Index

Note: Page numbers followed by ‘f’ and ‘t’ represent figures and tables respectively.

absolute deadline, 66
abstract logical model, 160
acceptance test, 151, 201–202, 202f
accepted user requirements, 147
accident, system safety, 104
accumulators, 59–60
activation rate, 66
active safety systems, 13
actuators, 3, 218
diagnostics, 116–117, 329
intelligent, 51
sensors and, 9, 10f, 12, 12f, 14, 256
adaptive cruise control (ACC) system, 8, 15–16, 16f, 84
additional programmer method, 294, 295f
addressing, 62, 89–90
aggregation relations, 126
application software, 34
arbitration
bus, 85
diagram, 65, 65f, 73
phase, 95
scheduling strategy, 211
arithmetic functions specification, 237–239, 238f–239f
ASAM-MCD-2D, 327–328
ASCET tool, 238–240, 238f–239f
assembler, 191
assumptions, system theory, 124
atomic operations, 76
attribute, nonpreemptive processor
scheduling strategy, 72
authentication, 334–335
automatic transmission, 9–10
automotive electronics, 1
AUTOSAR (Automotive Open System Architecture), 20, 23, 34–35, 36f, 63–64
availability, 98
basic task state model, 68, 69f
behavioral model
additional programmer method, 294
using block diagrams, 237–240
C language implementation, 295, 296f
using decision tables, 240–242
design and implementation of, 187, 188f
using high-level languages, 247, 248f
integration platform method, 294
“Integrator” class, 295, 295f
model-based specification, 294
software components
Boolean and an arithmetic instruction, 179, 179f, 180, 180f
data flow, 178–179, 179f
control flow, 179–180, 180f
software components plus program and data versions, 294, 295f
specification
deterministic, 245f
differentiations, 244
equivalent actions in, 245f
hierarchy state machines, 246–247, 247f
Mealy state machines, 243
Moore state machines, 243
nondeterministic situations, 244
start state, 244, 244f
state diagrams, 243
transition actions, 243
transitions with branching instructions, 246, 246f
“trigger()” method, 245f, 246
behavioral model (Cont.)
  using state machines, 242–247
body electronics, vehicle electronic systems
  active safety systems, 13
  installation space, 14
  passive safety group, 13
  sensors and actuators, 14
  software functions, 14
  user interfaces and setpoint generators, 13–14
  variants and scalability, 14–15
Boolean functions specification, 239–240
boot block, 332–333, 333
  shifting and flash programming, 337–338, 337
  boot loader, 195
Bosch Sensotronic brake control (SBC), 3, 3f
brake pedal, 3
brake-by-wire system, 225–226, 225f–226f
braking system, 12
bus arbitration, strategies for, 94, 94f
  centralized/decentralized implementation, 94–95
  controlled/random, 95–96
  event-driven and time-controlled, 96, 96f
bus system, 52
bypass systems
  communications, 254
  hooks, 253
  interface, 190, 253
  system, 254f

calibration, 200–201, 201f
  ECUs, 173
  online, 204
  software functions
    CAL-RAM management, 322–325
    measurement signals, 309
    microcontroller and instrumentation signals, 313–314
    offboard interfaces for online calibration, 316–322, 317f
    offline and online calibration procedures, 310–311, 311f
  offline calibration of parameters, 314–315, 315f
  online diagnostic data, downloading and evaluating, 314
  online calibration of parameters, 315–316, 316f
  parameter and data version management, 325–326
  software update through flash programming, 312–313, 312f
  calibration RAM (CAL-RAM), 315–316, 316f–317f
  communications protocols for, 322
  management
    memory resources, 322–325, 323f
    monitoring concept, 322
  parallel development interface
    additional, 321, 321f
    internal, 319, 320f
  serial development interface
    additional, 321, 321f
  serial preproduction interface
    additional, 320, 320f
    internal, 317–319, 318f–319f
  capability maturity model integration (CMMI), 127
  carrier sense multiple access (CSMA) strategy, 95
  characteristic curves and maps, 177, 178f,
    264–268, 265f–268f
  chassis, electronic system
    braking system, 12
    installation space, 12
    sensors and actuators, 12, 12f
    software functions, 12
    user interfaces and setpoint generators, 11
    variants and scalability, 12
  vehicle components, 12
  clock generator, 52
  closed-loop control task, 41
  collision avoidance (CA) strategies, 95
  collision detection (CD) strategies, 95
  communication, 79–81
    latency, 96
  and network management,
    organization of, 91–92, 91f–92f
communication matrix (C-matrix), 90, 90f, 222, 222f
complex device drivers, 174
component test, 150
component variants, 130, 130f
computer-based replication and execution, 249
configuration
    aspects, 133
    management, 133
    product and life cycle, 129, 129f
    variants and scalability, 130, 130f
    versions and configurations, 131–133, 131f–132f
    at system level, 132, 132f
constraints, 161
continual development and change management, 128–129
continuous time and value signals, 46, 46f
data memory, 52–53, 56–57
data model
    configuration parameters, 292, 293f
data structures and description files, 293, 294f
    flash programming, 291–292, 292f
    memory segment, 291
    software components, 177, 178f
    vehicle variants, 290–291
data structures and description files, 293, 294f
debouncing measures, 217
decision tables
    Boolean instructions as, 241, 241f
    definition, 240
    optimizing, 242f
    rule R, 240
    sequential linking, 241, 242f
degraded operating mode, 109
description files, 173
    software components, 192, 192f
destructive instruction set architectures, 61
deterministic state machines, 245f
development, methods and tools for calibrating software functions, 204, 308–326
electronic control units and tools, offboard interface, 205–206
    logical system architecture and technical system architecture
    distributed and networked systems, 217–222
    open-loop and closed-loop control systems, 207–211
    real-time systems, 211–217
    reliable and safe systems, 222–230
development, methods and tools for
(Cont.)
