Index

a priori testing. *See* offline testing
`Abstract`, 256–257. *See also* Attributes, NModel
abstract state machine (ASM), 53, 150, 275
abstract value, 247, 256–257, 315. *See also* object ID
abstraction, 5, 62, 211, 315
  behavioral, 62–63, 317
  data, 62, 64, 74–75, 318
  environmental, 62, 70, 320
  level of, 5, 59, 62–63, 75, 323
abstraction function. *See* state property
accepting state, 6, 50, 58, 95, 108–110, 198, 228, 315
accepting state condition, 108–110, 289–290, 315
`AcceptingStateCondition`, 108, 289–290. *See also* Attributes, NModel
access modifier
  private, 67, 70, 99
  public, 26, 67, 70, 73, 99, 281, 283
action, 4, 58, 63, 73, 83, 141–142, 170, 216–217, 226–228, 232, 235, 252, 284–287, 316
  atomic, 142, 317
  cleanup, 217, 317
  controllable, 8, 33, 142, 216, 318
  finish, 141, 216, 320
  internal, 216, 322
  observable, 8, 33, 142, 216, 259–261, 268, 324
  shared, 123, 241, 327
  split, 142, 286, 328
  start, 141, 328
  tester. *See* controllable action
  timeout, 268–270, 330
  unshared, 123–124, 241
  wait, 268–270, 331
`Action`, 68, 284–287. *See also* Attributes, NModel
action method, 68, 316
action symbol, 63, 142, 316
action vocabulary, 63, 142, 170, 224, 252, 316
active state, 262, 316
adaptive strategy, 270–271, 316
Agedis, 276. *See also* testing tools
algorithms
  exploration, 103–106
  postman tour, 137, 148, 151, 325
  test execution, 214–217, 268—270
Alloy, 150, 219. *See also* modeling languages
alternating refinement, 277
analysis
  liveness, 6, 50–51, 94, 108–113, 323
  model-based, 5–7, 10, 47–52, 106–113, 131–136, 228–230, 232–240, 308–310, 324
  preliminary, 5, 61–64, 72–75, 82–85, 325
  safety, 6, 48–49, 94, 106–108, 326
  static, 3, 47, 328
animation. *See* simulation
API. *See* application programming interface

© Cambridge University Press www.cambridge.org

CAMBRIDGE

Cambridge University Press
978-0-521-88655-0 - Model-Based Software Testing and Analysis with C#
Jonathan Jacky, Margus Veanes, Colin Campbell and Wolfram Schulte
Index
More information

application, 15, 20–23, 316. *See also* assembly
application programming interface (API), 15, 316
approximation, 7, 316
arc. *See* link
arity, 124, 241, 316
ASM. *See* abstract state machine
AsmL, 150, 219, 276. *See also* modeling languages
AsmL-T, 150, 219, 276. *See also* testing tools
aspect-oriented programming, 275
assembly, 18, 23–25, 99–100, 121, 157, 197, 283, 316. *See also* application, `csc`, component, Global Assembly Cache (GAC), library, .NET framework, source program
assertion, 24–26, 316
assignment, 58, 68, 316. *See also* update
assurance method, 3, 317
asynchronous stepper, 264–265, 274, 317
atomic, 58, 317
atomic action, 142, 317
attribute, 24–26, 317
attributes, NModel
  `Abstract`, 256–257
  `AcceptingStateCondition`, 108, 289–290
  `Action`, 68, 284–287
  `Domain`, 75, 288–289
  `Feature`, 118, 282–283
  `Requirement`, 228, 291–292
  `StateFilter`, 187, 290–291
  `StateInvariant`, 106, 290
  `StatePropery`, 189, 291
attributes, NUnit
  `Category`, 25
  `Test`, 24
  `TestFixture`, 24

B, 150, 219. *See also* modeling languages
bag, 167–169, 302–304, 317. *See also* data types
  for coverage points, 204–208
bag (case study), 194–217, 262–263
behavior, 58, 317
behavioral abstraction, 62–63, 317
behavioral coverage, 97, 204, 317
binary relation, 169, 317
Bogor, 276. *See also* model checking
bug, 29, 317

