

> HYPOTHESIS: A reference architecture for highly automated driving exists, which is general enough to be repeatedly instantiated to provide sufficient solutions for a variety of automated driving tasks in a variety of scenarios. Such a reference architecture will contain solution patterns and other value additions to help architects build apposite solutions for automated driving, especially with regards to performance and safety.

The reference architecture for cooperative driving has been successfully instantiated twice. In its extended form as a reference architecture for autonomous driving, it has been examined, instantiated, and validated twice. Most elements of the reference architecture, such as the world model, decision and control, vehicle abstraction, and their interconnections have been retained and sometimes, reused, across the instantiations. The algorithms within the elements, as well as the mapping of the elements to execution platform have undergone changes with each instantiation. The guidelines given for instantiation have helped to make implementation choices in real projects. The architectures have been peer reviewed by practicing engineers working in the area of heavy commercial vehicles, as well as passenger cars. The scientific description of the architectures, in the form of publications, has been peer reviewed by other researchers at workshops, and academic journals. All of this has contributed to high confidence in the reference architectures and in the absence of any contrary evidence so far, it is reasonable to claim that the hypothesis is valid.

The research question

> RESEARCH QUESTION: What is a suitable reference architecture for highly automated driving?

is answered by the architecture description and artifacts provided by the combination of academic publications, technical reports and

implemented vehicles. The answer states the main requirements, principles, and architectural patterns. It also outlines the technologies and development processes for implementing highly automated driving systems.

The method of engineering design turns out to be a good choice for this kind research. The task of actually producing a working prototype uncovered endless details that went into strengthening the reference architecture. For example, the splitting up of the world model into two parts, relating to the vehicle platform and the cognitive driving intelligence, instead of a unified model, is a direct result of numerous attempts to create a unified world model with sufficient performance. This turns to be rather difficult with existing technology and almost always puts excessively high computational processing requirements on the computer executing the world model. Similarly, the split between diagnostics and fault management, as well as deliberative and reactive control aspects are heavily influenced by experiences with concrete system implementations. The generalizations from these implementations have found their way into the reference architectures, adding a value that may not have been possible without this approach. The same holds true for the topics of selecting execution environments and communication technologies, which are mentioned as considerations for instantiating the reference architecture.

A criticism that is sometimes leveled at a functional reference architecture is that it lacks the technical details (like choice of implementation technologies) to create a practical implementation. This misses the entire point of the reference architecture. The details are missing precisely because they are implementation specific. The reference architecture is supposed to be a broad and somewhat abstract guideline that can be enhanced with rich practical detail specific to a project's requirements. The details of the implementations we have made are available in most cases, but even in our own experiments, we have come across a typical situation: Technical architectures and implementation details are considered valuable intellectual property and most companies are reluctant to share more than superficial information without strict non-disclosure agreements. Therefore, we do not expect the scientific literature to be flooded with detailed technical architectures anytime soon.

Other domains like robotics and AI have a strong influence on highly automated driving. Most research prototypes borrow heavily not just from mathematical results in these domains, but also function libraries, middleware, data visualization, and analysis tools. These borrowed items do not always meet the robustness, safety, and reliability requirements of the automotive domain. It will therefore be interesting to see whether they are developed further into reusable, automotive grade, open source components, or whether automotive

companies will develop proprietary and custom tools based on the open source technologies. Within Europe, the initiative for Cyber-Physical Systems Engineering (CPSE Labs) [1] acknowledges the need to create open platforms for autonomous systems, but it is unclear to what extent these platforms will be applicable to, and adopted for, highly autonomous driving.

