



Automotive Software Engineering

Principles, Processes, Methods, and Tools

List of Chapters:

1. Introduction and Overview

- 1.1 The Driver–Vehicle–Environment System
 - 1.1.1 Design and Method of Operation of Vehicle Electronic Systems
 - 1.1.2 Electronic Systems of the Vehicle and the Environment
- 1.2 Overview of Vehicle Electronic Systems
 - 1.2.1 Electronic Systems of the Powertrain
 - 1.2.1.1 User Interfaces and Setpoint Generators
 - 1.2.1.2 Sensors and Actuators
 - 1.2.1.3 Software Functions
 - 1.2.1.4 Installation Space
 - 1.2.1.5 Variants and Scalability
 - 1.2.2 Electronic Systems of the Chassis
 - 1.2.2.1 User Interfaces and Setpoint Generators
 - 1.2.2.2 Sensors and Actuators
 - 1.2.2.3 Software Functions
 - 1.2.2.4 Installation Space
 - 1.2.2.5 Variants and Scalability
 - 1.2.3 Body Electronics
 - 1.2.3.1 User Interfaces and Setpoint Generators
 - 1.2.3.2 Sensors and Actuators
 - 1.2.3.3 Software Functions
 - 1.2.3.4 Installation Space
 - 1.2.3.5 Variants and Scalability
 - 1.2.4 Multimedia Systems
 - 1.2.5 Distributed and Networked Electronic Systems
 - 1.2.6 Summary and Outlook
- 1.3 Overview of the Logical System Architecture
 - 1.3.1 ECU and Function Networks of the Vehicle
 - 1.3.2 Logical System Architecture for Open-Loop/Closed-Loop Control and Monitoring Systems

1.4 Processes in Vehicle Development

- 1.4.1 Overview of Vehicle Development
- 1.4.2 Overview of the Development of Electronic Systems
 - 1.4.2.1 Trend from Hardware to Software
 - 1.4.2.2 Cost
 - 1.4.2.3 Long Product Life Cycles
 - 1.4.2.4 Safety Requirements—High and Still Rising
- 1.4.3 Core Process for Electronic Systems and Software Development
- 1.4.4 Support Processes for Electronic Systems and Software Development
 - 1.4.4.1 Customer/Supplier Relationships
 - 1.4.4.2 Simultaneous Engineering and Different Development Environments
- 1.4.5 Production and Service of Electronic Systems and Software

1.5 Methods and Tools for the Development of Software for Electronic Systems

- 1.5.1 Model-Based Development
- 1.5.2 Integrated Quality Management
 - 1.5.2.1 Quality Assurance Guidelines
 - 1.5.2.2 Quality Control, Validation, and Verification Measures
- 1.5.3 Reducing the Development Risk
 - 1.5.3.1 Early Validation of Software Functions
 - 1.5.3.2 Reuse of Software Functions

- 1.5.4 Standardization and Automation
 - 1.5.4.1 Standardization
 - 1.5.4.2 Automation
- 1.5.5 Development Steps in the Vehicle

2. Essential System Basics

- 2.1 Open-Loop and Closed-Loop Control Systems
 - 2.1.1 Modeling
 - 2.1.2 Block Diagrams
- 2.2 Discrete Systems
 - 2.2.1 Time-Discrete Systems and Signals
 - 2.2.2 Value-Discrete Systems and Signals
 - 2.2.3 Time- and Value-Discrete Systems and Signals
 - 2.2.4 State Machines
- 2.3 Embedded Systems
 - 2.3.1 Microcontroller Construction
 - 2.3.2 Memory Technologies
 - 2.3.2.1 Read/Write Memory
 - 2.3.2.2 Non-Erasable Read-Only Memory
 - 2.3.2.3 Reprogrammable Nonvolatile Memory
 - 2.3.3 Microcontroller Programming
 - 2.3.3.1 Program Version and Data Version
 - 2.3.3.2 Functional Principles of Microcontrollers
 - 2.3.3.3 Principal Microcontroller Operations
 - 2.3.3.4 Microprocessor Architecture and Instruction Set
 - 2.3.3.5 I/O Module Architecture
- 2.4 Real-Time Systems
 - 2.4.1 Defining Tasks
 - 2.4.2 Defining Real-Time Requirements
 - 2.4.2.1 Instants of Task Activation and Task Deadline
 - 2.4.2.2 Hard and Soft Real-Time Requirements
 - 2.4.2.3 Defining Processes
- 2.4.3 Task States
 - 2.4.3.1 Basic Task State Model (per OSEK-OS)
 - 2.4.3.2 Extended Task State Model (per OSEK-OS)
 - 2.4.3.3 Task State Model (per OSEK-TIME)
- 2.4.4 Strategies for Processor Scheduling
 - 2.4.4.1 Processor Scheduling—In Sequential Order
 - 2.4.4.2 Processor Scheduling—By Priority
 - 2.4.4.3 Processor Scheduling—Combined Sequential and Priority Strategy
 - 2.4.4.4 Processor Scheduling—Preemptive Strategy
 - 2.4.4.5 Processor Scheduling—Nonpreemptive Strategy
 - 2.4.4.6 Processor Scheduling—Event-Driven and Time-Controlled Strategies
- 2.4.5 Organization of Real-Time Operating Systems
- 2.4.6 Interaction Among Tasks
 - 2.4.6.1 Synchronization
 - 2.4.6.2 Cooperation
 - 2.4.6.3 Communication
 - 2.4.6.4 Interaction Among Tasks in the Logical System Architecture
- 2.5 Distributed and Networked Systems
 - 2.5.1 Logical and Technical System Architecture
 - 2.5.2 Defining Logical Communication Links
 - 2.5.2.1 Client/Server Model
 - 2.5.2.2 Producer/Consumer Model
 - 2.5.3 Defining the Technical Network Topology
 - 2.5.3.1 Star Topology
 - 2.5.3.2 Ring Topology
 - 2.5.3.3 Linear Topology
 - 2.5.4 Defining Messages
 - 2.5.4.1 Addressing
 - 2.5.4.2 Communications Matrix