C# 4, 8, 17, 23, 34, 54. *See also* `csc`, .NET framework
  delegate, 204, 214, 264
  enumerator, 156–157
  event, 34, 39, 264
  fully qualified name, 20, 321
  indexer, 145, 199, 322
  namespace, 20, 67, 247, 324
  overloading, 69
  partial class, 171
  simple name, 20, 327
  static class, 75, 328
  string, 156–159
C# compiler. *See* `csc`
callback, 264
`Category`, 25. *See also* attributes, NUnit
case studies
  bag, 194–217, 262–263
  client/server (temperature monitor), 13–31, 33, 72–82, 115–121, 127–131, 134–136, 138–147, 192–193
  payroll, 247–257
  reactive system (process controller), 32–52, 82–92, 106–113, 134–136
  revision control system, 169–182
  sample protocol (client/server), 224–244, 260–261, 264–268
  user interface (news reader), 59–72, 95–103, 132–133
Chinese postman tour. *See* postman tour
circular reference, 248. *See also* object graph
cleanup action, 217, 317
CLI. *See* Common Language Infrastructure
client/server (temperature monitor case study), 13–31, 33, 72–82, 115–121, 127–131, 134–136, 138–147, 192–193. *See also* sample protocol
closed system, 8, 33, 137, 317

© Cambridge University Press www.cambridge.org

CAMBRIDGE

Cambridge University Press
978-0-521-88655-0 - Model-Based Software Testing and Analysis with C#
Jonathan Jacky, Margus Veanes, Colin Campbell and Wolfram Schulte
Index
More information

Common Language Infrastructure (CLI), 54. *See also* .NET framework
compiler. *See* `csc`
complete, 47
component, 4, 18, 317. *See also* assembly
component testing, 9–10
composition, 4, 103, 115, 121–129, 131–136, 188, 223–246, 253–254, 275–276, 317
compound value, 158–159, 172, 292–293, 318. *See also* data types
collection types, 157–169, 292–305. *See also* data types
concurrent system, 4, 272, 318
conform, 260, 318
conformance relation, 260–261, 318
Conformance Tester (tool). *See* `ct`
conjunction, 198, 228, 318
contract model program, 58, 117, 184, 318
control state, 64, 74, 80–81, 84–85, 318
control structure, 23, 47, 62, 67, 69, 75, 83, 318
controllable action, 8, 33, 142, 216, 318
cover, 318
coverage, 193, 203–209, 318
coverage point, 191, 204–209, 318
CppUnit, 23. *See also* testing tools
crosscutting, 244. *See also* design
`csc` (C# compiler), 17–18, 23. *See also* assembly, C#, .NET framework
`ct` (Conformance Tester Tool), 7–8, 137, 146–148, 192–193, 201, 208–210, 212–217, 267–270, 312–314
current state, 95, 318
cycle, 96, 318
cyclic, 182

data abstraction, 62, 64, 74–75, 318
data state, 64, 74, 85, 92, 318
data types, 157–169, 248–257, 292–307
  bag, 167–169, 302–304, 317
  compound value, 158–159, 172, 292–293, 318
  enumerator, 156–157
  labeled instance, 248–253, 257, 305–306
  map, 162–164, 172–173, 247–248, 296–298, 323
  pair, 169, 304–305, 325
  sequence, 164–166, 172, 298–301, 327
  set, 159–162, 173, 293–296, 327
  string, 156–159
  triple, 169, 305, 330
  value array, 166, 301–302, 330
dead state, 6, 50, 94, 108–111, 138, 318
deadlock, 6, 50, 94, 112–113, 318
decrementing action weight, 217–218, 319
De Morgan's Law 107, 319
defect, 29–31, 46–47, 319
delegate, 204, 214, 264. *See also* C#
dependent feature, 230, 319
derived state, 173, 319
design, 3, 4, 8–10, 46–47, 319. *See also* crosscutting
deterministic, 47, 271, 319
difference, 162, 319
directed graph, 96, 319
disabled, 97, 173, 319
disjunction, 319
dispatcher, 34
distributed system, 4, 58, 72, 272, 277, 319
domain, 75, 81–82, 106, 120–121, 183–186, 230, 252, 288–289, 319
`Domain`, 75, 288–289. *See also* Attributes, NModel
dynamic function, 162, 319

edge. *See* link
efficiency, 5
embedded computer, 14, 32, 277
enabled, 40, 46–48, 63–64, 68–69, 81, 83–85, 96–97, 101–104, 107–113, 118–120, 123–126, 192, 199, 201–202, 207, 216–218, 228, 261–262, 268–271, 319
enabling condition, 62–63, 68–70, 187–188, 229, 287–288, 320. *See also* strengthen
end state. *See* next state
engineering, 9–10
enumerator, 156–157. *See also* C#, data types
environmental abstraction, 62, 70, 320

