Abstract data type(s), 4, 72, 86, 223, 244
abstract operations, 4, 7, 244
encapsulation of concrete representation, 4
Abstraction layer, 51
Abstract Windowing Toolkit (AWT), 255
Adaptive programming, 71. See also DemeterJ
ADJ group, 211, 224
ADO.NET, 33, 242, 251
Advice, 56–58
custom attributes (in LOOM.NET), 81
Agile programming, 202–203, 205–206
eXtreme Programming (XP), 202, 205, 228, 236
relation to AOP, 205–206, 208
Scrum, 202–203
AJAX, 250, 251, 258, 259
Algebraic specification methods, 226
ALGOL 5, 246, 248
ALPHARD, 224, 244
Ancestor class, 245. See also Object-oriented programming
ANTLR, 215, 265, 267, 268
Applet, 44, 255
Argument consistency checking, 153–154
ArgumentException, 137–139
Aspect(s), 2, 55–56
classes (in LOOM.NET), 81
cross-language interoperability, 84
generic, 90
implementation language, 60
in Aspect.NET, 88–90
action(s), 89
condition(s), 89
after, 95, 100, 116
before, 91, 110, 116
instead, 100, 116
data, 89, 100, 105
header, 91
module(s), 89, 92, 93, 98, 105, 109, 114
name, 88, 111, 115
representing by custom attributes, 92–94
(weaving) rule(s), 89, 93–94, 109, 117
issues of introducing specification
features into implementation
language
conceptual confusion, 60, 68, 75
dependence on one’s own specific
tools for extended language
support, 61
difficult integration with common
use tools and IDE, 61
locating, 57
mining, 7, 29–30, 61–62, 65, 89–90,
165. See also Aspectizing
using lexical structure of the source
code
using types defined in the
application, 62
reusability, 54
specification language, 60–61
Synchronization, 90–92
updating, 20
AspectDNG, 80–81
Aspectizing, 89–90, 94, 98, 165
AspectJ, x, 66–71
advice declarations, 67
advices, kinds of
after, 68–69
around, 68–69
before, 68–69
ant tasks support, 67
aspect browser (ajbrowser), 67
aspect definitions, 67
alternative form using annotations,
69
compiler (ajc), 66
development tools (AJDT), 66
documentation utility (ajdoc), 66
integration with AspectWerkz, 69
intertype declarations, 67
language, 66–67, 69
load-time weaver, 67
named pointcut definitions, 67
pointcuts, types of
call, 69
execution, 69
handler, 69
initialization, 69
set, 69
privileged aspects, 68
thisJoinPoint, 67, 68
Aspect-oriented modeling, 275
Aspect-oriented programming (AOP),
2, 270, 275
adequacy for trustworthy computing,
2
annotations,
language neutral, 95, 96
applying for
argument consistency checking, 153,
156
business integrity, 29–30
design by contract, 129, 168–183
error handling, 129, 136–142
formal specification and verification,
164–165
logging, 157–161
multithreading and multicore-based
applications, 145–149
privacy, 25, 149–153
profiling, 161–164
reliability, 27, 153–157
security, 22, 132–136
synchronization, 142–145
framework, requirements to, 90
generic nature, 85
issues
“blind” weaving, 98
conceptual confusion, 75
efficiency, 63
losing aspect configuration
information, 75–76
recipes of growing AOP popularity,
63
security, 63
semantic gap between the sources
and the application code after
weaving, 75
metalanguage, 88–90, 92, 93, 95, 96,
98, 99, 101, 102, 104, 106, 107,
108, 114. See also AOP
specification language
perspectives, 123
increasing application in the
industry, 123
integration to knowledge
engineering, 124
ubiquitous use, 124
“pitfalls,” 8, 73, 75–76
recipes of trustworthy applying for trustworthy computing, 130–132