offboard interfaces, 203
online calibration, 204
software functions. See software
functions
technical system architecture, 203
development ECUs, 173
diagnostic instrumentation, 329
diagnostic test equipment, 5
diagnostic trouble code (DTC), 117
digital data communications, 57
digital–analog (D/A) conversion, 47
DIN 19250 safety regulations, 23
direct-memory I/O (DMA), 63
discrete systems
ECU model, 45f
state machines, 48–50, 49f–50f
time- and value-discrete systems and
signals, 48, 48f
time-discrete systems and signals,
46–47
value-discrete systems and signals,
47–48
dispatcher, 70
cycle interval, 73
round, 73
table, 73, 74f
distributed and networked systems
analysis and specification of
actuators, 218
communications matrix, 222, 222f
influencing factors evaluation, 219
logical system architecture of, 218f
microcontrollers, 220f
onboard communications network,
217
sampling rate, 221
sensor signal preprocessing, 219
signal resolution, 221
software functions with sampling
rate and signals, 219f
benefits in automobile, 83–84
bus arbitration, strategies for, 94, 94f
centralized/decentralized
implementation, 95
controlled/random, 95–96
event-driven and time-controlled,
96, 96f
communication and network
management, organization of,
91–93, 91f–92f, 94f
definition, 82
ECU network, 83
electronic systems, 15–16, 16f, 81–82, 82f
implementation, 83f
logical and technical system
architecture, 84–85, 85f
logical communication links
client/server model, 85–86, 86f
sender/receiver model, 87, 87f
messages, 89, 89f
addressing, 89–90
communication matrix, 90, 90f
payload data, 89, 89f
multifunctional system, 82
networked ECUs, 82f
technical network topology, 88f
linear topology, 88–89
ring topology, 88
star topology, 88
truly concurrent tasks, 83
DMA. See direct-memory I/O (DMA)
documentation, software components,
192–193, 193f
drive program, 310
driver-vehicle-environment system
electronic systems, 5–6, 6f
functional features, 2
technical implementation, 2
vehicle electronic systems, design and
method of operation, 3–5
driveshaft, electronic system, 9
driving program, operating states, 175
dynamic RAM (DRAM), 54
electrical EPROM, 56
electrical erasable PROM (EEPROM), 56
electrics/electronics and software
architecture
ECU network, 119–121, 120f
gateway ECUs, 120
electrohydraulic braking system, 3–4
electromagnetic compatibility (EMC), 1,
37
Index

electronic control unit (ECU), 3–4
calibration, 173
discrete systems, 45f
distributed and networked systems, 82f, 83
electrics/electronics and software architecture, 119–121, 120f
eMBEDDED SYSTEMS, 51
fault memory, 117, 117f
function and networks of vehicle, 18, 18f
function networks and networked, 31
gasoline-engine, 10f
gateway, 7f, 8, 120
instrumentation, 303
model, discrete systems, 45f
onboard communications, 87, 87f
per vehicle, 17f
production, 172–173
schematic, 7f, 17
simulation and rapid prototyping, 254, 255f, 258, 259f
vehicle development processes, 20
vehicle electronic systems, 3–4, 6–8, 7f
in virtual environment, 300f
electronic throttle control (ETC) system, 106
embedded systems, 4
definition, 50
ECUs function development, 51
IDENTIFIERS, 50, 51f
intelligent actuators, 51
memory technologies
non-erasable read-only memory, 55
overview, 55f
read/write memory, 54
reprogrammable nonvolatile memory, 55–56
semiconductor memory, 53–54
semiconductor storage, 54
word length, 54
microcontroller
construction, 52–53, 53f
functional principles, 57, 58f
I/O module architecture, 61–63
microprocessors, 56, 58–61
operations, 58
program version and data version, 57
response characteristics, 51
user interface, 50
emergency stop, 109
empirical failure
function, 99
rate, 100
empirical reliability function, 99
end-of-line configuration, 292
engine management tasks, 64
environment models, microcontrollers, 44
erasable PROM, 55
error detection and correction, 108–109
event messages, 93
event-driven strategy, 73
evolutionary prototype, 257–258
execution rate, 66
time, 67
extended task state model, 69, 69f
fail-operational (FO) system, 109
fail-reduced (FR) system, 109
failure, 98
frequency, 99
probability, 99
failure-tolerant design, 84
fault, 97–98
diagnostics, 107–108
memory manager, 116–118, 117f
recognition, 106–108
fault memory, 56, 115, 229, 289, 314, 328
manager, 117–118, 117f
first-in, first-out (FIFO) principle, 70–71
fixed-point representation, 269
Flash EPROM, 56
flash memory, erasing and programming, 332
flash programming
integration, system components, 196
software functions, design and implementation, 291–292, 292f
software update through
availability requirements, 336–337, 337f
boot block shifting and flash programming, 337–338, 337f
