abstract behavior, 62
abstract classes, 16, 58–59, 206
abstract data type (ADT), 23, 62–63, 66, 67, 70
abstract interfaces, 16
abstraction, 167
in code, 85
in documentation, 205, 206–207
in programming languages, 12
in software reuse, 127
abstraction relation, 196, 200, 201
abstract models, 62, 131
acceptance rule, 138
action classes, 117
Ada, 2, 18, 58, 95, 97, 98, 106
components in, 104
definitions in, 67
generic packages in, 64
Standard package of, 4
structure graphs (ASG) in, 101
ADAGE, 2, 79
adept system engineers, 31, 32, 37, 39
ADL. See architecture description language
ADT. See abstract data type
Advanced Distribution Simulation Program, 87
Aesop, 84, 85, 87, 88, 91
aggregation, 57, 189
in module expressions, 6
algebraic systems, 64–65
amortization, 140–141
Anna, 2
anomaly detection, 90
Answer Garden, 131–132
API. See application programming interfaces
Application Domain-Based Testing, 114–123
application domains, 124, 149
types of, 31
application frameworks, 16, 20, 85, 205
application generators
See generators
application objects, 95, 96, 98
application programming interfaces (API), 97, 98, 100
application view, 97, 98
architectural descriptions, 85, 86–87
architectural design, 85, 148
architectural models, 102, 103
architectural reuse, 84, 87, 97
in parameterized programming, 6–7
and software architecture, 86–87
architecture description language (ADL), 2
See also module connection language
architecture-oriented decomposition, 106, 108
architectures
defined, 3
and validation, 80
architecture views, 97, 98
Army, U.S., 94, 97, 99
ARPA, 2
asynchronous typed datastreams, 86
attribute grammars, 77
attributes, 181, 189–190
Automated Cartridge System (ACS), 114, 115, 117
avionics systems, 7, 73, 79, 85
Avoca, 169
base classes, 15, 16, 205, 207
basic patterns, 34, 35, 37
behavioral subtypes, 195
behavior-centered systems, 37–38
best of breed (DoD), 27
Biggerstaff's Rules of Three, 124
bindings, 108, 179
bind interface, 105
black-box reuse, 63, 114, 195, 199, 203
BMS, 180
book rate of return, 140–141, 143
BORE, 125, 127, 130
brokers, 156, 162–163
BRRR, 192
building block method, 105–107, 106, 108–110
building block reuse, 110
business functions, 20, 31–34, 38, 94
business objects, 95, 96, 98, 102
C, 17, 18
C++, 13–18, 55, 58, 170, 180, 197
class templates in, 64
and reuse, 13–18
CAMP project, 104
CAP. See Cartridge Access Port; common
application platform
Capability Maturity Model, 22, 27
capital rationing, 144
CARE. See computer aided reuse engineering
Cartridge Access Port (CAP), 114
CASE, 94, 101, 124, 125, 128, 129, 131
case-based repository, 127-128, 131, 132
case-based tools, 125, 126-127
case links, 130
case searches, 129
cash flow analysis, 137
category browser, 190
category theory, 2
class designers, 195, 203
classes, 95
and C++, 13, 14
hierarchies and, 13, 15, 16, 188
inheritance and, 174
and reuse, 84, 188
user defined, 57
ClassExpert, 188-193
classification systems, 86, 192
class templates, 199, 201
Clear specification language, 2
CLOS, 174
CMM level 3 project management, 25
COCOMO model, 20, 48
code inheritance, 195-203
code reuse, 11, 53, 84, 86
code segments, 56, 64
cognitive mechanisms, 148
command and control domain, 85
command language syntax, 114
commercial off the shelf software (COTS), 138–139
common application platform (CAP), 107
common object model (COM), 174
common patterns, 34, 35, 37
communication domain, 177
communication mechanisms, 105
compile time binding, 176
compile time calls, 108
compile time compatibility, 56, 107
compile time composition, 170
compliance tests, 62, 64, 66
component compositions, 72-80, 85
component dependencies, 54
component design, 20, 38, 147, 195
component documentation, 146
component extraction, 147
component identification, 148–149, 150
component implementations, 202
component integration, 146
component interfaces, 105, 166
components, 73, 101-102
automatic specialization of, 127
defined, 94
and generators, 72, 171
reuse of, 12–13, 53, 54, 59, 105, 151-153
and software architecture, 86
composition, 58, 168
computationally distinguishable, 65, 66, 70
computationally reachable, 65, 66, 70
computer aided reuse engineering (CARF), 147
Computer Software Copyright Act, 161
conceptual integrity, 110
conceptual models, 52, 62, 70
configurability, 107, 108
configuration binding, 178, 183
configuration file, 176–177, 184
configuration language, 176–185
configuration rules, 95
configurations
and software architecture, 86
and testing, 121
See also patterns
congruence relations, 64
connectors, 86–87, 89, 90, 101-102
defined, 94
and software architecture, 86
consistency checking, 74–75, 79
See also validation
constraint retention, 92
constraints, 94, 102
construction sets, 105, 106, 108, 109
constructors, 64, 71
consumers, 42, 44, 156, 162
container classes, 58
container data structures, 74, 167
containers
and algorithms, 17–18
data structures as, 72–80
context, 86, 149, 150
context code, 58, 59
context dependencies, 54–55
context reuse, 54, 54, 57, 59
continuous incremental refinement, 130
contract dependencies, 55
contracts, 156, 162–163
controllability, 63–64, 66, 67, 70, 169
control specifications, 102
copyright protection, 158–159
CORBA, 72
cost, 43–46, 48, 142
cost drivers, 48
cost estimation, 138
cost of reuse, 45
COTS. See commercial off the shelf software
cumulative discounted cash flow (CDCF), 142
cursors, 74, 170
customizability, defined, 55
data abstractions, 97, 108
Database Description Language (DDL), 101
databases
  and data integrity, 99
  and generators, 169
  and layout compatibility, 14
  system programming, 3
data-centered systems, 37–38
data collection methods, 48
data dictionaries, 102
data elements, 41, 46
data flow diagrams (DFD), 102
data manipulation, 108
data mapping, 99
data-oriented approach (DOA), 37–38
data structures, 73, 84
data view, 97, 99–100
DaTE, 79
DCE. See Distributed Computing Environment
decomposition, 84, 108
Deconomic models, 48–50
Defense, Department of, 94, 97
Defense Information Infrastructure, 94
Defense Information Systems Agency (DISA), 94, 97
Delivered Source Instructions (DSI), 110
dependencies, 54–55, 57
  and component reuse, 12, 13
  in parameterization, 17
  and reuse, 53
  and software development, 8
derived classes, 15, 16, 34, 35
design decisions, 62–64, 63, 70
design elements, 84, 89
Designer Assistant, 131–132
Design Maintenance Systems, 86
design patterns, 88–89
design processes, 31, 32, 35
design reuse, 11, 84, 86, 92–93
design rule checking (DRC), 72, 74, 75–76, 78–80, 87
design rules, 127, 168, 172
development/maintenance view, 99, 101
development practices, 130–131
development processes, 39, 132
DIANA, 6
DISA. See Defense Information Systems Agency
Distributed Computing Environment (DCE), 95, 100
distribution view, 97, 100
DOA. See data-oriented approach
documentation, 146–154
documentation reuse, 205–214
domain abstractions, 125, 167n2
domain analysis, 22, 25, 31, 37, 96, 105, 106, 114, 131
  in business, 31–39
  and generators, 168
  a priori, 114–118
  process tailoring and, 27–28
  reverse engineering in, 118–119
  and specification patterns, 35
domain analysis method, 32–34
domain based testing. See Application Domain Based Testing
domain engineering, 105
domain independence, 77
domain-independent components, 94, 95
domain knowledge, 8
domain lifecycle, 124, 125–126, 126, 131, 132
domain maturity, 124, 125–126
domain modeling, 31, 37–38
  in reuse economics, 48
domain models
  and design rules, 74
  and generation, 166
  and system testing, 114–123
domain-specific architectures, 85
domain-specific components, 94, 167
domain-specific design, 38–39, 84, 124, 125–126
decisions and, 72
domain-specific frameworks, 96
domain-specific reuse, 95, 105
domain-specific software architecture (DSSA), 2, 85, 87, 107
DRACO, 79, 85
DRC. See design rule checking
dreck, 78, 80
dynamic analysis, 147
dynamic behavior rules, 119
economic models, 20, 41–50, 48
economics, 20, 41, 43–46, 48, 136, 144
editors, 6, 190
EDLC. See Evolutionary Domain Life Cycle
efficiency and reuse, 12, 13
Eiffel, 151, 202–203
enabling reuse, 176–185
  and classes, 57, 59
  and design rules, 173
  and generators, 72, 74
  and parameterization, 6, 7
entity-relationship diagrams, 102
enumerator interface, 68–69
ESPRIT, 176n1
estimate software complexity, 23
evolution and traceability, 3, 7–8, 183
Evolutionary Domain Life Cycle (EDLC), 31
experience, 105, 131
explicit dependencies, 55, 57
extendibility, 97
extensibility, 79, 84, 211
external environment interface, 97
External Interface Manager (EIM), 99