2.5.5	Organization of Communications and Network Management	2.6.5.2	Monitoring Setpoint Generators, Sensors, Actuators, and Control Functions
2.5.5.1	Communications (per OSEK-COM)	2.6.6	Organization of a Diagnostic System for Electronic Control Units
2.5.5.2	Network Management (per OSEK-NM)	2.6.6.1	Offboard Diagnostic Functions
2.5.6	Strategies for Bus Arbitration	2.6.6.2	Onboard Diagnostic Functions
2.5.6.1	Bus Access Strategies—Centralized or Decentralized Implementation	2.6.6.3	Diagnostics for Setpoint Generators and Sensors
2.5.6.2	Bus Access Strategies—Controlled or Random	2.6.6.4	Diagnostics for Actuators
2.5.6.3	Bus Access Strategies—Event-Driven and Time-Controlled	2.6.6.5	Fault Memory Manager
2.6	System Reliability, Safety, Monitoring, and Diagnostics	2.6.6.6	Offboard Diagnostic Communications
2.6.1	Basic Terms	2.6.6.7	Model-Based Fault Recognition
2.6.2	System Reliability and Availability	2.7	Summary
2.6.2.1	Definition of Reliability Function $R(t)$ and Failure Rate $\lambda(t)$	3.	Support Processes for Electronic Systems and Software Engineering
2.6.2.2	Definition of Mean Time to Failure (MTTF)	3.1	Basic Definitions of System Theory
2.6.2.3	Definition of Mean Time to Repair (MTTR)	3.2	Process Models and Standards
2.6.2.4	Definition of Mean Availability	3.3	Configuration Management
2.6.3	System Safety	3.3.1	Product and Life Cycle
2.6.3.1	Definition of Terms in Safety Technology	3.3.2	Variants and Scalability
2.6.3.2	Determining Risk	3.3.3	Versions and Configurations
2.6.4	System Monitoring and Diagnostics	3.4	Project Management
2.6.4.1	Monitoring	3.4.1	Project Planning
2.6.4.2	Fault Recognition and Fault Diagnostics	3.4.1.1	Quality Planning
2.6.4.3	Error Detection and Correction	3.4.1.2	Cost Planning
2.6.4.4	Safety Logic	3.4.1.3	Project Scheduling
2.6.4.5	Functional Software Safety	3.4.1.4	Development Roles and Responsibilities
2.6.5	Organization of a Monitoring System for Electronic Control Units	3.4.2	Project Tracking and Risk Management
2.6.5.1	Microcontroller Monitoring Functions	3.5	Subcontractor Management
		3.5.1	System and Component Responsibilities
		3.5.2	Interfaces for Specification and Integration
		3.5.3	Defining the Cross-Corporation Development Process
		3.6	Requirements Management
		3.6.1	Mining, Recording, and Interpreting User Requirements
		3.6.2	Tracking User Requirements