Index

flash programming (Cont.)
cost-saving potential, 331
flash memory, erasing and programming, 332
offboard diagnostic interface, 332–333, 333f
security requirements, 334–335, 335f
Flash segments, 313
FlexRay, 120–121
floating-point representation, 269–270
fullpass system, 255, 255f
fully preemptive scheduling, 72
functional and nonfunctional system requirements, 161
functional software safety, 110–111
gasoline-engine ECU, 10f
gateway ECUs, 7f, 8, 120
graphical function model, 29
handshake/acknowledgment, 108
hard and soft real-time requirements, 67–68
hardware abstraction layer (HAL), 35
hardware interrupt system, 59
hardware-in-the-loop simulation, 300
hardware-oriented programming, 61
hazard, system safety, 104
hierarchy building, 126
high-level languages, behavioral model using, 247, 248f
holding elements, 48
horizontal and vertical prototypes, 252, 253f
hydraulic test bench, 305
IEC 61508 safety regulations, 23
ignition map, 44, 44f
input and output (I/O) modules, 52
installation space, 10, 12, 14
instruction set, 61
integrated quality management
quality assurance guidelines, 31
quality control, validation, and verification measures, 31–32, 32f
software quality characteristics, 31
integrated safety management, 7
integration
platform method, 294
system theory, 126
test, 150
“Integrator” class in C language, 247, 248f
interrupts, 59
interrupt service routine, 59
interrupt-driven I/O, 63
I/O module architecture
addressing, 62
operating mode, 62–63
schematic diagram, 61, 62f
ISO 26262 safety regulations, 23
isolated I/O, 62
JASPAR, 34
K-Line, 9
lab cars, 30
labeled memory, 238
laboratory vehicles, 30
liability and copyright protection, 35
limit risk, 104
limp-home operating mode, 109
linear topology, 88–89
linear-order layer model, 174
line-of-visibility (LOV) diagrams
artifact, 143
process description using, 143, 144f
symbols used in, 143, 143f
logical communication links, 84–85
client/server model, 85–86, 86f
sender/receiver model, 86–87, 87f
logical system architecture, 18, 81–82, 81f
abstract logical model, 160
constraints, 161
control and monitoring systems, 19, 19f
ECU schematic, 7f, 17
function and ECU networks of vehicle, 18, 18f
function network, 162
functional and nonfunctional system requirements, 161
instrument cluster, 162, 163f
nonstandard system functions, 161
standard system functions, 161
technical system architecture, 18
user requirements and, 160, 161f, 162, 162f
vehicle functions to vehicle subsystems, 18, 18f
low-fuel indicator lamp, 49
machine code, 190
machine instruction, 190
machine numbers, 270
malfunction, 98
display, 118
man/machine interface (MMI), 331
master/slave architecture, 95
Mealy state machines, 243
mean availability, 102–103
mean time to failure (MTTF), 99, 101–102
mean time to repair (MTTR), 102
memory segment, 291
memory technologies, embedded systems
non-erasable read-only memory, 55
overview, 55f
read/write memory, 54
reprogrammable nonvolatile memory, 55–56
semiconductor memory, 53–54
semiconductor storage, 54
word length, 54
memory-mapped I/O, 62
memory/memory architecture, 60
memory/register architecture, 60
messages, 89, 89f
addressing, 89–90
communication matrix, 90, 90f
payload data, 89, 89f
sequence chart, 76, 76f
microcontrollers, 220f
abstraction layer, 63
construction, 52–53, 53f
instrumentation signals, 313–314
monitoring functions, 112–113
programming
functional principles, 57, 58f
I/O module architecture, 61–63
microprocessor architecture and instruction set, 58–61, 59f
microprocessors, 56
principal microcontroller operations, 58
program version and data version, 57
microprocessor, 52
architecture and instruction set, 59f
accumulators, 59
control registers, 59
hardware interrupt system, 59
inbound data, 58
interrupt service routine, 59
operand addresses, 60–61
operand memory, 60
mnemonics, 191
model-based development, electronic systems
automated code generation methods, 30
development steps, 30f
engineering functions, 29
function networks and networked ECUs, 31
graphical function model, 29
laboratory vehicles, 30
software functions, calibration, 30
model-based fault recognition, 118–119, 119f
model-based software development, 157–158
model-in-the-loop simulation, 250
modularization, system theory, 126
module-based specification of interfaces to real-time OS, 235, 235f
monitoring, 107, 107f
communication links, 108
physical properties, 108
program execution
secure data processing indicator, 108
setpoint generators, sensors, actuators and control functions, 113, 114f
Moore state machines, 243
motor vehicle development, project planning, 136–139
multimaster architecture, 95