avoid weaving conditions with intersection of scopes, 131
browsing and deselecting join points if appropriate before weaving, 130
capturing join point context, 131
making the weaving conditions as narrow as possible, 130
relation to knowledge management, 9, 95, 97
scope of applying, 62
secure tools of, 63
specification language, complicated, as an AOP issue, 87
language-agnostic, 87
tools
requirements summary, 90
for .Java, summary, 78
for .NET, summary, 84
ubiquity, 85
Aspect#, 80–81
aspect(s)
self-documenting, 120. See also AspectDescription
Aspect class, when using customs attribute directly, 120–121
AspectDotNet namespace, 94, 110, 158, 171, 174
attributes for representing aspects
AspectAction (version 2.1), 120–121, 158
AspectDef (version 1.0), 114
AspectDescription (version 2.1), 110, 114, 120–122
convenience features, summary, 206
converter (preprocessor), 120
efficiency assessment, 193–202. See also performance assessment
framework, 61, 85, 92, 97, 99, 103, 104, 106, 107, 122, 125, 127, 128, 130, 135, 139, 167, 186, 187, 189, 204
Aspects tab, 111, 113, 135, 188
Find joinpoints button, 111, 135
options, 113–114
selection and deselecting of join points before weaving, 130, 167, 186, 188, 204, 207
Weave aspects button, 135, 158, 188
join point(s), 140, 167
call, 91, 100, 102, 109, 110, 116, 117, 120, 121, 125, 126
context capturing, 114, 118–120
performance assessment, 184–187, 193–202. See also efficiency assessment
perspectives, 125–127
aspect debugger, 126
aspectizer, 127
aspect repository, 127
aspect script actions, 126
%assign and %use join points, 126
displaying changes in the source code, 126
generic aspects, 90–92, 126
join point filters using attributes to mark entities, 125–126
productization, 125
rule sets, 126
TargetMemberInfo feature, 104, 109, 110, 116, 119, 122, 158
version 1.0, 103, 114
version 1.1, 103, 114
version 2.0, 104
version 2.1, 91, 102–105, 114, 117, 118, 119, 122, 136, 164, 190
porting from Aspect.NET 1.0 or 1.1, 104
scanning phase, 105
weaving phase, 105
keyword(s)
args, 116
aspect, 99, 109, 114–115
modules, 115
INDEX

RetValue, 119
rules, 115
SourceFileLine, 119, 120
SourceFilePath, 119
TargetMemberInfo, 119, 122
TargetObject, 119
This, 119
withincode, 118
WithinMethod, 119
WithinType, 119
rule sets, 126. See also Aspect.NET perspectives

ASP.NET, 250
Assembly, 21, 32–40. See also .NET
Assertion(s), 168, 169, 217, 246, 254
Atomic synchronization operations, 146
Attack
distributed denial of service (DDoS), 12, 220
Attributes, 33, 35–38, 39, 41. See also .NET
custom, 42
Authentication, 34, 37, 45, 46
Kerberos, 45
passport, 37
Windows, 37
AuthentiCode, 24

BESM-6, Russian mainframe, 214, 260
BoogiePL specification language, 41. See also Spec#
Bowl-of-spaghetti software, 3, 221
Boxing, 13, 44, 211, 253, 254
bug
closing, 29
duplicates of, 28
fixing, 29
Burroughs 5000/5500/6700/7700 computers, 5, 212, 216, 238
Business integrity, 10, 12, 13, 19, 27–30

C, 247
CaesarJ, 76, 77
CafeOBJ, 211
Casting, 247
CDL/2, 51
Chief programmer’s team, 214, 235
CLU, 5, 90, 137, 211, 214, 223–224, 244, 248, 254, 257
Cluster (in CLU), 224
CMM, 235, 236
COCOMO model, 184–186
Coercion, 246–247
COM, 241
Communications, software process principle, 17–18
Compiler
ahead-of-time (AOT), 214, 267
development, 213–215, 218, 263–268
error diagnostics, 265
error recovery, 265
just-in-time (JIT), 32, 242, 250, 264, 267