- 3.7 Quality Assurance
 - 3.7.1 Integration and Testing Procedures
 - 3.7.2 Software Quality Assurance Methods
- 4. Core Process for Electronic Systems and Software Engineering**
 - 4.1 Requirements and Prerequisites
 - 4.1.1 Shared System and Component Responsibilities
 - 4.1.2 Coordination of Systems Engineering and Software Engineering
 - 4.1.3 Model-Based Software Development
 - 4.2 Basic Definitions and Notations
 - 4.2.1 Processes, Process Steps, and Artifacts
 - 4.2.2 Methods and Tools
 - 4.3 Analysis of User Requirements and Specification of Logical System Architecture
 - 4.4 Analysis of Logical System Architecture and Specification of Technical System Architecture
 - 4.4.1 Analysis and Specification of Open-Loop/Closed-Loop Control Systems
 - 4.4.2 Analysis and Specification of Real-Time Systems
 - 4.4.3 Analysis and Specification of Distributed and Networked Systems
 - 4.4.4 Analysis and Specification of Reliable and Safe Systems
 - 4.5 Analysis of Software Requirements and Specification of Software Architecture
 - 4.5.1 Specification of Software Components and Associated Interfaces
 - 4.5.1.1 Specification of Onboard Interfaces
 - 4.5.1.2 Specification of Offboard Interfaces
 - 4.5.2 Specification of Software Layers
 - 4.5.3 Specification of Operating States
 - 4.6 Specification of Software Components
 - 4.6.1 Specification of Data Model
 - 4.6.2 Specification of Behavioral Model
 - 4.6.2.1 Specification of Data Flow
 - 4.6.2.2 Specification of Control Flow
 - 4.6.3 Specification of Real-Time Model
 - 4.6.3.1 State-Dependent Reactive Execution Model
 - 4.6.3.2 State-Independent Reactive Execution Model
 - 4.7 Design and Implementation of Software Components
 - 4.7.1 Consideration of Requested Nonfunctional Product Properties
 - 4.7.1.1 Differentiation Between Program Version and Data Version
 - 4.7.1.2 Limitation of Hardware Resources
 - 4.7.2 Design and Implementation of Data Model
 - 4.7.3 Design and Implementation of Behavioral Model
 - 4.7.4 Design and Implementation of Real-Time Model
 - 4.8 Software Component Testing
 - 4.9 Integration of Software Components
 - 4.9.1 Generating Program Version and Data Version
 - 4.9.2 Generating Description Files
 - 4.9.3 Generating Documentation
 - 4.10 Software Integration Testing
 - 4.11 Integration of System Components
 - 4.11.1 Integration of Software and Hardware
 - 4.11.1.1 Download
 - 4.11.1.2 Flash Programming
 - 4.11.2 Integration of ECUs, Setpoint Generators, Sensors, and Actuators
 - 4.12 System Integration Test
 - 4.13 Calibration
 - 4.14 System and Acceptance Test

5. Methods and Tools for Development

- 5.1 Offboard Interface Between Electronic Control Units and Tools
- 5.2 Analysis of Logical System Architecture and Specification of Technical System Architecture
 - 5.2.1 Analysis and Specification of Open-Loop and Closed-Loop Control Systems
 - 5.2.2 Analysis and Specification of Real-Time Systems
 - 5.2.2.1 Schedulability Analysis
 - 5.2.2.2 Verifying Schedulability by Means of Measurements
 - 5.2.2.3 Monitoring and Handling Deadline Violations in the Operating System
 - 5.2.3 Analysis and Specification of Distributed and Networked Systems
 - 5.2.4 Analysis and Specification of Reliable and Safe Systems
 - 5.2.4.1 Failure Rate Analysis and Calculation of Reliability Function
 - 5.2.4.2 System Safety and Reliability Analysis
- 5.3 Specification of Software Functions and Validation of Specification
 - 5.3.1 Specification of Software Architecture and Software Components
 - 5.3.1.1 Object-Based Software Architecture Modeling
 - 5.3.1.2 Module-Based Specification of Interfaces to Real-Time Operating System
 - 5.3.1.3 Class-Based Specification of Reusable Software Components
 - 5.3.2 Specification of Data Model
 - 5.3.3 Specification of Behavioral Model Using Block Diagrams
 - 5.3.3.1 Specification of Arithmetical Functions
 - 5.3.3.2 Specification of Boolean Functions
 - 5.3.4 Specification of Behavioral Model Using Decision Tables
 - 5.3.5 Specification of Behavioral Model Using State Machines
 - 5.3.5.1 Specifying Flat State Machines
 - 5.3.5.2 Specifying Transitions with Branching Instructions
 - 5.3.5.3 Specifying Hierarchy State Machines
 - 5.3.6 Specification of Behavioral Model Using High-Level Languages
 - 5.3.7 Specification of Real-Time Model
 - 5.3.8 Validating the Specification Through Simulation and Rapid Prototyping
 - 5.3.8.1 Simulation
 - 5.3.8.2 Rapid Prototyping
 - 5.3.8.3 Horizontal and Vertical Prototypes
 - 5.3.8.4 Target System Identical Prototypes
 - 5.3.8.5 Throw-Away and Evolutionary Prototypes
 - 5.3.8.6 Reference Prototype for ECU Verification
- 5.4 Design and Implementation of Software Functions
 - 5.4.1 Consideration of Requested Nonfunctional Product Properties
 - 5.4.1.1 Runtime Optimization Through Consideration of Varying Access Times to Different Memory Segments
 - 5.4.1.2 Runtime Optimization Through Distribution of Software Function to Several Tasks
 - 5.4.1.3 Resource Optimization Through Division into Online and Offline Calculations
 - 5.4.1.4 Resource Optimization Through Division into Onboard and Offboard Calculations
 - 5.4.1.5 Resource Optimization for Characteristic Curves and Maps