multimedia systems, vehicle electronic systems, 15
navigation feature, vehicle electronic systems, 5
network nodes, 84
network structures, 131, 131f
node addressing, 90
node monitoring as per AUTOSAR, 93–94, 94f
nondestructive instruction set architectures, 61
non-erasable read-only memory, 55
nonstandard system functions, 161
object-based software architecture modeling
antilock braking system (ABS), 233
attributes, 233
behavior, 233
class, 233
graphical visualization of, 235f
public methods, 233
unified modeling language (UML), 233
“Vehicle” class, class diagram, 234f
wheel revolutions per minute (rpm), 233–234
offboard communications, 5. See also onboard communications
offboard diagnostics. See also onboard diagnostics
communications, 118
functions, 114, 115f
interface, 332–333, 333f
offboard interfaces
for online calibration, 316–322, 317f
software architecture, 172–173, 173f
offline calibration of parameters, 314–315, 315f
execution, 6
and online calibration procedures, 310–311, 311f
onboard and offboard calculations, resource optimization, 262–263, 262f–263f
onboard communications, 5, 9. See also offboard communications
among ECUs, 87, 87f
network, 217
onboard diagnostics. See also offboard diagnostics
data, downloading and evaluating, 314
functions, 115–116, 115f–116f
onboard interfaces, software architecture, 172, 172f
online calibration of parameters, 315–316, 316f
online execution, 6
open systems interconnection model, 91
open-loop and closed-loop control systems
block diagram, 40–45, 41f
control function, 207
control units, 40
Controller 3, 208, 209f
input and output (I/O) parameters, 207
logical system architecture in, 207f–208f
modeling, 40
software function, external view, 210, 210f
technical system architecture, design, 210f
time constants, 210
open-loop control task, 41
operand addresses, 60–61
operand memory, 60
organizational clarity, 126, 126f
OSEK automotive standards, 20
OSI model, 91
overflow/underflow handling, 271
package requirements, 14
parameter and data version management, 325–326
parameter set, 43
partitioning, 126
passive safety group, 13
payload data, 89, 89f
personal profiles, 329
pins, I/O module architecture, 61
point in time, electronic systems, 6
polled I/O, 63
Post-Shutoff operating state, 176
powertrain of vehicle
driveshaft, 9
Index

installation space, 10
sensors and actuators, 9, 10f
software functions, 10
units and components, 9
user interfaces and setpoint generators, 9
variants and scalability, 10
process models and standards, 127–129, 128f
processor scheduling, strategies for, 69
combined sequential and priority strategy, 71–72, 71f
event-driven and time-controlled strategies, 73–74, 74f
nonpreemptive strategy, 72–73, 72f
preemptive strategy, 72
by priority, 70
in sequential order, 70
task management, state sets, 69–70, 70f
product and life cycle, configuration management, 129, 129f
production and service, methods and tools
electronic systems, startup and testing, 338
offboard diagnostic interface, 327–329, 328f
software functions, parameterization, 329–331, 330f
software update through flash programming
availability requirements, 336–337, 337f
boot block shifting and flash programming, 337–338, 337f
cost-saving potential, 331
flash memory, erasing and programming, 332
offboard diagnostic interface, 332–333, 333f
security requirements, 334–335, 335f
standardized communications protocols, 327
production ECU, 172–173
program and data memory, 52
program version and data version logistics, 335
programmable ROM (PROM), 55
programmed I/O, 63
project management, 133–134, 134f
project planning
cost planning, 136
development roles and responsibilities, 139–140, 140f
motor vehicle development, 136–1396
project phase, 135, 135f, 138f
project phases, 135, 135f
project scheduling, 136
quality planning, 136
project targets, 134, 134f
project tracking and risk management, 140–141
proportional-plusintegral (PI) controller, 42
prototype model, systems engineering and software engineering, 157
prototyping and simulation
bypass communications, 254
bypass hooks, 253
bypass interface, 253
bypass system, 254f
computer-based replication and execution, 249
definition, 249
ECU and experimental system, communications, 254, 255f
ECU verification, 258, 259f
environmental components, 250
experimental system, 249, 256f
experimentation tool, 249
fullpass system, 255, 255f
horizontal and vertical prototypes, 252, 253f
model-in-the-loop simulation, 250
organization of, 249, 250f
program and data versions, 249
sensors and actuators, 256
software functions in physical vehicle, 251, 252f
software prototype, 251
target system identical, 257
