

Automotive Software Architectures

Miroslaw Staron

Automotive Software Architectures

An Introduction

Second Edition



Springer

Miroslaw Staron
Department of Computer Science
and Engineering
University of Gothenburg
Gothenburg, Sweden

ISBN 978-3-030-65938-7 ISBN 978-3-030-65939-4 (eBook)
<https://doi.org/10.1007/978-3-030-65939-4>

© Springer Nature Switzerland AG 2017, 2021

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

The publisher, the authors, and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, expressed or implied, with respect to the material contained herein or for any errors or omissions that may have been made. The publisher remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

This Springer imprint is published by the registered company Springer Nature Switzerland AG.
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

*To my family – Sylwia, Alexander, Viktoria
and Cornelia*

Foreword

“The fear of error is the error itself.” Famous philosopher G.F.W. Hegel, whose 250th birthday we currently commemorate, underlined the very necessity of innovation and thinking out of the box. Innovation needs guidance but must not be overconstrained. As engineers, we should follow critical rules but also allow error and learn from it—in order to move forward and not administrate the past. This book will provide guidance toward innovative automotive architectures and services—along the lines of Hegel.

Software and IT are the major drivers of modern cars—both literally and from a marketing perspective. Modern vehicles have more than 70 electronic control units (ECUs), with premium cars having more than 100 such embedded computer systems. Some functions, such as engine control or dynamics, are hard real-time functions, with reaction times going down to a few milliseconds. Practically all other functions, such as infotainment, demand at least soft real-time behaviors.

The complexity of automotive systems and services is growing fast. Each automotive area has its own requirements for computation speed, reliability, security, safety, flexibility, and extensibility. Automotive electronic systems map functions such as braking, powertrain, or lighting controls to individual software systems and physical hardware. The resulting complexity has reached a limit that demands an architectural restart (Fig. 1). At the same time, innovative functions such as connectivity with external infrastructures and vehicle-to-vehicle communication demand IT backbone and cloud solutions with service-oriented architectures (SOA).

Software and IT in vehicles and their environments are evolving at a fast pace. Multimodal mobility will connect previously separated domains like cars and public transportation. Mobility-oriented services such as car sharing create completely new ecosystems and business models far away from the classic “buy-your-own-car” approach. Autonomous driving demands highly interactive services with multisensor fusion, vastly different from the currently deployed functionally isolated control units. Connectivity and infotainment have transformed the car into a distributed IT system with cloud access, over-the-air functional upgrades, and high-bandwidth access to map services, media content, other vehicles, and surrounding

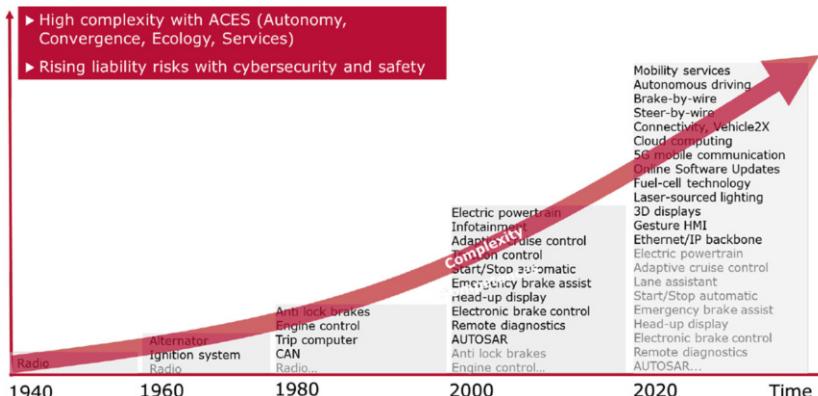


Fig. 1 The convergence of IT and EE fuels automotive technology

infrastructure. Energy efficiency evolves the classic powertrain toward high-voltage hybrid and electric engines.

The major driver in the 2020s is convergence. We face a fast integration of the previously separated concepts of IT and E/E. Software engineering for automotive systems encompasses modern embedded and cloud technologies, distributed computing, real-time systems, mixed safety and security systems, and, not least, the connection of all that to long-term sustainable business models.