trustworthy, 219, 264
verifying, 40, 219
Component-oriented software, 241
Compose*, 80
Composition filters, 72
implementation part, 72
interface part, 72
input and output filter sets, 72
superimposition clause, 72
Conceptual explosion, 5, 75, 218, 225
Concern, 52
core, 55
cross-cutting, 2, 55, 226
graph, 72. See also FEAT
modular, 55
separation of, 71
multidimensional, 71
Concurrency, 211
Contract
in .NET Common Type System, 246. See also Design by contract
CORBA, 241
cpp preprocessor, use with Java, 48
Critical section(s), 47, 143
C++, 21, 257, 268

Deadlock, 47, 49, 143, 146, 148, 149
Definition
current effective, for an identifier, 213
Demand, imperative security method, 36
DemeterJ, 71
Denotational semantics, 26, 219, 226
Deprecated method, 46–47
<table>
<thead>
<tr>
<th>Term</th>
<th>Page(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Descendant class</td>
<td>245</td>
</tr>
<tr>
<td>Descriptor</td>
<td>6</td>
</tr>
<tr>
<td>Design by contract</td>
<td>40, 168–183, 219, 222, 284, 295</td>
</tr>
<tr>
<td>contract notion</td>
<td>168</td>
</tr>
<tr>
<td>Eiffel</td>
<td>168</td>
</tr>
<tr>
<td>postcondition (ensure)</td>
<td>168</td>
</tr>
<tr>
<td>precondition (require)</td>
<td>168</td>
</tr>
<tr>
<td>DISPAK</td>
<td>260</td>
</tr>
<tr>
<td>DLL hell</td>
<td>33</td>
</tr>
<tr>
<td>E-mail(s)</td>
<td>12–13, 23–24, 28, 216, 235–238</td>
</tr>
<tr>
<td>as a tool for software process</td>
<td>235–236</td>
</tr>
<tr>
<td>Embedded systems</td>
<td>240</td>
</tr>
<tr>
<td>Encryption</td>
<td>24–25</td>
</tr>
<tr>
<td>Engineers’ education, element of SDL</td>
<td>17–18</td>
</tr>
<tr>
<td>implementation</td>
<td></td>
</tr>
<tr>
<td>Enumerations in Java</td>
<td>256</td>
</tr>
<tr>
<td>EL-76</td>
<td>5</td>
</tr>
<tr>
<td>Encase</td>
<td>80</td>
</tr>
<tr>
<td>EPIC architecture</td>
<td>214, 215, 264, 266, 268</td>
</tr>
<tr>
<td>ERATO (experience, retrospective, analysis, theory, oncoming perspectives) teaching paradigm</td>
<td>9, 208–211, 252</td>
</tr>
<tr>
<td>Error code</td>
<td>139–140, 153</td>
</tr>
<tr>
<td>Evidence</td>
<td>34–35, 37, 46</td>
</tr>
<tr>
<td>Exception handling</td>
<td>33, 36, 38, 39, 46, 136–140</td>
</tr>
<tr>
<td>in Java</td>
<td>254</td>
</tr>
<tr>
<td>FAT</td>
<td>261, 263</td>
</tr>
<tr>
<td>FEAT</td>
<td>72–73</td>
</tr>
<tr>
<td>Floyd–Hoare’s program calculus</td>
<td>26, 219, 226</td>
</tr>
<tr>
<td>Formal methods</td>
<td>2, 4, 26, 27, 40, 47–49, 218, 219, 226</td>
</tr>
<tr>
<td>specification(s)</td>
<td>2, 4, 26, 218–219, 224–226, 229, 246</td>
</tr>
<tr>
<td>verification</td>
<td>2, 4, 218–220</td>
</tr>
<tr>
<td>FORTRAN</td>
<td>2, 137, 140, 264</td>
</tr>
<tr>
<td>Fuzz testing (fuzzing)</td>
<td>20</td>
</tr>
<tr>
<td>FxCop utility</td>
<td>18, 20, 21, 39–40</td>
</tr>
<tr>
<td>Generative programming</td>
<td>73</td>
</tr>
<tr>
<td>Generics</td>
<td>52, 90, 96, 249, 251, 256–259 in Java, 256–259</td>
</tr>
<tr>
<td>get</td>
<td>38</td>
</tr>
<tr>
<td>Global Assembly Cache (GAC)</td>
<td>38</td>
</tr>
<tr>
<td>grep utility</td>
<td>7, 8, 62, 74</td>
</tr>
<tr>
<td>HIPO diagrams</td>
<td>226</td>
</tr>
<tr>
<td>Human interface proofs (HIP)</td>
<td>23</td>
</tr>
<tr>
<td>HyperJ</td>
<td>55, 70</td>
</tr>
<tr>
<td>hypermodule</td>
<td>71</td>
</tr>
<tr>
<td>hyperslice</td>
<td>71</td>
</tr>
<tr>
<td>hyperspace</td>
<td>71</td>
</tr>
<tr>
<td>composition rules of</td>