throw-away and evolutionary, 257–258
quality assurance guidelines, 150
integration and testing procedures, 150–151
quality testing procedures, 150
software quality assurance methods, 151–152, 152f
quality control assurance guidelines, 31
validation and verification measures, 31–32, 32f
quality planning, 136
quantization/resolution, 280
quantizing errors, 47, 187
quasi-concurrent tasks, 76
queued messages, 92
rapid prototyping, 159, 159f. See also prototyping and simulation tools, 190
read-only memory (ROM), 55
read/write memory, 54
real-time model
software components
order of execution in ASCET, 181, 181f
processes and tasks, instructions, 181, 181f
requirements, 181
state-dependent reactive execution model, 182, 182f
state-independent reactive execution model, 182–183, 183f
software functions, specification and validation of, 249
real-time requirements, 66, 66f
real-time systems
analysis and specification of deadline violations in OS, monitoring and handling, 217
definition, 212f
schedulability, 213–216, 215f, 217
specification of, 211
worst-case execution time (WCET), 213, 216
worst-case response time (WCRT), 212–213
AUTOSAR, 63
interaction among tasks, 75
communication, 79–81, 80f
cooporation, 77–79, 77f–79f
in logical system architecture, 81, 81f
synchronization, 76–77, 76f–77f
organization of, 74–75, 75f
processor scheduling, strategies for, 69
combined sequential and priority strategy, 71–72, 71f
event-driven and time-controlled strategies, 73–74, 74f
nonpreemptive strategy, 72–73, 72f
preemptive strategy, 72 by priority, 70
in sequential order, 70
task management, state sets, 69–70, 70f
requirements
hard and soft real-time requirements, 67–68
processes, 68
task activation and task deadline, 66–67, 66f
software, real-time behavior, 63
tasks, 64–65, 64f–65f
basic task state model, 68, 69f
extended task state model, 69, 69f
redundant value check, 108
reference value check, 107–108
reference version, 314
registers, 57
relative cumulative failure frequency, 99
relative deadline, 66
reliability, 97
complex electronic systems, 223
customer desires, 222
failure rate analysis
reliability block diagram, 224–226, 225f–226f
required function, 224
requirement profiles, 224
steps, 223
system boundaries, 224
function and failure rate, 99–101
function calculation, 226–227, 227f
Index

representative requirement profiles, 224
reprogrammable nonvolatile memory, 55–56
requested nonfunctional product properties
characteristic curves and maps, resource optimization, 264–268, 265f–268f
hardware resources limitation, 184–186, 185f
memory configuration of real-time OS, 260, 260f
onboard and offboard calculations, resource optimization, 262–263, 262f–263f
program version and data version, 184
software function distribution over several tasks, 261, 261f
unnecessary message copies, offline optimization, 262, 262f
requirements management. See also user(s), requirements configuration management, 144
tracking requirements, 149, 149f
user requirements, 144–148, 146f–148f
vehicle-related projects, 143
residual bus simulation, 304, 305f
reusable software components, class-based specification, 236, 236f
ring topology, 88
risk
determination, 105–106, 105f–106f
reduction
early validation of software functions, 33
function development, 32
reuse of software functions, 33–34
system safety, 104
ROM/PROM, 55
rule R, 240
runtime environment (RTE), 35
runtime overhead, 73
safety, 98, 104
logic, 109–110
relevance of electronic systems in vehicle, 97
and reliability analysis
electronic throttle control (ETC) system, 228
failure mode analysis, 227
functions types, 229, 230f
IEC 26262 standard, 228
microcontroller monitoring functions, 228
monitoring concept, 228, 229f
permissible limit risk, 227–228
protective measures, 227
quality assurance measures, 230
technology, 103–105
sampling
elements, 47
rate, 47, 221
scalable system architectures, 130, 130f
scheduler, 70
security requirements
production and service, methods and tools, 334–335, 335f
semiconductor
memory, 53–54
storage, 54
sender/receiver model, 86–87, 87f
sensors
and actuators, 9, 10f, 12, 12f, 14, 256
diagnostics, 329
signal preprocessing, 219
Sensotronic brake control (SBC), 3, 3f
setpoint generators and sensors, 116–117
shared system and component responsibilities, core process, 155, 155f
signal resolution, 221
signature verification, 334–335
simulation and rapid prototyping bypass
communications, 254
hooks, 253
interface, 253
system, 254f
computer-based replication and execution, 249
definition, 249
ECU
and experimental system, communications, 254, 255f
verification, 258, 259f
simulation and rapid prototyping (Cont.)