Automotive engineers must master both domains, paired with functional safety and cybersecurity. Today automotive software is spearheading IT innovation. The everyday relevance of automotive software to today's software engineers is high, and it is the focus of this book to bring this message to practitioners.

Technology trends are converging across industries (Fig. 2). What used to be a clear-cut differentiation can be summarized today by the quest for ACES, i.e., autonomous systems, convergence, ecology, and services. Business trends are similar in developed and emerging economies. Ten years ago, only 2 out of 10 most valuable public companies by market capitalization were tech companies. Today, almost all are highly driving, and driven by, software technology. Failures to recognize future trends and challenges would be like entering the next decade with all senses closed.

While converging to the new normal, priorities are shifting heavily. Autonomy, until recently still a number one shooting star, has started its slowdown along the hype cycle. At the same time, ecology gets to speed with a high focus especially of the young generation on our future and the sustainability of our earth. Convergence leverages the two forces of competitiveness and innovation toward a sustainable business prospective for technology companies. Services are the major driver. Services are very appealing and we have been talking about them for many years. It follows the Kano model at its best because a good service for a mediocre product can create real excitement. Provide 24/7 online support and you earn a big “wow” if you deliver.

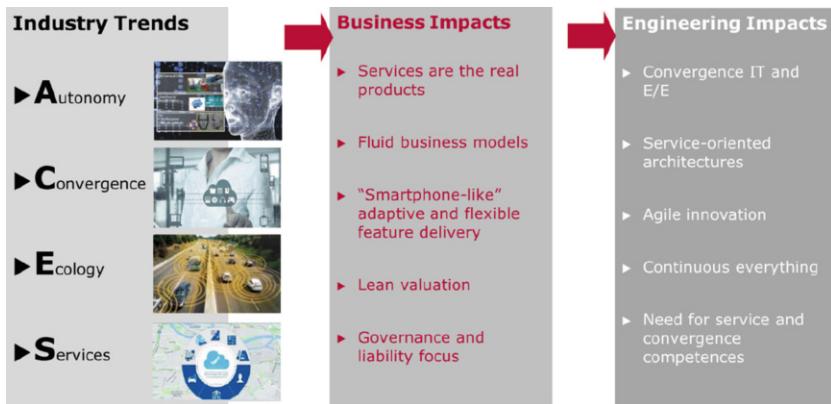


Fig. 2 Prepare for the future: ACES makes digital winners

To master this fast-growing complexity, automotive software needs a clear architecture. Architecture evolution today is the major focus across companies, and thus the book arrives just at the right time. The impacts of architecture are manifold, such as systems modeling, testing, and simulation with models in the loop; the combination of several quality requirements such as safety; service-oriented advanced operating systems with secure communication platforms, such as adaptive AUTOSAR (Automotive Open System Architecture); multisensor fusion and picture recognition for ADAS (advanced driver-assistance systems) and autonomous driving; distributed end-to-end security for flexible remote software updates directly into the car's firmware; and connectivity of cloud technologies and IT backbones with billions of cars and their onboard devices for infotainment, online apps, remote diagnosis, and emergency call processing.

This second edition of the already classic primer on Automotive Software comprehensively introduces to automotive software architecture. Authored by renowned expert Miroslaw Staron, this book provides a guided tour through the methodology and usage of automotive software architecture. Starting with a brief introduction to software architecture paradigms, it quickly moves to current application domains, such as AUTOSAR. Architecture analysis with methods such as ATAM (Architecture Trade-off Analysis Method) of the Software Engineering Institute provides hands-on guidance, keeping in mind the current paradigm shift from classic networking controllers toward the three-tier model of future automotive IT.