<td>71</td>
</tr>
<tr>
<td>IBM 360</td>
<td>2</td>
</tr>
<tr>
<td>ICED-T model</td>
<td>187–190</td>
</tr>
<tr>
<td>ildasm utility</td>
<td>197</td>
</tr>
<tr>
<td>Implementation, as software life-cycle stage</td>
<td>226</td>
</tr>
<tr>
<td>coding style</td>
<td>228</td>
</tr>
<tr>
<td>Hungarian notation</td>
<td>228</td>
</tr>
<tr>
<td>coding templates</td>
<td>227–228</td>
</tr>
<tr>
<td>choosing the language</td>
<td></td>
</tr>
<tr>
<td>customer factor</td>
<td>227</td>
</tr>
<tr>
<td>first language factor</td>
<td>227</td>
</tr>
<tr>
<td>proprietary languages</td>
<td>227</td>
</tr>
<tr>
<td>self-documenting code</td>
<td>228</td>
</tr>
<tr>
<td>Inheritance</td>
<td>218, 224, 225, 253</td>
</tr>
<tr>
<td>Instant messaging</td>
<td>210, 215, 261, 263</td>
</tr>
<tr>
<td>Integrated solutions</td>
<td>239</td>
</tr>
<tr>
<td>Intentional programming</td>
<td>73</td>
</tr>
<tr>
<td>Interceptor</td>
<td>77–78, 81–82</td>
</tr>
<tr>
<td>Internet aware software</td>
<td>240</td>
</tr>
<tr>
<td>Internet Explorer InfoBar</td>
<td>24</td>
</tr>
<tr>
<td>Invariant</td>
<td>40, 41, 168–170, 183</td>
</tr>
<tr>
<td>abstract</td>
<td>224</td>
</tr>
<tr>
<td>class</td>
<td>168, 169, 183</td>
</tr>
</tbody>
</table>
concrete, 224
object, 40
Iterator, 72, 224, 244

JAC, 76
Jackson’s method (of software design), 226
Java, 42, 55–61, 63–64, 66–72, 75–79, 83–88, 90, 97, 124
annotations, 69, 76, 78, 87, 253, 257, 260
bytecode, 59, 66, 75, 77, 241, 253
class file, 46
Community Process (JCP), 252, 260
Compatibility Kit (JCK), 229
compiler, javac, 48, 61
Cryptography Extension (JCE), 45
Development Kit (JDK), 253, 259, 260
Native Interface (JNI), 43
native method, 43
notify method, 47
Public Key Infrastructure (PKI), 45
reliability issues, 46–48
Standard Edition, 42
stop method, 48
StringBuffer class, 254
suspend method, 46
teaching, 251–260
Virtual Machine (JVM), 43, 45, 46, 229, 241, 259
wait method, 47
JavaBeans, 241, 255, 257, 259
JavaScript, 21, 73, 264
Java Security Aspect Library (JSAL), 133
Javassist, 76–77
JavaTest, 231
JBoss, 76–77
JMangler, 76, 78
Join point, 52, 56, 58, 60, 61, 64, 67–69, 74, 78, 81, 85, 89, 94, 95, 97, 98, 104, 105, 106, 109, 111–114, 117–119, 122, 125, 126, 200, 203, 204, 207
JUnit, 231
Knowledge management, 251
applicability for TWC, 129–131, 135, 137, 140, 167, 186–188, 192–194
Knowledge.NET, 15, 251
Least privilege, security design principle, 17, 19
Life cycle, see also Security development life cycle
of software development, 214, 221, 228, 229, 232
waterfall model, 18, 202, 221
Lint, verification tool, 21, 48
Linux, 214, 261–263
LISP, 244
Logging, 3, 67, 69, 92, 101, 107, 116, 123, 157–161
LogicAJ, 76
LOOM.NET, 80–81
Maintenance, as part of the software life cycle, 231, 232, 234, 235
bug fixing, 27–30, 231–232
bug report(s), 232–233, 237
recommendations to software teams, 27–30
relation to AOP tasks, 7, 56, 62, 63, 70, 90, 165–168, 232
Managed code, 32. See also .NET
Managed execution, 32, 37–42, See also .NET
Managed pointer, 32. See also .NET
Manufacturing, as part of the software life cycle, 234, 235
product documentation, 234
source code control systems, 234–235. See also RCS, SCCS, Teamware
stages of shipping the product, 234
version numbers, 234
mdbg—Microsoft Managed Debugger, 83
Mean time between failures (MTBF), 25–26
Memory management, 215, 242, 249, 251, 259, 262, 263