- environmental components, 250
- experimental system, 249, 256f
- experimentation tool, 249
- fullpass system, 255, 255f
- horizontal and vertical prototypes, 252, 253f
- model-in-the-loop simulation, 250
- organization of, 249, 250f
- program and data versions, 249
- sensors and actuators, 256
- software
  - functions in physical vehicle, 251, 252f
  - prototype, 251
  - target system identical, 257
  - throw-away and evolutionary, 257–258
- simultaneous engineering, project schedule, 136
- situation-oriented testing, 308
- software and hardware diversity, 108
  - integration, system components download, 195
  - flash programming, 196
- software architecture
  - base and application software, 286–287, 287f
  - base software components, standardization, 287–288
  - components and subsystems, 170
  - operating states, 174–176, 176f
  - software components and associated interfaces
    - CAN bus module, 171
    - data flow and control flow, 171
    - information fragments, 171
    - offboard interfaces, 172–173, 173f
    - onboard interfaces, 172, 172f
    - software layers, 174, 175f
  - software requirements and specification of, 170, 171f
  - standardized software components, configuration, 288–289, 289f–290f
- software architecture and components interfaces, 232
  - to real-time OS, module-based, 235, 235f
- object-based modeling, 233–235, 234f–235f
- reusable components, class-based, 236, 236f
- software components
  - behavioral model, 178–180, 180f–181f
  - data model, 177, 178f
  - design and implementation of, 183–188, 183f
  - behavioral model, 187, 188f
  - data model, 186, 187f
  - nonfunctional product properties, 184–186, 185f
  - real-time model, 188
  - integration, 189, 189f
    - description files generation, 192, 192f
    - documentation generation, 192–193, 193f
    - program version and data version generation, 190–191, 190f
    - testing, 193–194, 194f
  - real-time model, 180–183, 181f–183f
  - and software integration tests, integration, 25
  - specification, 24, 177f
  - testing, 189, 189f
- software development for electronic systems
  - automation, 36–37, 36f
  - component-related delays, 29
  - development risk reduction, 32–34
  - development steps, 37
  - integrated quality management, 31–32, 32f
  - model-based development, 29–31, 30f
  - standardization, 34–35, 34f
  - tool integration, 29
  - tool-assisted methods, 28
  - user requirements, 29
  - V-Model, 29
- software diversity, 108
- software function, 6–7, 10, 12, 14, 204
  - integration and testing
experiments, design and automation, 307–308
intermediate steps in software and systems, 298f
laboratory vehicles and test benches, 299–306
objectives, 297f
software-in-the-loop (SiL) simulation, 298–299, 299f
parameterization, 329–331, 330f
sampling rate and signals, 219f
software functions, design and implementation
behavioral model
additional programmer method, 294
C language implementation, 295, 296f
integration platform method, 294
“Integrator” class, 295, 295f
model-based specification, 294
software components plus program and data versions, 294, 295f
data model
configuration parameters, 292, 293f
data structures and description files, 293, 294f
flash programming, 291–292, 292f
memory segment, 291
vehicle variants, 290–291
fixed-point and floating-point arithmetic, 282–285
guidelines, modeling and implementation, 286
numbers in digital processors, 268–271
overflows and underflows handling, 274–275
physical interrelation and fixed-point arithmetic, 278–281, 279f
physical model level and implementation level, 281–282
rounding errors in integer division, 271–273
shift operations, 274
fixed-point and floating-point arithmetics
addition, subtraction and multiplication, 273
error propagation, 275–278
nonfunctional product properties, 259
requested nonfunctional product properties
characteristic curves and maps, 264–268, 265f–268f
memory configuration of real-time OS, 260, 260f
onboard and offboard calculations, 262–263, 262f–263f
software function distribution, 261, 261f
unnecessary message copies, 262, 262f
software architecture
base and application software, 286–287, 287f
base software components, 287–288
standardized software components, 288–289, 289f–290f
software functions, specification and validation
abstraction level of, 231
behavioral model using block diagrams, 237–240
decision tables, 240–242
high-level languages, 247, 248f
state machines, 242–247
data model, 237
formal and informal, 231
model-based, 231
real-time model, 249
simulation and rapid prototyping.