Federal Information Processing Standards (FIPS), 97
Ficus, 169–170
finite state machine systems, 177, 184
FIPS. See Federal Information Processing Standards
flexibility, 55, 56, 97, 176
FOOPS, 8
Fortran, 18
frame editor, 190
frameworks, 3n1, 72, 84, 94
in applications, 16, 20, 85, 205
component-based, 126
documentation and, 188
in economic models, 41
and generators, 173
object-oriented, 59, 72
one-class, 192
and packages, 146, 148–151, 150, 152
and process reuse, 23–24
FSM. See finite state machine systems
functional behavior, 63, 66
functionality, 6, 64
function-centered systems, 37, 38
fundamental abstractions, 73
generality, 12, 55, 169
generation, 166–175
generators, 72–80
and architecture development, 84–85, 92
and testing, 119
generic frames, 190
generic module interfaces (theory modules), 4
generics, 57–58, 89, 105, 106, 108
Genesis, 72, 77, 78, 169
GenVoca generators, 72–80, 166–175
gift systems, 31–32
global variables, 56, 58, 59
glue, 86–87
Goal-Question-Metric (GQM) paradigm, 137
goal-question-metric paradigm, 25
grammars, 72, 73, 77
See also lexicon; vocabulary
graphical user interface (GUI), 98, 101, 114, 139

