<table>
<thead>
<tr>
<th>Term</th>
<th>Page(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>abstract base classes (ABCs)</td>
<td>133–7, 146, 202</td>
</tr>
<tr>
<td>abstraction</td>
<td>81, 108</td>
</tr>
<tr>
<td>AccountHolders</td>
<td>59</td>
</tr>
<tr>
<td>ACM OOPSLA conference</td>
<td>115</td>
</tr>
<tr>
<td>actors</td>
<td>171, 181, 237</td>
</tr>
<tr>
<td>ADAPTER design pattern</td>
<td>125</td>
</tr>
<tr>
<td>advocate</td>
<td>54</td>
</tr>
<tr>
<td>Agile Manifesto</td>
<td>4, 7, 23</td>
</tr>
<tr>
<td>Agile Systems</td>
<td></td>
</tr>
<tr>
<td>Agile Systems principles</td>
<td>3–4</td>
</tr>
<tr>
<td>scope of</td>
<td>8–9</td>
</tr>
<tr>
<td>production</td>
<td>7–10</td>
</tr>
<tr>
<td>Algol</td>
<td>233</td>
</tr>
<tr>
<td>algorithm</td>
<td>220, 245</td>
</tr>
<tr>
<td>algorithmic scaling</td>
<td>144–6</td>
</tr>
<tr>
<td>antlr</td>
<td>121</td>
</tr>
<tr>
<td>AOP (Aspect-Oriented Programming)</td>
<td>99, 302</td>
</tr>
<tr>
<td>APIs</td>
<td>21, 40, 99</td>
</tr>
<tr>
<td>application generators</td>
<td>120–2</td>
</tr>
<tr>
<td>application-oriented language</td>
<td>121</td>
</tr>
<tr>
<td>architect role</td>
<td>42</td>
</tr>
<tr>
<td>architectural completeness</td>
<td>152</td>
</tr>
<tr>
<td>architectural effort</td>
<td>160</td>
</tr>
<tr>
<td>ARCHITECTURE ALSO IMPLEMENTS pattern</td>
<td>53</td>
</tr>
<tr>
<td>architecture carryover</td>
<td>128–9</td>
</tr>
<tr>
<td>ARCHITECTURE CONTROLS PRODUCT pattern</td>
<td>53</td>
</tr>
<tr>
<td>architecture testing</td>
<td>149–52</td>
</tr>
<tr>
<td>artifacts</td>
<td>294–7</td>
</tr>
<tr>
<td>as-builds</td>
<td>127</td>
</tr>
<tr>
<td>‘ask five times’ exercise</td>
<td>64</td>
</tr>
<tr>
<td>Aspect-Oriented Programming (AOP)</td>
<td>99, 302</td>
</tr>
<tr>
<td>assertions</td>
<td>137–44</td>
</tr>
<tr>
<td>assessment of architecture</td>
<td>160–2</td>
</tr>
<tr>
<td>Atomic Event architecture</td>
<td>157, 158, 206–8</td>
</tr>
<tr>
<td>domain objects</td>
<td>221</td>
</tr>
<tr>
<td>form and architecture of</td>
<td>220–6</td>
</tr>
<tr>
<td>object roles, interfaces and the model</td>
<td>221–4</td>
</tr>
<tr>
<td>one-to many mapping of object roles to</td>
<td>225–6</td>
</tr>
<tr>
<td>objects</td>
<td></td>
</tr>
<tr>
<td>use cases, atomic event architectures</td>
<td></td>
</tr>
<tr>
<td>and algorithms</td>
<td>224–5</td>
</tr>
<tr>
<td>vs DCI</td>
<td>286–7</td>
</tr>
<tr>
<td>autonomy</td>
<td>91</td>
</tr>
<tr>
<td>of third-party software</td>
<td>125–6</td>
</tr>
<tr>
<td>autopoietic systems</td>
<td>68</td>
</tr>
<tr>
<td>BabyUML environment</td>
<td>268, 269, 275</td>
</tr>
<tr>
<td>BabyUML project</td>
<td>301</td>
</tr>
<tr>
<td>base class object</td>
<td>104</td>
</tr>
<tr>
<td>baseline</td>
<td>179</td>
</tr>
<tr>
<td>behavior</td>
<td>45, 245</td>
</tr>
<tr>
<td>Behavior Driven Development (BDD)</td>
<td>46, 57, 175</td>
</tr>
<tr>
<td>behavior modeling</td>
<td>46</td>
</tr>
<tr>
<td>Bell laboratories</td>
<td>88</td>
</tr>
</tbody>
</table>
Big Architecture Document 62