See simulation and rapid prototyping
software architecture and software components, 232–236
specification methods, 231f
Software Parameterization, 175–176
software prototype, 251
software quality characteristics, 31
software requirements, 24
software update through flash programming, 312–313, 312f
software-in-the-loop (SiL) simulation, 298–299, 299f
specification and integration, interfaces for, 142
spiral model, systems engineering and software engineering, 157
standard system functions, 161
standardization, software development, 34–35, 34f
star topology, 88
start state, 244, 244f
startup block, 332
state diagrams, state machines, 243
state machines
behavioral model specification
deterministic, 245f
differentiations, 244
equivalent actions in, 245f
hierarchy state machines, 246–247, 247f
low-fuel indicator lamp, 243
Mealy state machines, 243
Moore state machines, 243
nondeterministic situations, 244
start state, 244, 244f
state diagrams, 243
transition actions, 243
transitions with branching instructions, 246, 246f
“trigger()” method, 245f, 246
discrete systems, 48–50, 49f–50f
operating states, 175
state messages, 93
state-dependent reactive execution model, 182, 182f
state-independent reactive execution model, 182–183, 183f
static RAM (SRAM), 54
strict-order layer model, 174
subcontractor/supplier management
 cross-corporation development process, 142–143, 143f–144f
specification and integration, interfaces for, 142
system and component responsibilities, 141–142, 141f
support processes, 26
electronic systems and software development, 123, 124f
switching network, 240
synchronization, 76–77, 76f–77f
synchronized execution, 6
system and acceptance test, 201–202, 202f
system and component responsibilities, 141–142, 141f
system components integration, 194–195, 195f
 component, subsystem and system environments, 197f
component test and system test, interdependencies, 197f
prototype vehicles, 196
software and hardware integration, 195–196, 196f
synchronization point, 196
test routines, 196
system deviation, 42
system integration test, 198f
benefits, 199–200
consistent testing process, 200, 200f
physical components, 198
program version and data version, 200
requirements, 198
test object and testing environment, 198, 199f
virtual hardware, 200
virtual hardware platform, 200
virtual network environment, 199, 199f
system monitoring and diagnostics
error detection and correction, 108–109
and fault diagnostics, 107–108
fault recognition, 106–108
functional software safety, 110–111
monitoring, 107, 107f
safety logic, 109–110
system reliability and availability
diagnostic system organization for ECU
 actuator diagnostic functions, 117
fault memory manager, 117–118, 117f
model-based fault recognition, 118–119, 119f
offboard diagnostic communications, 118
offboard diagnostic functions, 114, 115f
onboard diagnostic functions, 115–116, 115f–116f
setpoint generators and sensors, 116–117
mean availability, 102–103
mean time to failure, 101–102
mean time to repair, 102
monitoring system organization for ECU, 111, 112f
microcontroller monitoring functions, 112–113
monitoring setpoint generators, sensors, actuators and control functions, 113, 114f
reliability function and failure rate, 99–101
system monitoring and diagnostics error detection and correction, 108–109
fault recognition, 106–108
functional software safety, 110–111
monitoring, 107, 107f
safety logic, 109–110
system safety risk determination, 105–106, 105f–106f
safety technology, terms in, 103–105
system test, 151
system theory, 124–127, 125f–126f
aggregation relations, 126
assumptions, 124
block diagram, 125f
definitions, 125–126
hierarchy building, 126
integration, 126
modularization, 126
organizational clarity, 126, 126f
partitioning, 126
target system-identical prototyping, 257
task assignment, 20
task states
  basic model, 68, 69f
  extended model, 69, 69f
technical communication links, 84
technical network topology, 88f
  linear topology, 88–89
  ring topology, 88
  star topology, 88
technical system architecture, 18, 24
  constraints, 164
designing, for instrument cluster, 166, 166f
distributed and networked systems, 169, 169f–170f
instrument cluster hardware, 166, 167f
logical system architecture and specification, 163, 164f
man/machine interface, 166
MOST ring topology, 166, 167f
objectives, 165–166
open-loop/closed-loop control systems, 167, 168f
real-time systems, 168
reliable and safe systems, 169, 170f
specification of, 165f