- 5.4.2 Design and Implementation of Algorithms for Fixed-Point and Floating-Point Arithmetic
 - 5.4.2.1 Representation of Numbers in Digital Processors
 - 5.4.2.2 Rounding Errors in Integer Division
 - 5.4.2.3 Overflow and Underflow in Addition, Subtraction, and Multiplication
 - 5.4.2.4 Shift Operations
 - 5.4.2.5 Handling Overflows and Underflows
 - 5.4.2.6 Error Propagation with Algorithms in Fixed-Point Arithmetic
 - 5.4.2.7 Physical Interrelation and Fixed-Point Arithmetic
 - 5.4.2.8 Physical Model Level and Implementation Level
 - 5.4.2.9 Notes on Implementation in Fixed-Point Arithmetic
 - 5.4.2.10 Notes on Implementation in Floating-Point Arithmetic
 - 5.4.2.11 Modeling and Implementation Guidelines
- 5.4.3 Design and Implementation of Software Architecture
 - 5.4.3.1 Platform and Application Software
 - 5.4.3.2 Standardization of Platform Software Components
 - 5.4.3.3 Configuration of Standardized Software Components
- 5.4.4 Design and Implementation of Data Model
 - 5.4.4.1 Definition of Memory Segment
 - 5.4.4.2 Setting Data Variants via Flash Programming
 - 5.4.4.3 Setting Data Variants via Configuration Parameters
 - 5.4.4.4 Generation of Data Structures and Description Files
- 5.4.5 Design and Implementation of Behavioral Model
- 5.5 Integration and Testing of Software Functions
 - 5.5.1 Software-in-the-Loop Simulations
 - 5.5.2 Laboratory Vehicles and Test Benches
 - 5.5.2.1 Test Environment for Standalone ECUs
 - 5.5.2.2 Test Environment for ECUs, Setpoint Generators, Sensors, and Actuators
 - 5.5.2.3 Test Environment for ECU Network
 - 5.5.2.4 Test Bench
 - 5.5.3 Experimental, Prototype, and Production Vehicles
 - 5.5.4 Design and Automation of Experiments
- 5.6 Calibration of Software Functions
 - 5.6.1 Offline and Online Calibration Procedures
 - 5.6.2 Software Update Through Flash Programming
 - 5.6.3 Synchronous Measuring of Microcontroller and Instrumentation Signals
 - 5.6.4 Downloading and Evaluating Onboard Diagnostic Data
 - 5.6.5 Offline Calibration of Parameters
 - 5.6.6 Online Calibration of Parameters
 - 5.6.7 Classification of Offboard Interfaces for Online Calibration
 - 5.6.7.1 Serial Preproduction Interface with Internal CAL-RAM (Method 1)
 - 5.6.7.2 Serial Development Interface with Internal CAL-RAM (Method 2)
 - 5.6.7.3 Parallel Development Interface with Internal CAL-RAM (Method 3)

- 5.6.7.4 Serial Preproduction Interface with Additional CAL-RAM (Method 4)
- 5.6.7.5 Serial Development Interface with Additional CAL-RAM (Method 5)
- 5.6.7.6 Parallel Development Interface with Additional CAL-RAM (Method 6)
- 5.6.7.7 Communications Protocols for Calibration Tools and Microcontrollers
- 5.6.8 CAL-RAM Management
 - 5.6.8.1 CAL-RAM Management with Sufficient Memory Resources
 - 5.6.8.2 CAL-RAM Management with Limited Memory Resources
- 5.6.9 Parameter and Data Version Management
 - 5.6.9.1 Binary Program and Data Version File Calibration
 - 5.6.9.2 Model or Source Code Calibration and Optimization
- 5.6.10 Design and Automation of Experiments

6. Methods and Tools for Production and Service

- 6.1 Offboard Diagnostics
- 6.2 Parameterization of Software Functions
- 6.3 Software Update Through Flash Programming
 - 6.3.1 Erasing and Programming Flash Memory
 - 6.3.2 Flash Programming Through the Offboard Diagnostic Interface
 - 6.3.3 Security Requirements
 - 6.3.4 Availability Requirements
 - 6.3.5 Boot Block Shifting and Flash Programming
- 6.4 Startup and Testing of Electronic Systems

7. Summary and Outlook

References

Illustration Credits

List of Acronyms

Index

About the Authors