Metadata, 32, 38, 39, 42, 43. See also .NET
Metric, software
  Halstead’s, 25
  McCabe’s (cyclomatic complexity), 25
Microsoft
  Developer’s Network Academic
  Alliance Curriculum
  Repository (MSDNAA CR), 9,
  104, 209, 212, 239, 262, 265,
  266, 268
Outlook, 216
SQL server, 242
Midlet, 257, 259
Minimal attack surface, security design
  principle, 19, 20, 35
Mixin, 78
Mobile devices
  software, 240
Model–view–controller (MVC)
  paradigm, 226, 256
Modula, 253
Modular programming, 3–5, 214, 218,
  222, 268
Module(s), 2–5, 51–53, 55–57, 65, 70
  arguments, 51
  cohesion, 217, 222
  coupling, 217, 222
  exceptions, 51
  implementation, 51
  interface, 51
  results, 51
  semantic net of, 52
  semantics, 51
  side effect, 51
  unit of programming knowledge (or
  independent idea), 52
Monitor, 211, 215, 262
Mono, 31, 241, 251, 268. See also
  .NET
MS-DOS, 6, 261–262. See also
  Operating systems
Multicore hardware, 214, 264, 268
Multithreaded (MT) safety, 2, 52,
  145–149
Multithreading, 46–47, 145, 211,
  214–215, 249, 251, 252, 254,
  260, 262
  in Java, 254
Mutual exclusion, 47, 142, 146, 148
.NET, 31–34, 39, 40–47
  Active Oberon (Zonnon).NET, 32
  as a factor for compiler development,
  264
  as a platform for AOP, 87
common language infrastructure
  (CLI), 31
common language runtime (CLR),
  32, 213, 241, 242, 250
common language specification
  (CLS), 31, 33, 213, 215, 243,
  247–248
common type system (CTS), 31, 33,
  250, 267
course, 238–249
C#, 31
course, 238–249
Eiffel.NET, 32, See also Eiffel
  Framework, 61, 204, 242, 243
  basic class library (BCL), 243
JScript.NET, 32
J#.NET, 32
Managed C++ .NET, 32
Microsoft Intermediate Language
  (MSIL), 32
shared-source common language
  infrastructure (SSCLI), 31,
  250
Visual Basic.NET, 32
.NET 3.0, 42
NetBeans, 210, 215, 258–259
  C/C++ pack, 258
Network
  heterogeneous, 241
Networking
  in Java, 256
  ISO/OSI model, 214, 263
Never install, option, 24
NullPointerException (in Java), 254. See
  also Exception handling
NullReferenceException (in .NET),
  39. See also Exception handling
OBJ, algebraic specification language,
  26, 211, 219, 225, 226
object invariant in Spec#, 40
object-oriented programming (OOP), 5,
  55, 61, 63, 70, 73, 75, 86, 89, 96,
INDEX

124, 176, 209, 218, 224–225, 245, 247–248, 254
pitfalls, 5, 55, 218, 224, 225
operating system(s) (OS), 2, 9, 11–12, 14, 16–17, 33, 51, 124, 188, 208–209, 214, 216–218, 220, 260–263, 266, 268, 271
Elbrus, 238, 260–264
Linux, 214, 261–263
Solaris, 25, 214, 217, 261–263
THE, 50
Optimization, 214, 215, 265, 266–268
Paging, 215, 261, 263
Parsing, 214, 264–266
Pascal, programming language, 3, 5, 6, 124, 137, 140, 213, 222, 224, 227, 244, 248, 256
Pascal, Sun compiler, 27–29, 213
Pharming, 13, 23
Phishing, 13, 23, 24
Phoenix, 83, 92, 95, 97, 103–105, 122, 125, 264, 270
Academic Program, 210
AddNopTool / MAddNopCounter
sample, 104, 122, 161
High-level intermediate representation (Phoenix HIR), 97
Phx.Morph, 80, 83. See also Wicca
Pointcut, 56, 60, 64, 67, 68, 74, 78, 81, 85, 86, 87, 89, 90, 97, 98, 124, 126
designator, 56
named, 56, 67–68, 126
Portable executable (PE) file, 32. See also .NET
Postcondition, 40, 41
Postobject programming, 63
PostSharp, 80–82
Laos, weaver, 81–82, 133
Precondition, 26
PREfast, security tool, 21
PREfix, security tool, 21
Principal, 36, 37
PrincipalPermission, 36, 37. See also .NET.