Terminal 15, operating states, 176
Terminal R, operating states, 176
throw-away prototype, 257–258
time- and value-discrete systems and signals, 48, 48f
time division multiple-access (TDMA) strategy, 95–96
time driven strategy, 73
time-discrete systems and signals, 46–47
token-controlled and time-controlled methods, 95
tool reference page, 314, 315f
traction control system (TCS), 4
transfer function of PI controller, 43
transport layer, 91
tree structures, 131, 131f
“trigger()” method, state machines, 245f, 246
truly concurrent tasks, 83
units and components, 9
unqueued messages, 92–93
user(s)
  interfaces, 5
  and setpoint generators, 9, 11, 13–14
requirements
  accepted, 147
category, 146
groups identification, 145–146, 146f
logical and technical system
architecture, 148, 148f
tracking, 149, 149f
types, 145
vehicle-specific, 147, 147f
validation, 32, 32f
value-discrete systems and signals, 47–48
variants and scalability, 10, 12, 14–15
configuration management, 130, 130f
vehicle components, 12
vehicle development processes
components, 19
core process, 24–26, 25f
electronic systems, 21f
age development methods, 20
automotive standards, 20
capability maturity model
integration, 20
cost, 22
ECUs, 20
long product life cycles, 22, 23f
safety requirements, 23
software process improvement and
capability determination, 20
trend from hardware to software,
21–22
V-Model, 20
electronic systems and software,
production and service, 28
responsibilities and accountabilities, 20
secondary subsystems, 19
support processes, electronic systems
and software development
customer/supplier relationships,
27, 27f
definition, 26
simultaneous engineering and
development environments,
27–28, 28f
task assignment, 20
vehicle variants, 20
vehicle electronic systems
body electronics, 13–15
chassis, 12–13
design and method of operation
actuators, 3
Bosch Sensotronic brake control
(SBC), 3, 3f
brake pedal, 3
control and monitoring systems,
block diagram, 4–5, 4f
controlled system, 3
ccontroller area network (CAN) bus,
4
diagnostic test equipment, 5
electrohydraulic braking system,
3–4
electronic control unit (ECU), 3–4,
6–8, 7f
interfaces, 5
navigation feature, 5
traction control system (TCS), 4
distributed and networked electronic
systems, 15–16, 16f
multimedia systems, 15
powertrain of vehicle, 9–10
vehicle functions to vehicle subsystems,
18, 18f
vehicle test record, 117
vehicle variants, 20, 290–291
vehicle-specific user requirements, 147,
147f
verification, 32, 32f
version-capable, 132
versions and configurations,
configuration management,
131–133, 131f–132f
V-Model, 24, 127–128, 156–157, 158f
electronic system development, 20
in quality control and testing, 31
quality inspection and test procedures,
24
software development for electronic
systems, 29
watchdog module, 52
Weibull distribution, 100
word length, 54
working page, 314, 315f
worst-case execution time (WCET), 74
worst-case response time (WCRT), 212
About the Authors

Dipl.-Ing. Jörg Schäuffele studied Mechanical Engineering at the University of Stuttgart with a focus on control engineering. He started his professional career in 1993 with the firm of ISG GmbH, a spinoff of the Institute of Control Engineering at the University of Stuttgart. In 1995, he joined ETAS GmbH and supported the introduction of ETAS tools in numerous engineering and consulting projects with various customers in the automotive industry worldwide. From 1999 to 2001, he worked in the Systems Engineering Methods, Software Process, Tools department of the Electronics Development section at BMW Headquarters in Munich. From 2001 to 2006 he coordinated the various product divisions at ETAS. Since 2007 he works as Product Manager at Vector Informatik GmbH.

Dr.-Ing. Thomas Zurawka studied electrical engineering at the University of Stuttgart and wrote a detailed doctorate thesis on digital signal processing. He started his professional career in 1992 with ZF Friedrichshafen AG before joining Advanced Development at Robert Bosch GmbH in 1993, where he worked on real-time operating systems. Having joined ETAS in its early days Dr. Zurawka initially developed code generators. He subsequently headed the development of the ASCET engineering tool for software development. Since 1997, he has been responsible for the overall development at ETAS and since 2001 CEO of ETAS. He worked as Executive Vice President for Robert Bosch GmbH. Since 2006 he is shareholder and managing director at Systecs Informationssysteme GmbH.