bison 121

blit 88

block worlds 218

Book Publication Data 128

brainstorming 176, 181

BUFD acronym 159–60

build/buy tradeoffs 74

business rules 191–3

C 114, 138

C# 118, 156, 158, 251

account example 315–19

DCI in 299

C++ 95, 114, 156, 300

abstract base classes in 134, 136, 146,
158, 240, 251, 252

assertions in 138, 140–2

coding 257–9

commonalities and variations for 118,
119, 120

member function 143

object role traits 288–90

programming language

extension 148–9

traits in 254–5

caring 19–21

case methods 157

CASE tools 85, 163

case-effect charts 177

classification 88

class objects 126

COBOL 286

code 17–18

design and 33–4

rough framing 131–52

code base 79

code readability 245

coder label 55

cohesion 92, 102

collaborative game 177, 187

commonality categories 112–14, 245

commonsense 19–21

compression 108, 109–10, 245

Concept Maps 47

conservation 84

consistency 84

consolidation 176

Context 47, 108, 109, 157, 166, 173

Context classes 243–6

Context framework 261–83

C++ code 265–7, 269

Ruby code 263–4, 269

Context object 153, 157–8, 244–6

in user mental model 290–4

contract approach 144, 184

convergent thinking 176

Conway’s Law 49, 56, 65, 85, 90–3,
95, 96, 106, 217, 236

CORBA 122, 150

cost

development 50

discovery 85

as end-user value 37, 38

schedule tradeoffs 74

software development 85

coupling 92, 102

CRC (Candidate object, Responsibilities
and Collaborators) cards 209,
229–31, 233

creeping featurism 75

Curiously Recurring Template

Idiom 288

customer engagement 15, 23, 60

customers 50–2

contrasted with end users 50–2

in the Value Stream 52

data structure 113

database transaction logs 202

Data-Context-Interactionarchitecture see

DCI architecture

dataflow diagrams (DFD) 174, 178

DAY CARE pattern 19, 110

DCI architecture 9–10, 34, 157, 158, 171,
220, 235–303
artifacts 294–7
basic 239–40
Context classes 243–6
Context framework 261–83
context layering 283
Context Objects in 290–4
design inputs 246
documentation 300–1
Domain Logic 285–90
equivalent 246–85
history 301–3
identifiers 250–3
information hiding 283–4
inputs to design 246–7
methodful object roles 240–2, 253–61
methodless object roles 250–3
in a nutshell 236–8
MVC and 238
overview 238–46
in Ruby 297–8
selective object role injection 284
traits 242–3
use cases 247–50
variants 283–5
decision tables 177
definition of architecture 2, 81–2
deliberation 83
derived class object 104
design by contract 138
design for testability (DFT) 57
design style 100–2
commonality and variation 104–5
commonality, variation and scope 108–11
explicit commonalities and variations 111–14
structuring vs partitioning 102–3
tacit commonality and variation 105–8
design technique 101
design, code and 33–4
designer label 55
developer role 42
developers 55–6, 169–70
DEVELOPING IN PAIRS pattern 110
development costs 50
deviations 196
direct manipulation metaphor 85, 114, 207
discovery cost 85
disposable analysis 210
divergent thinking 176
DIVERSE GROUPS pattern 100
documentation 16–19, 21, 127–9, 162, 162–3, 209–11, 231–2
DCI 300–1
problem definition 76
Dolphin 218
domain class method elaboration 287
domain dictionary 128
domain expert 42
DOMAIN EXPERTISE IN ROLES
pattern 110
domain experts 52–5, 95, 117, 150
domain knowledge 95
domain logic in DCI 285–90
domain members 119
domain method 157
domain modeling 46
domain-specific language (DSL) 121, 129
state of art in 121
place in architecture 121–2
domains 94–6, 118
Dorado 218
Dove 218
doxygen 232
DynaBook 215, 217
dynamic component of architecture 82
Eiffel 144
ElectroCard 93, 94, 97, 103, 111, 114, 117, 131
embedded code 210, 216
enabling specification 170
end user cognitive model 44
end user expectations 43–4, 60
end user landscape 47
end user mental model 31, 32, 45, 46
end users as value stream anchors 36–41
end users role 43–7
Index