Miroslaw Staron with his coauthors target with this book both engineers and decision-makers in the automotive electronics and IT domain. They guide engineers, developers, and managers along the convergence of the two worlds of IT and embedded systems. Education however has only in rare cases dedicated programs for engineering these converging IT and embedded systems. Business models will evolve toward flexible service-oriented architectures and ecosystems. Reference

points based on industry standards such as three-tier cloud architectures, adaptive AUTOSAR, and Ethernet connectivity facilitate reuse across companies and industries. The classic functional split is replaced by a more service-oriented architecture and delivery model. Development in the future will be a continuous process that will fully decouple the rather stable hardware of the car from its functionality driven by software upgrades. Hierarchic modeling of business processes, functionality, and architecture from a systems perspective allows early simulation while ensuring robustness and security. Agile service delivery models combining DevOps, micro-services, and cloud solutions will allow functional changes far beyond the traditional V approach.

The techniques presented in this book are not supposed to be the ultimate truth. Yet they provide direction in this fast-evolving field. It will help you as well as your organization to grow your maturity. Our society and each of us depend on seamless mobility, and so we need to also trust these underlying systems of infrastructure and vehicles. Let us evolve the necessary technology, methods, and competencies in a positive direction to stay in control of automotive software and avoid the many pitfalls of classic IT systems. For this matter, I wish you all the best and success.

As with all architecture independent of application domain, do not forget to deliver value and results to your markets. Your future is based on your competitiveness—both corporate and personal. It is not those to succeed who now shrink engineering and IT innovation, but those who navigate well in the magic triangle of quality, competitiveness, and innovation. Thinker, politician, and novelist Goethe got it straight: “Knowing is not enough; we must apply. Willing is not enough; we must do.” This is the wake-up call to use innovation and guts to stay competitive amidst a meager economic outlook. Business history is littered with the skeletons of those who take neither ownership nor risks.

Stuttgart, Germany
October 2020

Christof Ebert
Managing Director, Vector Consulting Services

Preface

Software is omnipresent in our society. It controls everything from the backbone of electrical infrastructure, to telecommunication equipment to our watches. Cars are no exception, and the amount of software in modern cars is more than in any other consumer product. I was once asked by a colleague at a conference whether the car would still run if we kill the electronic components. The answer was “no” as basically all elements of modern cars are controlled by software—engine, brakes, windshield wipers, blinkers, radio, you name it.

In the last few years, the amount of software in cars has increased as electrification, connectivity, and autonomous drive became more prevalent in all segments. The complexity of scenarios for autonomous driving is so large that cars cannot drive autonomously all the time. Yet, they can drive in various scenarios without changing lanes, and they can change lanes in certain scenarios or even park themselves without anyone in the driver’s seat.

When this complexity grows, we face new challenges in the design of automotive software—more functions become safety critical, more functions interact and communication busses get overcrowded. We need to design the software with that in mind and we need to do it in a new way.

In 2017, we published the first edition of this book, which became popular among students and practitioners alike. Many readers connected with me and asked for certain elements, pointed out to important new developments, and asked questions. I’ve taken these suggestions into consideration and I, once again, managed to convince my colleagues—Dr. Darko Durisic and Dr. Per Johannessen—to help in revising the book.

The purpose of the book is to introduce the concept of software architecture as one of the cornerstones of software in modern cars. The book is a result of my work in the area of software engineering, with a particular focus on safety systems and software measurement. Throughout my research, I have worked with multiple companies in the automotive and telecom domains and I have noticed that over time these domains became increasingly similar. The processes and tools for developing software in modern cars became very similar to those used in the development of telecommunication systems. The same is true about software architectures—

initially very different, today they are increasingly similar in terms of architectural styles, programming paradigms, and architectural patterns.