© Cambridge University Press www.cambridge.org

CAMBRIDGE

Cambridge University Press
978-0-521-88655-0 - Model-Based Software Testing and Analysis with C#
Jonathan Jacky, Margus Veanes, Colin Campbell and Wolfram Schulte
Index
More information

event, 8, 34–43, 46–48, 83–85, 259, 264, 272–274
  C# 34, 39, 264
event-driven system. *See* reactive system
exhaustive, 52, 57, 100, 320
explorable, 121, 183–186, 230, 267, 320
exploration, 6, 94–114, 150, 183–190, 228–230, 232–234, 235, 238, 245, 320
  algorithm, 103–106
  exhaustive, 52, 100
  interactive, 101–102, 105, 107, 111, 132, 263
  partial, 263

factory method, 99–100, 103, 118, 127–128, 142, 185, 210, 212, 249, 267, 281–282, 320
fail, 24–25, 320
failure, 28–29, 44–46, 320
fault. *See* defect
feature, 4, 61–62, 73, 83, 115, 117–121, 196, 223, 244, 282–283, 320
`Feature`, 118, 282–283. *See also* Attributes, NModel
feature-oriented modeling, 223–246, 320
field map, 247–249, 320
finalization, 245
finish action, 141, 216, 320
finite automata, 123, 150, 276
finite state machine (FSM), 6, 47–52, 94–114, 127–135, 150, 156–157, 320
finitely branching, 156, 320
finitize, 7, 57, 321
formal, 6, 321
forms, 34. *See also* .NET framework
frontier, 104–105, 111, 321
Foundations of Software Engineering group (FSE), 53
functional. *See* state-independent
FSE. *See* Foundations of Software Engineering group
FSM. *See* finite state machine
FSM model program, 128, 321
FSM text file, 128, 321
fully qualified name, 20, 321. *See also* C#, namespace

GAC. *See* global assembly cache
game (for testing), 277
generic failure, 44, 321
Global Assembly Cache (GAC), 23, 321. *See also* assembly, .NET framework
graph, 96, 321
group label. *See* state property
grouping. *See* state grouping
guard, 40, 93, 321. *See also* enabling condition
guarded update rule, 93, 321
guarded update program, 93, 321

handler, 34
helper method, 68, 321
hidden state, 148, 191, 271, 321

immutable, 158–160, 162, 164, 167, 169, 248, 321
implementation, 4, 57, 137, 142, 321
implementation under test (IUT), 7–8, 23–26, 191, 198, 211, 214–219, 259–262, 264, 271–273, 277, 322
indexer, 145, 199, 322. *See also* C#
"infinite," 7, 57, 64, 74, 99, 155–157, 322
infinitely branching, 156, 322
informal, 3, 322
initial state, 58, 95, 322
inspection, 3, 47, 322
interactive exploration, 101–102, 105, 107, 111, 132, 263
interface automata, 277
interesting (state), 218, 322
interleave, 116, 123, 129
internal action, 216, 322
intersection, 123, 150, 242, 322
  language, 53, 123, 150
  runs, traces, 123, 238, 242, 322
  sets, bags, 159, 162, 167, 322
invariant, 106–107, 228, 290, 322. *See also* safety requirement
invocation, 95, 96, 103, 199, 322

© Cambridge University Press www.cambridge.org

CAMBRIDGE

Cambridge University Press
978-0-521-88655-0 - Model-Based Software Testing and Analysis with C#
Jonathan Jacky, Margus Veanes, Colin Campbell and Wolfram Schulte
Index
More information

IOCO theory, 277
isomorphic states, 256–257, 276–277, 322
IUT. *See* implementation under test