envisioning 176
envisaging 176
evolution 104
Excel 174

FACADE patterns 148
factoring 226–31, 287
FACTORY METHOD 148
failure-proof constraints see Poka-Yoke
families 96, 118, 119
FAST system 129
fault-tolerance software patterns 123
fear of failure 82
feature lists 173–4
feature testing 45
feedback 23, 49, 60, 69
focus 84
forces 122
form 10
form 30–2, 44, 45, 81
  vs accessible services 146–7
  foundations of 30–2
forming stage 65
FORTRAN 286
FRESH WORK BEFORE STALE
  pattern 100
function, as end-user value 37
functionality 10, 165–211

genchi genbutsu 56, 60
genralization 108–9
GET ON WITH IT pattern 87
graphical interface 216
GUI 62, 93

habits 128, 182–5, 225, 277–83
HAS-A relationship 153
history 24–5
  of DCI 301–3
  of computer science 218–20
  of use cases 211
HOLISTIC DIVERSITY 36
Honda 39
IBM 64, 163
impediment list 75
implementation procedures 122
increments 187

information nicknames 128
infrastructure organizations 110
interactive code 216
interactive code 216
invariant 104
Java 92, 114, 138, 156, 158
  DCI in 299–300
  documentation 300
Javadoc 232
Joint Application Design (JAD)
  24–5
jUnit 86
just-in-time delivery 22, 37, 197,
  288, 289

known knowns 176
known unknowns 176
Knuth’s Law 22

law of continuity 152
Laws of Demeter 92
Lean
  goals 132
  definition 1–2
  management 1–2, 22–3
  principles 2–4
  feature development 4–6
  overlap with Agile 11–14
Lean Secret 2, 14, 27, 38–41, 57, 61, 96,
  160, 185
line management 49
Liskov Substitutability Principle
  (LSP) 144
Lisp 55
locus of attention 88
lost practices 14–21
Lucid 55