The book starts with a historical overview of the evolution of software in modern cars and the description of the main challenges which drive the evolution. Chapter 2 describes the main architectural styles of automotive software and their use in cars' software. Chapter 3 is a new addition, where we learn about the modern software architectures—federated and centralized ones. In Chap. 4, the reader can find a description of software development processes used to develop software on the car manufacturer's side. Chapter 5 introduces AUTOSAR—an important standard in automotive software. In this edition, this chapter discusses both the classic and adaptive AUTOSAR. Chapter 6 goes beyond simple architecture and describes the process of detailed design of automotive software with the use of Simulink, which helps us understand how the detailed design links to the high-level design. Chapter 7 is a new one and focuses on machine learning in automotive software development. Chapter 8 presents a method for assessing the quality of the architecture—ATAM (Architecture Trade-off Analysis Method)—and provides an example assessment. Chapter 9 presents an alternative way of assessing the architecture, namely, by using quantitative measures and indicators. In Chap. 10, we dive deeper into one of the specific properties discussed in Chap. 11—safety—and can read about the important standard in that area—ISO/IEC 26262. This time, this chapter contains more information about the hardware than in the first edition of the book. Finally, Chap. 12 presents a set of future trends that seem to emerge today that have the potential to shape automotive software engineering in the coming years.

Gothenburg, Sweden
October 2020

Miroslaw Staron

Acknowledgements

First and foremost, I would like to thank the coauthors of some of the chapters in this book—Darko Durisic, Per Johannessen, and Wilhelm Meding. I have had the privilege of working with them for a number of years and I am deeply thankful for their insights into the car and telecom industries.

I am greatly indebted to my family—Sylwia, Alexander, Viktoria, and Cornelia—who support me in taking on challenges and see to it that I am successful. They are the most fantastic family one could imagine.

I would also like to thank my publisher—Ralf Gerstner from Springer—who has proposed the idea of the book and helped me throughout the process. Without his encouragement and practical pointers, this book would have never happened. After so many years, he still believes in me and provides me with precious advice. I hope that more authors have a chance to be taken care of by such a competent and dedicated publisher.

Many thanks to dSpace GmbH for permitting me to use images of their equipment as part of the book. I also thank Jan Söderberg from Systemite for providing me with figures and explanations on how the SystemWeaver tool keeps the different construction artifacts together. Many thanks go to Volvo Cars, who provided me with several figures in the book.

I am grateful to my colleagues from Volvo Cars who have taught me about practicalities of the automotive industry. I have met many persons from the fantastic team of Volvo Cars and had many great discussions about how cars are designed today, but in particular I am indebted to Kent Niesel, Martin Nilsson, Niklas Baumann, Anders Svensson, Hans Alminger, Ilker Dogan, Lars Rosqvist, Sajed Miremari, Mikael Sjöstrand, and Peter Dahlslund. I would also like to thank Mark Hirche and Malin Folke for their comments on the draft of the book.

I would also like to thank my colleagues from the research community for their help and support in both writing this book and in my research activities leading to this book. In particular, I would like to thank Imed Hammouda for his feedback and comments on the ATAM evaluation chapter.

Finally, I would like to thank the Software Center, Swedish Innovation Agency Vinnova, the Swedish Strategic Research Foundation (SSF), and the Software Center for providing me with research funding that allowed me to pursue my research interests in the area of this book.

Contents

1	Introduction	1
1.1	Software and Modern Cars	1
1.2	History of Software in the Automotive Industry	2
1.3	Trends Shaping Automotive Software Development.....	5
1.4	Organization of Automotive Software Systems	8
1.5	Architecting as a Discipline	9
1.5.1	Architecting vs. Project Management	10
1.5.2	Architecting vs. Design	11
1.6	Content of This Book	12
1.6.1	Chapter 2: Software Architectures	13
1.6.2	Chapter 3: Modern Software Architectures: Federated and Centralized	13
1.6.3	Chapter 4: Automotive Software Development.....	14
1.6.4	Chapter 5: AUTOSAR Reference Model	14
1.6.5	Chapter 6: Detailed Design of Automotive Software....	14
1.6.6	Chapter 7: Machine Learning in Automotive Software	15
1.6.7	Chapter 8: Evaluation of Automotive Software Architectures	15
1.6.8	Chapter 9: Metrics for Software Design and Architectures.....	15
1.6.9	Chapter 10: Functional Safety of Automotive Software	16
1.6.10	Chapter 11: Current Trends in Automotive Software Development	16
1.6.11	Motivating Examples in the Book	16
1.7	Knowledge Prerequisites	17
1.8	Where to Go Next.....	17
	References.....	18