Java, 276
JUnit, 23. *See also* testing tools

Korat, 276. *See also* testing tools

labeled instance, 248–253, 257, 305–306. *See also* data types, object
labeled transition system (LTS), 275, 277
lazy, 105, 267, 323
learn, 271, 323
level of abstraction, 5, 59, 62–63, 75, 323
level of integration, 8, 323
library. *See* assembly
  .NET 15–20, 23–25, 323
  NModel, 4, 57, 67, 103–104, 157–169, 281–307
library model program, 99, 118, 127–128, 157, 323
link, 96, 323
livelock, 7, 50, 94, 111–112, 323
liveness, 108, 323. *See also* safety
liveness analysis, 6, 50–51, 94, 108–113, 323
liveness requirement, 34, 108
lockstep execution, 7, 323
loop extension, 124–125, 241, 323
LTS. *See* labeled transition system

map, 162–164, 172–173, 247–248, 296–298, 323. *See also* data types
match
  actions, 125–126, 284–287, 323
  traces, 242
memoryless strategy, 202, 323
Microsoft Research, 53
model (engineering), 9–10
model checking, 6, 150, 219, 276–277, 324
  Bogor, 276
  Uppaal-Tron, 277
model program, 4–5, 8–11, 57–93, 281–282, 324
Model Program Viewer (tool). *See* `mpv`
model-based analysis, 5–7, 10, 47–52, 106–113, 131–136, 228–230, 232–240, 308–310, 324
model-based testing, 7–8, 10, 137–149, 191–218, 259–274, 311–314, 324
modeling languages, 150, 219, 276. *See also* NModel
  Alloy, 150, 219
  AsmL, 150, 219, 276
  B, 150, 219
  Promela, 150, 219
  Spec#, 150, 219, 276
  TLA, 150, 219
  Unity, 159, 219
  VDM, 150, 219
  Z, 150, 219
`mpv` (Model Program Viewer Tool), 6, 47–52, 94, 99–114, 127–129, 308–310
multiplexing, 272–274, 277, 324
multiplicity, 167, 324
multiset. *See* bag

namespace, 20, 67, 247, 324. *See also* C#, fully qualified name, simple name
.NET framework, 4, 15–23, 34–40, 54. *See also* assembly, C#, CLI, `csc`
  data types, 157–166
  Forms, 34
  Global Assembly Cache (GAC), 23, 321
  timer, 34, 39, 83
news reader (case study). *See* user interface
next state, 95, 324
NModel framework, 4, 67, 281–314. *See also* `ct`, `mpv`, `otg`
  attributes, 282–292
  commands, 308–314
  data types, 157–169, 248–257, 292–307
  library, 4, 281–307
  program structure, 67, 281–282
node, 95, 324
nondeterminism, 8, 28, 33, 62, 69–70, 261–263, 324
NUnit, 23–25. *See also* testing tools, unit testing

© Cambridge University Press www.cambridge.org

CAMBRIDGE

Cambridge University Press
978-0-521-88655-0 - Model-Based Software Testing and Analysis with C#
Jonathan Jacky, Margus Veanes, Colin Campbell and Wolfram Schulte
Index
More information

object, 99, 142, 157, 247. *See also* labeled instance
object graph, 182, 248. *See also* reachable
object ID, 247, 253. *See also* abstract value
object-oriented modeling, 247–258, 276–277, 324
observer
  model program, 211, 324. *See also* probe
  asynchronous stepper, 264, 324
observable action, 8, 33, 142, 216, 259–261, 268, 324
observation queue, 268–270, 324
offline testing, 7, 137–149, 191, 324
Offline Test Generator (tool). *See* `otg`
on-the-fly testing 7, 137, 191–218, 259–274, 277, 324
online testing. *See* on-the-fly testing
operational profile, 218
oracle, 4, 7, 26, 54, 145, 238, 325
`otg` (Offline Test Generator Tool), 7, 137–141, 148, 311
overloading, 69. *See also* C#

pair, 169, 304–305, 325. *See also* data types
pair state, 125–126
parameter generation, 105–106, 120–121, 183–186, 229, 245–246, 325
partial class, 171. *See also* C#
partial exploration, 263. *See also* interactive exploration
partial order, 272, 325
partially explored state, 271, 325
pass, 4, 7, 24–26, 325
passive state, 262, 325
passive testing, 272, 325
path coverage, 148, 325
payroll (case study), 247–257
placeholder, 124, 241, 286, 325
postman tour, 137, 148, 151, 325. *See also* algorithms, traversal
precondition. *See* enabling condition
predicate, 228, 325
preliminary analysis, 5, 61–64, 72–75, 82–85, 325
private access, 67, 70, 99
public access, 26, 67, 70, 73, 99, 281, 283
Promela, 150, 219. *See also* modeling languages
probe, 211–214, 325
process. *See* software process
process controller (case study). *See* reactive system
project. *See* software process
projection, 126–127, 132–136
product, 121, 325
program structure, 67, 281–282
progress, 7, 32, 47, 325
protocol, 14–16, 223–226, 326
property checking, 246. *See also* temporal property
pruning, 7, 186–190, 218, 326
pure, 69, 326
purpose, 61, 72–73, 82