An important consideration for automotive companies is the topic of legacy. It is not enough to prepare a good clean sheet architecture; ways to encapsulate legacy and migration paths from existing architectures to new ones need to be considered. One way to do this is to wrap the existing functionality via a 'gateway', which abstracts how the legacy systems work. This is more or less similar to the approach taken in the Scoop project, where the entire vehicle motion and actuation systems are behind a gateway that exchanges a fixed set of messages. While simple in theory, there are complicating factors that limit this practice. One factor is that due to the way legacy systems have evolved, the allocation of functionality to control units is not always optimal. For example, it is often the case that the cruise control function resides in the engine management system, because this is often the first system to which electronic control is applied. It remains there for legacy reasons, but from the viewpoint of a clean sheet design, the engine management system is not the ideal location for cruise control, especially when other traction management systems are present, or when the cruise control is being evolved to adaptive or cooperative cruise control. Another factor is that legacy systems can not always be directly reused, even though they have all the needed functionality. This is because of extra-functional requirements like reliability and safety. Both these extra-functional requirements are expected to be stricter for autonomous driving, because the societal acceptance of safety incidents triggered by autonomous vehicles is likely to be lower than for human operated vehicles. The existing designs therefore need to be upgraded to meet the more stringent extra-functional requirements. The brake system architecture described in [F] is an example of this type of upgrade.

The matter of constructionist vs constructivist AI architectures, outlined in [E.1.6] is a trickier problem. The constructionist approach of careful, cross-checked, hand-crafted designs has proven essential and successful in the construction of safety critical systems. Such system designs are marked by conservative approaches and favor extreme determinism, predictability, and regulated changes to system behavior during runtime. To replace this approach by a "wilder" scenario where the machine does not include carefully crafted models, but rather builds them during operation, necessarily by a trial-and-error approach sounds unreasonable. Yet, if this is indeed the way forward, due to reasons of complexity management and endowing the machine with a sense of "Self", then there still need to be hybrid architec-

tures where both approaches can be combined into one. But beyond the purely philosophical issues, another problem is the lack of interaction between the communities engaging in the research. Researchers working on machine consciousness or the theory of Mind rarely interact with those working on automotive functional safety, for example. Such interaction is essential to accelerate the rate of progress in automated driving.

In the context of highly automated driving, an interesting question is: How intelligent is intelligent enough? How smart does the car of the future really need to be? While answering this question, it is necessary to consider that an autonomous vehicle is expected to be just one part of a much larger Intelligent Transportation System (ITS). Future urban (and even non-urban) transport scenarios are expected to be inter-connected grids of data streams between vehicles, infrastructure, government agencies, and private companies. Latest cars already have the capability to report operational data to manufacturers, or receive over-the-air upgrades. It is reasonable to expect an increase in such capabilities for different purposes. Such a trend is already evident in aviation, where operational data from the engines, for example, is streamed back to operations and maintenance bases. Patents for active teleoperation of autonomous vehicles have already been issued. Experiments for regulating vehicle speeds at traffic intersections, via direct commands to the vehicle drivetrain, are also being conducted. Thus, the balance and distribution of intelligence between on-board and off-board systems is an interesting area for future work. There are a large number of stakeholders in this area, including automotive companies, governments, regulatory bodies, insurance companies, traffic management authorities, and vehicle owners. It is reasonable to expect that the stakeholders have their own vested interests which will more often than not be at odds with each other. So it will be fascinating to see how the situation evolves within the next 10-15 years. As with many cases, those ahead on the technology curve will be first-to-market and establish the trends, while possibly encountering risks that were improperly assessed.

3.1 FUTURE WORK

This thesis work has considered notions from the relatively abstract areas of self-awareness and consciousness to very concrete implementations of working vehicles. The balance between abstract and concrete has been struck in the form of reusable reference architectures that bring demonstrable value to the field.

The work can be extended along at least five directions

1. Development of methods and tools for safety analysis and assurance. At the moment, the entire burden of ensuring safety is placed on the team instantiating the reference architecture to a

technical architecture. The supporting research needs to include better hazard analysis (current techniques depend on the availability of a human to take control), redundancy and monitoring patterns, and safety contracts and their implementation.

2. Development of supporting Model Based Systems Engineering processes and tools. As a parallel with the concrete engineering work undertaken in this thesis, proposed Architectural Description Languages (ADLs) and analysis techniques need to be complemented by mature tools that can be used by practicing engineers. Similarly, tool adapters and data-interchange links for facilitating "round trip" data flows between tools are necessary to bring MBSE techniques into practice

3. Research for accelerating and reducing the testing, verification, validation and qualification (TVVQ) efforts. Rapid TVVQ is necessary for achieving comprehensive test coverage, deploying new research results, and enabling continuous integration, and fixing post-development problems.