2 Software Architectures—Views and Documentation	19
2.1 Introduction	19
2.2 Common View on Architecture in General and in the Automotive Industry in Particular	20
2.3 Definitions.....	21
2.4 High-Level Structures	22
2.5 Architectural Principles.....	24
2.6 Architecture in the Development Process.....	26
2.7 Architectural Views	27
2.7.1 Functional View	27
2.7.2 Physical System View.....	29
2.7.3 Logical View	31
2.7.4 Relation to the 4+1 View Model.....	31
2.8 Architectural Styles	33
2.8.1 Layered Architecture.....	34
2.8.2 Component-Based.....	37
2.8.3 Monolithic	38
2.8.4 Microkernel.....	39
2.8.5 Pipes and Filters	40
2.8.6 Client–Server	41
2.8.7 Publisher–Subscriber.....	42
2.8.8 Event–Driven	43
2.8.9 Middleware	43
2.8.10 Service-Oriented	44
2.9 Describing the Architectures	46
2.9.1 SysML	46
2.9.2 EAST ADL	48
2.10 Next Steps	50
2.11 Further Reading	50
2.12 Summary	50
References.....	51
3 Contemporary Software Architectures: Federated and Centralized	55
3.1 Introduction	55
3.2 Federated Software Architectures	56
3.3 Centralized Software Architectures	59
3.4 Examples	61
3.4.1 Federated Architecture of a Car	62
3.4.2 Pipes and Filters in Autonomous Drive	63
3.4.3 Infotainment Systems	64
3.5 On Truck Architectures	64
3.6 Summary	65
References.....	65

4 Automotive Software Development	67
4.1 Introduction	67
4.1.1 V-Model of Automotive Software Development	68
4.2 Requirements.....	69
4.2.1 Types of Requirements in Automotive Software Development	71
4.3 Variant Management.....	75
4.3.1 Configuration	76
4.3.2 Compilation	76
4.3.3 Practical Variability Management	78
4.4 Integration Stages of Software Development.....	78
4.5 Testing Strategies	79
4.5.1 Unit Testing.....	79
4.5.2 Component Testing.....	81
4.5.3 System Testing	83
4.5.4 Functional Testing.....	83
4.5.5 Pragmatics of Testing Large Software Systems: Iterative Testing	84
4.6 Construction Database and Its Role in Automotive Software Engineering.....	85
4.7 Further Reading	89
4.7.1 Requirements Specification Languages	91
4.8 Summary	92
References.....	92
5 AUTOSAR (AUTomotive Open System ARchitecture).....	97
5.1 Introduction	97
5.2 AUTOSAR Classic Platform	99
5.2.1 Reference Architecture.....	101
5.2.2 Development Methodology	102
5.2.3 AUTOSAR Meta-Model	108
5.2.4 AUTOSAR ECU Middleware	117
5.3 AUTOSAR Adaptive Platform	119
5.3.1 Reference Architecture.....	122
5.3.2 Development Methodology	123
5.3.3 AUTOSAR Meta-Model	125
5.3.4 AUTOSAR ECU Middleware	130
5.4 AUTOSAR Foundation	130
5.5 Further Reading	131
5.6 Summary	133
References.....	134
6 Detailed Design of Automotive Software	137
6.1 Introduction	137
6.2 Simulink Modelling.....	138
6.2.1 Basics of Simulink	139
6.2.2 Sample Model of Digitalization of a Signal	143