Halstead method, 23
high level modelling, 178, 180–182
horizontal components, 5–6, 7, 95, 140, 143
Host Software Component (HSC), 114, 121
HTML, 8, 151
hyperdocuments, 151–153
hypergraph models, 8
hypermedia, 146–154, 152
hyperprogramming, 3, 8, 9
hyperrequirements, 8, 9
hypertext, 91, 192, 205
IBIS. See Issue-Based Information System
IEF tool, 129, 130
implementation hierarchy, 16
implementation issues, 149, 202
implication operator, 78
implicit dependencies, 55, 57
import interface, 105
inference engine (IE), 139
informal dependencies, 55
information hiding, 4, 106, 205
inheritance, 3, 5, 58, 58, 106, 118, 181, 189, 195–203
in architectural styles, 6, 7
and context reuse, 59–60
and documentation, 205, 206–207, 208
multiple, 214, 217
INRECA, 192
Inscape environment, 72, 74–75, 79, 80
instance frames, 190
instance specification patterns, 34, 35, 37
instantiation, 5, 54, 57, 74–76, 84, 90–91, 166, 206
Integrated Software Engineering Environment (ISEE), 101, 102
intellectual property, 156, 158, 161–163
interfaces, 74, 98, 105, 114, 178, 192
in C++, 16
and component reuse, 12, 13
and conformance, 58, 58, 59
design of, 56
guidelines for, 125
and objects, 166
toolkits for, 85
internal rate of return (IRR), 141–142, 143
International Business Machines Corporation (IBM), 141
ETVX format of, 22
tape library systems of, 121
invariants, 202–203
investment analysis, 137, 138
Issue-Based Information System (IBIS), 39
Iterators, 18, 55

Japan gift season, 32
Java, 15

Kalman filter, 7, 89, 90, 91
Key process areas (KPA), 27–28
Knowledge, 53, 105
Acquisition of, 130, 190
Dissemination of, 128–130
And problem solving, 124–132
Knowledge-based tools, 188–193

Language features, 52–80
Language independence, 18, 119, 211
Languages. See individual languages;
Programming languages
Larch, 64
Large scale software reuse, 84, 104–111, 147
LaSSIE, 192
Layered structures, 106, 108, 181
Layers, 66, 73, 75, 167
In applications, 98
See also components
Layer specific operations, 168, 168–169
Legacy code, 52, 99, 100
Legal issues, 156–163
Lex, 85
Lexicons, 41, 42, 46
See also grammars; vocabulary
Libraries, 17, 18, 59, 72, 80, 104
And components, 177
As components, 12
RAPID Center, 44
And reuse, 188
Robot tapes, 114–115
Library Storage Module (LSM), 114, 115, 117
Licenses, 163
Life cycle, 24–25, 31, 180
LIL, 2, 80
LILEANNA, 2, 5–7, 80
Linear documentation, 151
Lines of code, 23, 42
Liskov Substitutability Principle (LSP), 195, 196
List abstractions, 197–198
Literate programming, 205–214
Logical database models, 102
Logical resource management (LM), 108
Loral Federal Systems, 2, 94