Privacy, 11–15, 23–25
Process, 142, 214, 243, 255, 260, 262, 263
lightweight, 214, 255, 260, 262
PROLOG, 86, 87
Property, 38, 44. See also .NET
Pseudocode, 3
Public key token, 33. See also .NET
Puzzle.NET NAspect, 80
Python, 32, 73, 264
Race condition, 142, 143, 146, 148
Rapid prototyping, 3
RCS, 235
Refactoring, 84, 103, 124
Reference types, 248–249. See also .NET common type system
Reflection, 38–39, 42, 46, 68, 77, 119, 158, 161, 200, 249, 250, 253, 255
in Java, 253
Reflex, 76
Regedit, utility, 25
Reliability, 11–15, 25–27
methods of design and implementation, 26
methods of formal specification and verification, 26
methods of software testing, 26
Repeatable process, as element of SDL implementation, 18
Representation, procedural, 38
RISC architecture, 214, 266
Rotor, see .NET, shared-source common language infrastructure
Ruby, 32, 73, 264
Sandbox model, 44–45
SCCS, 234–235
SDC software process principles, 17–18, 221
SDL, algebraic specification language, 26–27, 211, 219–220, 225–226
Seasar, 76
Seasar.NET, 80
Secure by default, software process principle, 18
Secure by design, software process principle, 17
Secure coding templates, 20
Secure in deployment, software process principle, 18
Secure software engineering, 217, 221, 225, 228, 238
Security, 3, 12, 15–23
annotation framework (SAF), 133
central team, 19
classification and analysis, 16
code access, 34–35
declarative style in .NET, 35
development life cycle (SDL), 17–19, 22, 221, 226
evidence-based, 34
final review (FSR), 22
for home users, 16
for office users, 17
for software developers, 17
imperative style in .NET, 36
in Java, 44–46
manager (in Java), 45
policy, 35, 37, 45
product advisor, 19
provider (in Java), 46
push, 21–22
quantitative assessment, 16
role-based, 34–37
stack walk in .NET, 36, 45
technologies and tools, 15
user-group model, 35
SecurityException, 36, 45
sed, 186
Semaphore, 89, 143–145, 160, 211, 214, 215, 229, 262
P operation, 89
V operation, 89
Serialization, 242, 243, 255
set, 38. See also Property.