random strategy, 191–193, 202–203, 326
reachable
  object, 249, 252, 290, 326. *See also* object graph
  state, 136, 228, 249, 326
reactive system, 4, 8, 32, 137, 259–274, 277, 326
reactive system (process controller case study), 32–52, 82–92, 106–113, 134–136, 326
reference (assembly), 23
reference equality, 158, 326
refinement, 246, 277
regular languages, 123, 150
requirement, 61, 228–229, 267, 291–292
`Requirement`, 228, 291–292. *See also* Attributes, NModel
restricted model, 197, 326
review. *See* inspection
revision control system (case study), 169–182
reward, 206–209, 217, 218, 326
run, 7, 58, 96–97, 326. *See also* trace
run-time check 70, 326

safety, 106, 326. *See also* liveness
safety analysis, 6, 48–49, 94, 106–108, 326
safety condition. *See* safety requirement

© Cambridge University Press www.cambridge.org

CAMBRIDGE

Cambridge University Press
978-0-521-88655-0 - Model-Based Software Testing and Analysis with C#
Jonathan Jacky, Margus Veanes, Colin Campbell and Wolfram Schulte
Index
More information

safety requirement, 34, 106–107, 246, 326. *See also* invariant
sample. *See* case study
sample protocol (client/server case study), 224–244, 260–261, 264–268
sampling, 7, 190, 217, 326
sandbox, 8, 28, 41, 327
scenario, 6–9, 25–27, 59, 96–97, 115, 212, 327
scenario control, 115–123, 127–132, 138, 188, 244–245, 271, 276, 282, 327
scenario FSM, 97, 133–134, 253, 254, 257, 327
scenario model program, 59, 127–129, 132–134, 238, 327
self-loop, 124, 241, 327
sequence, 164–166, 172, 298–301, 327. *See also* data types
set, 159–162, 173, 293–296, 327. *See also* data types
shared action, 123, 241, 327
side effect, 69, 327
signature, 286, 287, 327
simulation, 6, 44–46, 70–72, 327
simulator, 34, 38–41
simple name, 20, 327. *See also* C#, namespace
slice, 59, 62, 327
sliding window, 224–226, 327. *See also* protocol
socket, 15–16, 54, 75
software process, 8–11
source program, 18, 327. *See also* assembly, `csc`
Spec# 150, 219, 276. *See also* modeling languages
Spec Explorer, 150, 219–220, 276. *See also* testing tools
specification, 3, 4, 8–10, 58, 327
split action, 142, 286, 328
start action, 141, 328
startup, 245
state, 4, 26, 58, 171, 328. *See also* state variable
  accepting, 6, 50, 58, 95, 108–110, 198, 228, 315
  active, 262, 316
  control, 64, 74, 80–81, 84–85, 318
  current, 95, 318
  data, 64, 74, 85, 92, 318
  dead, 6, 50, 94, 108–111, 138, 318
  derived, 173, 319
  end. *See* next state
  hidden, 148, 191, 271, 321
  initial, 58, 95, 322
  isomorphic, 256–257, 276–277, 322
  next, 95, 324
  pair, 125–126
  partially explored, 271, 325
  passive, 262, 325
  reachable, 136, 228, 249, 326
  target. *See* next state
  unsafe, 6, 48–49, 94, 106–108, 228–229, 330
state filter, 186–187, 290–291, 328
state grouping, 189–190, 328
state-independent, 26, 328
state property, 189–190, 291, 328
state space, 64, 191
state transition, 95, 328
state transition diagram, 6, 95, 328
state transition table, 95, 328
state variable, 58, 64, 67–68, 74–75, 84–85, 171–173, 226, 232, 235, 244, 283–284, 328. *See also* state
`StateFilter`, 187, 290–291. *See also* Attributes, NModel
`StateInvariant`, 106, 290. *See also* Attributes, NModel
`StatePropery`, 189, 291. *See also* Attributes, NModel
static analysis, 3, 47, 328
static class, 75, 328. *See also* C#
step, 53, 58, 328. *See also* action
stepper, 142–146, 192, 198–200, 211–212, 328. *See also* test harness
stochastic, 190, 328
stopping rule, 105, 329