4. Development of reusable, scalable, high maturity, and safety critical platforms for implementing highly automated driving systems. These platforms should ideally synchronize with the MBSE development methods and tools to maximize verification of system properties at the modeling and implementation levels.

5. Closer collaboration between research from AI/computational consciousness and domains of control, embedded systems, and safety so as to create practical methods for enabling self-awareness in machines.

Items 1. and 2. above shall be examined by the follow-up project, ARCHER, in collaboration with Scania CV AB. Item 4. is being examined via the European innovation actions for Cyber-Physical Systems (CPS).

Research in highly automated driving is still in its infancy. This thesis work provides one part of a much larger puzzle.

BIBLIOGRAPHY

[1] Cyber-Physical Systems Engineering Labs. http://www.cpse-labs.eu/. Accessed: 2015-10-02.

[2] IEEE Standard Glossary of Software Engineering Terminology. *IEEE Std 610.12-1990*, pages 1–84, Dec 1990. doi: 10.1109/IEEESTD.1990.101064.

[3] ISO/IEC/IEEE 42010:2011 Systems and software engineering — Architecture description. International Organization for Standardization, 2011.

[4] SAE J3016: Taxonomy and Definitions for Terms Related to On-Road Motor Vehicle Automated Driving Systems. Sae International, 2014.

[5] J. Albus. Outline for a theory of intelligence. *Systems, Man and Cybernetics, IEEE Transactions*, 21(3):473–509, 1991. doi: 10.1109/21.97471.

[6] TLJ Ferris. Engineering design as research. In *Research Methodologies, Innovations and Philosophies in Software Systems Engineering and Information Systems*. IGI Global, February 2012. doi: 10.4018/978-1-4666-0179-6.

[7] Alan R. Hevner, Salvatore T. March, Jinsoo Park, and Sudha Ram. Design science in information systems research. *MIS Quarterly*, 28(1):75–105, March 2004. ISSN 0276-7783.

[8] INCOSE Definition. What is Systems Engineering? http://www.incose.org/AboutSE/WhatIsSE. Accessed: 2015-10-02.

[9] INCOSE SE Vision 2020 (INCOSE-TP-2004-004-02). Model Based Systems Engineering. http://oldsite.incose.org/ProductsPubs/pdf/SEVision2020_20071003_v2_03.pdf, 2007. Accessed: 2015-10-23.

[10] INCOSE Systems Engineering Handbook. What is a System? http://sebokwiki.org/wiki/What_is_a_System%3F. Accessed: 2015-10-23.

[11] Philippe Kruchten. *The Rational Unified Process*. Rational Software White Paper. Addison-Wesley, 2003. ISBN 0321197704.

[12] John McCarthy. What is artificial intelligence. http://www-formal.stanford.edu/jmc/whatisai/whatisai.html. Accessed: 2015-10-11.

[13] A. Meystel. *Autonomous Mobile Robots: Vehicles with Cognitive Control*. Series in automation. World Scientific, 1991. ISBN 9789971500894.

[14] National Highway Traffic Safety Administration (NHTSA). Critical Reasons for Crashes Investigated in the National Motor Vehicle Crash Causation Survey. http://www-nrd.nhtsa.dot.gov/pubs/812115.pdf. Accessed: 2015-10-17.

[15] Oxford dictionaris. Definition of consciousness. http://www.oxforddictionaries.com/us/definition/american_english/consciousness. Accessed: 2015-10-23.

[16] D L Parnas and P C Clements. A rational design process: How and why to fake it. *IEEE Transactions on Software Engineering*, 12 (2):251–257, February 1986. ISSN 0098-5589.

[17] Herbert A. Simon. The architecture of complexity. *Proceedings of the American Philosophical Society*, 106(6):pp. 467–482, 1962. ISSN 0003049X.

[18] L. von Bertalanffy. *General System Theory: Foundations, Development, Applications*. Penguin University Books. G. Braziller, 2003. ISBN 9780807604533.