6.2.3	Translating Physical Processes to Simulink.....	146
6.2.4	Sample Model of Car's Interior Heater	149
6.3	Simulink Compared to SysML/UML	155
6.4	Principles of Programming of Embedded Safety-Critical Systems.....	157
6.5	MISRA	158
6.6	NASA's Ten Principles of Safety-Critical Code	160
6.7	Detailed Design of Non-safety-Critical Functionality	161
6.7.1	Infotainment Applications	161
6.8	Quality Assurance of Safety-Critical Software	163
6.8.1	Formal Methods	163
6.8.2	Static Analysis.....	164
6.8.3	Testing	166
6.9	Further Reading	166
6.10	Summary	168
	References.....	168
7	Machine Learning in Automotive Software	171
7.1	Introduction	171
7.2	Fundamentals of Supervised Learning	174
7.3	Neural Networks	176
7.4	Image Recognition Using Convolutional Neural Networks	178
7.5	Object Detection	180
7.6	Reinforced Learning and Parameter Optimization	181
7.7	On-Board and Off-Board Machine Learning Algorithms.....	182
7.8	Challenges with Using Machine Learning in Automotive Software	185
7.9	Summary	186
	References.....	187
8	Evaluation of Automotive Software Architectures.....	189
8.1	Introduction	189
8.2	ISO/IEC 25000 Quality Properties	190
8.2.1	Reliability.....	191
8.2.2	Fault Tolerance	193
8.2.3	Mechanisms to Achieve Reliability and Fault Tolerance	193
8.3	Architecture Evaluation Methods	195
8.4	ATAM	197
8.4.1	Steps of ATAM	197
8.4.2	Scenarios Used in ATAM in Automotive	198
8.4.3	Templates Used in the ATAM Evaluation	201
8.5	Example of Applying ATAM.....	202
8.5.1	Presentation of Business Drivers	203
8.5.2	Presentation of the Architecture	203
8.5.3	Identification of Architectural Approaches	205

8.5.4	Generation of Quality Attribute Tree and Scenario Identification.....	206
8.5.5	Analysis of the Architecture and the Architectural Decision	209
8.5.6	Summary of the Example	210
8.6	Further Reading	211
8.7	Summary	211
	References.....	212
9	Metrics for Software Design and Architectures	215
9.1	Introduction	215
9.2	Measurement Standard in Software Engineering—ISO/IEC 15939	216
9.3	Measures Available in ISO/IEC 25000	218
9.4	Measures	220
9.5	Metrics Portfolio for the Architects	221
9.5.1	Areas	222
9.5.2	Area: Architecture Measures	222
9.5.3	Area: Design Stability.....	224
9.5.4	Area: Technical Debt/Risk	224
9.6	Industrial Measurement Data for Software Designs	226
9.7	Further Reading	228
9.8	Summary	230
	References.....	230
10	Functional Safety of Automotive Software.....	235
10.1	Introduction	235
10.2	Management and Support for Functional Safety	237
10.3	Concept and System Development.....	238
10.4	Planning of Software Development	242
10.5	Software Safety Requirements	244
10.6	Software Architectural Design	244
10.7	Software Unit Design and Implementation	247
10.8	Software Unit Verification	248
10.9	Software Integration and Verification	251
10.10	Testing Embedded Software.....	252
10.11	Examples of Software Design	253
10.12	Integration, Testing, Validation, Assessment and Release	254
10.13	Production and Operation	255
10.14	Further Reading	255
10.15	Conclusions	256
	References.....	256
11	Current Trends in Automotive Software Architectures	259
11.1	Introduction	259
11.2	Autonomous Driving	260

11.3	Self-*	261
11.4	Big Data	262
11.5	New Software Development Paradigms	264
11.5.1	Architecting in the Age of Agile Software Development	265
11.6	Other Trends	266
11.7	Summary	267
	References.....	267
12	Summary	269
12.1	Software Architectures in General and in the Automotive Software—A Short Recap	269
12.2	Chapter 2—Software Architectures.....	269
12.3	Chapter 3—Contemporary Software Architectures: Federated and Centralized	270
12.4	Chapter 4—Automotive Software Engineering	271
12.5	Chapter 5—AUTOSAR	271
12.6	Chapter 6—Detailed Design of Automotive Software	272
12.7	Chapter 7—Machine Learning in Automotive Software.....	272
12.8	Chapter 8—Evaluation of Automotive Software Architectures	272
12.9	Chapter 9—Metrics for Software Designs and Architectures	273
12.10	Chapter 10—Functional Safety of Automotive Software	273
12.11	Chapter 11—Current Trends	274
12.12	Closing Remarks	274