© Cambridge University Press www.cambridge.org

CAMBRIDGE

Cambridge University Press
978-0-521-88655-0 - Model-Based Software Testing and Analysis with C#
Jonathan Jacky, Margus Veanes, Colin Campbell and Wolfram Schulte
Index
More information

strategy, 8, 191, 199–209, 217–218, 329
  adaptive, 270–271, 316
  memoryless, 202, 323
  random, 191–193, 202–203, 326
strengthen, 187–188, 229, 329. *See also* enabling condition
string, 156–159. *See also* C#, data types
structural coverage, 204, 329
structural equality, 157, 248, 329
Symstra, 276. *See also* testing tools
system testing, 9–10, 23, 329

TCP/IP (Internet protocol), 13, 15
target state. *See* next state
temperature calibration (case study). *See* reactive system
temperature monitor (case study). *See* client/server
temporal logic, 150, 277
temporal property, 4, 6, 113, 115, 132, 150, 246, 329
term, 141, 144–145, 211–212, 306–307, 329
term label, 249, 253–254, 329
test (NUnit), 23–25, 329
`Test`, 24. *See also* attributes, NUnit
test case, 137, 329
test-driven development, 9, 329
test execution, 214–217, 268–270. *See also* algorithms
test fixture, 24, 329
test harness, 8, 142, 211, 254–256, 329. *See also* stepper
test method, 24, 329
test run, 217, 329
test runner, 23, 329
test suite, 137, 329
`TestFixture`, 24. *See also* attributes, NUnit
tester action. *See* controllable action
testing, 3, 23, 329
  a priori. *See* offline testing
  component, 9–10
  model-based, 7–8, 10, 137–149, 191–218, 259–274, 311–314, 324
  offline, 7, 137–149, 191, 324
  on-the-fly, 7, 137, 191–218, 259–274, 277, 324
  online. *See* on-the-fly testing
  passive, 272, 325
  system, 9–10, 23, 329
  unit, 9–10, 23–30, 40–44, 70–72, 330
testing tools. *See* `ct`, `otg`
  Agedis, 276
  AsmL-T, 150, 219, 276
  CppUnit, 23
  JUnit, 23
  Korat, 276
  NUnit, 23–25
  Spec Explorer, 150, 219–220, 276
  TorX, 277
  TGV, 277
  Uppaal-Tron, 277
  xUnit, 23, 54
TGV, 277. *See also* testing tools
timeout action, 268–270, 330
timer, 33–41, 83–85, 92. *See also* .NET framework
TLA, 150, 219. *See also* modeling languages
top level, 68
TorX, 277. *See also* testing tools
trace, 58, 63, 73–74, 83–84, 141, 147, 170–171, 238–244, 261, 330. *See also* run
transition, 6, 330
transition coverage, 148, 330
traversal, 7, 96–97, 330. *See also* postman tour
triple, 169, 305, 330. *See also* data types
true FSM, 7, 97, 115–116, 330
type. *See* data types

union, 159, 162, 330
unit, 4, 8, 23, 330
unit testing, 9–10, 23–30, 40–44, 70–72, 330
unity, 150, 219. *See also* modeling languages
update, 68, 330. *See also* guarded update
update rule, 68, 93. *See also* action method, guarded update rule
unwinding, 330. *See also* traversal
unsafe state, 6, 48–49, 94, 106–108, 228–229, 330

© Cambridge University Press www.cambridge.org

CAMBRIDGE

Cambridge University Press
978-0-521-88655-0 - Model-Based Software Testing and Analysis with C#
Jonathan Jacky, Margus Veanes, Colin Campbell and Wolfram Schulte
Index
More information

unshared action, 123–124, 241
Uppaal-Tron, 277. *See also* model checking, testing tools
user interface (new reader case study), 59–72, 95–103, 132–133

V-diagram, 8–10
validate, 5, 11, 82, 224, 330
value array, 166, 301–302, 330. *See also* data types
value type, 158–159, 172, 194, 248, 253, 330
VDM, 150, 219. *See also* modeling languages
vertex. *See* node
visualization, 94, 100, 308
vocabulary. *See* action vocabulary

wait action, 268–270, 331
weaken, 331
wrapper, 15, 75, 331

xUnit, 23, 54. *See also* testing tools

Z, 150, 219. *See also* modeling languages

© Cambridge University Press www.cambridge.org