main success scenario 180
maintainable code 205
maintainer label 55
managers 48–9
market objects 126
methodful object roles 240–2
methodless object role types 224, 240,
  250–3
Model-View-Controller (MVC)
  architecture 16, 123, 124, 129, 153, 158, 207, 233
  APIs 217
Model-View-Controller-User
  (MVC-User) 116, 213, 214–20
  modules 102, 106
  motivation, user’s 175
  Motorola 64
  mouse 219
  myths, Agile 86–7
  narratives 174–5, 177–8
  network of stakeholders 61–6
  nicknames 126, 184
  NIH (Not Invented Here) 124
  norming stage 65
  noun–verb interaction style 207
  object composition 153
  object orientation 115–17
  object relationships 158
  object roles 201, 202, 232, 237
  Objective-C 251
  object-oriented architecture 47, 155–8
  object-oriented design techniques 46
  object-oriented programming 29, 115, 206–8
  OOPSLA 163
  ORGANIZATION FOLLOWS
    LOCATION pattern 58, 66, 96
  ORGANIZATION FOLLOWS MARKET
    pattern 58, 66, 97
  overconstrained problems 73–4
  paradigms 101
  parallelism 56
  parameterization 95
  parameters of variation 119
  Pareto’s Law 22
  partitioning 88–100, 102–3
    architecture and culture 100
    complexity of 93–4
    dimensions of complexity 94, 96–9
    domain form vs behavioral form 89–90
    domains 94–6
  see also Conway’s Law
  PatientKeeper 19
  pattern languages 122–4
  performing stage 65
  Personas 171–2, 174, 175, 178
  phased program planning (PPP) systems 63
  pipelining 56
  placement 93
  PloPs (Pattern Languages of Programs) 123
  plumbing 147
  poka-yoke 83–4, 133, 138, 226
  polymorphism 244–5
  POSA series 123
  post-conditions 137–44, 187
  pre-conditions 137–44, 187
  problem definition 23, 29, 67–78
    Agile and 68
    documentation 76
    good and bad 70–2
    Lean and 68–9
    problems and solutions 72–3
    problem ownership 74
  Product Backlog 40
  Product Backlog Items (PBIs) 75
  product line 95
  Product Manager 35, 49
  Product Owner 40, 48, 49, 75
  programming by difference 104
  Project Manager 35
  prototypes 46, 59, 60, 176
  PROXY design pattern 125
  ‘pull’ 83
  ‘push’ 83
  Python
    account example 311–13
    DCI in 299
  Qi4j 268, 299, 327–9
  readable code 205
  RECOMMITMENT MEETING
    pattern 75
  re-factoring 86, 87, 226–31, 287
  relationships
    kinds 153–5
relationships (continued)
  testing 155
RESPONSIBILITIES ENGAGE
  pattern 58
return on investment (ROI) 37, 47, 54
reverse engineering 18
role methods 11
roles 156, 157, 221
route 132
routing 93
RPG 286
Ruby 133, 251, 284
  account example 321–6
coding 259–61
traits in 254–5
RUP (Rational Unified Process) 57, 211
scaffolding 147–9
Scala 148, 284, 285
  DCI and 298, 307–10
methods 113
scenarios 174, 175
scope 48, 186
  trimming 185
Scrum 12, 13, 15, 21, 23, 40, 51, 59, 63,
  178, 179, 232
  Problem Ownership 75
ScrumMaster 36, 48–9
selections 221
self 226
self-organization 23, 48, 68
Service-Oriented Architecture (SOA) 21
SHRDLU 218
Simula 29
Simula 67 programming 115, 233
Smalltalk 114, 115, 156, 163, 207, 251,
  275, 300
documentation 300
  traits in 253–4
sneak paths 155
solution domain experts 54–5
sort function 102
space/time tradeoffs 74
SQL 125, 126
Squeak, account example in 331–9
stakeholder analysis 178
stakeholder engagement 15–16, 27–9,
  35–66
  process elements 57–61
stakeholder relationships 62
stakeholders, key 41–57
state machines 177
STATE patterns 148
static clinging 142–4
static component of architecture 82
statistical process control (SPC) 24
storming stage 65
Storyboards 176, 178
stovepipe versus swarm 61–4
structure 10, 196
structure-preserving transformation 123
structuring 91, 102–3
subject matter experts 95
subsystems 90, 102
suitability of third-party software 126–7
Sunny Day scenario 187–93
decision-making 204
emerging requirements 204
input to architecture 205
release planning 204–5
testing 203–4
understanding of what to
develop 205–6
user’s workflow 203
SURROGATE CUSTOMER pattern 172
tacit knowledge 65
TAKE NO SMALL SLIPS pattern 75
teamwork 65
TEMPLATE METHOD pattern 95
Test-Driven Development (TDD) 86–7
testers 55–7
Theory X management 301
Theory Y management 301
thinking 19–21
third-party software 124–7
time, as end-user value 37, 38
tools 229–31
top-down design 88
Total Productive Maintenance (TPM) 24
Toyota Production System (TPS) 24, 25
Toyota Way 1, 4, 12, 24, 41
traits 11
approach 242–3
as building blocks 253
in C++ 254–5
in Ruby 256–7
in Smalltalk 253–4
trust 64
UML 21, 89, 211
uniformity of third-party software 125
UNIX systems 101, 138
unknown unknowns 9
usability
    foundations for 16
    testing 149, 208–9
usage narratives 174
use cases 167–9, 178, 181, 188, 200–5
    conversation in 188
    as decision framework 177
essential 188
    roles from 201–2
    structured 188
user expectations 8, 81
user mental model, context objects
    in 290–4
User Narratives 11
User Profiles 171, 172, 178
user roles 171–2, 173, 221–2
user stories 36, 166–7, 170–1, 179
user workflow 178
validation 45
value stream 35–41
    architecture in 37–8
    end users as anchors 36–7
variation 104
volitive world model 241
Von Neumann programming 89, 113, 117–20, 221
'why' of software 19
'wicked problems' 72
XML 126
XP 13, 15, 21, 31, 49, 211
yacc 121, 122
YAGNI acronym 159–60