Signal (in CLU), 211, 224, 254
Simplify, theorem prover, 41. See also Spec#
SIMULA-67, 73, 213, 224–225, 248
SmallTalk, 224, 225
SmartScreen, 23
SNOBOL, 244
Solaris, 25, 214, 217, 261–263, 266
Spam, 13, 23–24, 150, 216, 237
filter(s), 12, 23, 216
SPARC, 213–214, 266
SPBU.NET project, x, 208, 209, 212–215, 268
Spec#, 27, 39–41, 219, 224–226, 251
Spring framework, 76, 133
Spring.NET framework, 80
SQA engineer(s), 230, 236
SQE engineer(s), 230, 231, 236
SQFD model, 190–193, 225–226, 236
Statement, 3
goto, 3, 222, 230
if, 3
while, 3
Stepwise refinement, 3, 221, 226
Strictly typed language(s), 244. See also Type
STRIDE, threat categorization scheme, 17, 221
Strong name, 33, 35, 40
Structured programming, 3, 26, 214, 218, 221–222
Subject-oriented programming, 71
composition rules of, 71
Sustaining, See Maintenance
Swing, 256
pluggable look-and-feel (PLAF), 256
Synchronization, 2, 3, 4, 8, 214, 215, 255, 260, 262
event-type, 143
resource-type, 143
TeamWare, 234, 235
T-ERATO teaching paradigm, 215
Testing, as part of the software life
benchmarking, 229, 231. See also Testing, performance testing
black box testing, 26, 228, 229
boundary values testing, 20, 21, 26, 230
code inspection, 228, 230
code review, 230
compatibility testing, 229. See also Java compatibility kit
exclude lists, 231
golden files, 231
performance testing, 229. See also Testing, benchmarking
regression testing, 229
security testing, 20–22, 229
stress testing, 229
test base, 228, 231
test case(s), 228, 231
test coverage, 26, 228–230
test-driven development, 205, 206, 228
test harness, 124, 157, 231, 253. See also: JavaTest, JUnit
white box testing, 26, 228, 229
Threat modeling, 20
TIP (Technological Instrumental Package) technology, 7, 53, 54, 62, 226
access interface, 7, 54
concept level, 7, 54
creation/deletion interface, 7, 54
definition level, 7, 54
output interface, 7, 54
representation level, 7, 54
update interface, 7, 54
TrustSPBU.NET project, x, 208, 215–220
Trustworthiness (of software), 1, 10
Trustworthy computing (TWC), 13–16
networking issues, 13
primary issues, 11
principles, 12
sharing issues, 11
software reliability issues, 11
Trustworthy Computing Initiative by Microsoft, 1, 13–16, 215, 221
four “pillars”, 13
Type,
non-null in Spec#, 40, 41
parametrized, 224, 245, 256, 257. See also Generics
Typeless language(s), 244
UML, 65, 66, 190, 207, 226, 240, 272, 275
Unboxing, 43–44, 211, 253, 254
UNIX, 21, 33–37, 62, 186, 216, 261–263
Unmanaged code, 34. See also .NET
Unsigned types, 47–48
Usability, as the fifth “pillar” of TWC, 15
Value types, 248–249. See also .NET
common type system
Variable(s)
container approach, 248–249
object reference approach, 248–249
Versioning, 33. See also .NET
Vertical cut, 7, 53,
implementation by tangled actions, 53
implementation of extending function, 53
technology, 53
Vienna definition method (VDM), 226
Visual Studio integration program (VSIP), 85
VLIW architecture, 214, 215, 264, 268
Wasp for Java, 49
Wearable computer, 240
Weave.NET, 61, 80, 82
Weaving, 2, 54
four “blind,” 64, 73, 74, 98, 106, 112
dynamic, 58
rules, 2
static, 58
at class loading time, 59
at the intermediate code level, 59
at the object code level, 59
at the source code level, 59
during just-in-time compilation, 60
in Aspect.NET, 111
Web applications, 239, 256, 259
Web forms, 33. See also .NET
Web services, 34. See also .NET
Wicca, 80, 83. See also Phx.Morph
Windows CardSpace, 42. See also .NET 3.0
Windows communication foundation, 42. See also .NET 3.0
Windows forms, 33, 241–242, 250–251. See also .NET
Windows presentation foundation, 42. See also .NET 3.0
Windows workflow foundation, 42. See also .NET 3.0
Windows XP service pack, 24
Wrapper(s)
class, 44
dynamic (in JAC), 78
XML, 34, 35, 42, 61, 73, 75, 81, 83, 84–85, 96, 132, 188, 213, 216, 220, 240, 243, 258
configuration file(s), 77–78, 80, 84–85, 132, 188, 216, 220, 241
processing in Java, 258–259
Web service(s), 213, 251
Z notation, 26
ZFS, file system, 25