Macros, 54
MacSTILE, 86
Maintenance, 205

And reuse economics, 44, 49, 50
Makefiles, 6, 178n1, 179, 183, 184
Management Information Systems (MIS), 94–103
Mathematical models, 62–63
Mathematical theory
Versus program specification, 64
Matrix analysis, 46–47
McCabe method, 23
MCL. See module connection language
Megaprogramming, 3n2
Memphis Environment, 151
Method invocation, 85
Method wrappers, 174
Micro-architecture, 52, 56
Microprotocols, 169
Military Occupational Specialty (MOS), 100
MIS. See Management Information Systems
Mission area applications, 97
Mobile robotics domain, 85
Model-based specification, 195, 197
Modification, 188–193
Modula-2, 55
Modularization, 25
Modular verification, 68–69, 202
Module connection language (MCL) 2
Module expressions, 2, 6–7, 8
Module graphs, 2, 3–5
Module interconnection languages (MIL), 174,
184–185
Module interface languages, 85
Module-scale components, 86
MOTIF, 98
Name spaces, 16, 56–57
Naming rules, 34
National Institute of Standards and Technology
(NIST), 97
NATO reuse standards, 142–143
Net present value (NPV), 138–139, 141–142,
143
Network Management Offices (NMO), 99
NIST. See National Institute of Standards and
Technology
Novice system engineers, 37, 39
Noweb, 211–213, 214
Numeric libraries, 14
Object classes, 106, 108
Object element classification, 115
Object Element Glossary, 116
Object extraction, 34
Object hierarchy, 116, 118
object oriented analysis/design (OOA/OOD), 115
object-oriented design
patterns and, 85
object-oriented development, 8, 15, 54, 80, 131, 149, 188–193, 205–214
classes and, 57
and inheritance, 193
object-oriented frameworks, 59, 72, 84
object-oriented patterns, 85
OBJ system, 6, 80
obligations, 74–75
observability, 62–63, 66, 67, 69
observational satisfaction, 4n4
observers, 64
OORAM, 179, 180
Open Hyperdocument System (OHS), 151
Open System Environment, 94, 97, 102
opportunity cost, 138
optional specification patterns, 34, 35
Oracle, 98, 100
organizational learning, 124–125, 126
organization memory repository, 126–130
Organon, 131
output, 42
overhead, 14, 42, 49
and component reuse, 12, 13
See also performance
P++, 170–171, 173
P2, 72, 74, 77, 78, 167, 170
packaging tools, 3, 146–154
parameter binding rules, 117
parameter instantiation, 167
parameterized components, 64, 67, 69–70
parameterized programming, 5, 12, 72, 174
in C++, 17–18, 58
parameters
instantiation and, 76
and procedures, 57, 59
and transformations, 73
partslist approach, 107, 108
Pascal, 18, 55
patents, 160–161
patterns, 37, 38, 85, 146–154
payback, 139–140, 143
payoff threshold value, 139
PCL, 176, 179–180, 181, 185
PEAKS, 22
performance
and component reuse, 12, 13
See also overhead
perspectives, 166, 174
polling, 170
polymorphism, 58–59
POSIX, 4, 5
POSIX, 97, 98, 99
postconditions, 75, 77, 117
postrestrictions, 75, 76, 77
PowerBuilder, 128, 128, 130
preconditions, 75, 79, 117
prerestrictions, 75, 76
primitive predicates, 77, 80
procedures, 54, 56–57
process analysis, 25
process architecture, 23–26
Process Asset Library (PAL), 27
process code and unit test, 25–26
process design, 20–25, 27–28
process modifications, 20
process reuse, 23
defined, 22–23
process specifications, 102
process tailoring, 26, 27, 28
Process Weaver, 22
producers, 42, 156, 162
profit, 42, 49, 49
profitability index (PI), 44, 142, 143, 143
programming interfaces, 166
programming languages, 31
object-oriented, 196–197
reuse and, 11, 12–13, 52–53
programming operations, 63
programming style, 52
project management, 25
property formalism, 92
protected inheritance, 16
protected members, 16
PROTEUS, 176, 178, 179–184, 180, 185
prototypes, 8
pure virtual functions, 16
qualification, 147
quality, 62
quality indicators, 24
quantities, 42
query methods, 129
RAPID Center Library, 44
rationales, 8
RCS version management, 178, 184
realm declarations, 173
realm interfaces, 73, 169n5, 170–171
realms, 73, 77, 167–168
Real-Time Producer/Consumer (RTP/C)
components, 89, 90, 92
REBOOT facets, 193
recursion, 73, 87
reflectivity, 174
related components, 149, 150
relational database management systems (RDBMS), 98
relative comparison, 48
relative controllability, 69–70
relative observability, 70
remote procedure calls (RPC), 95, 97, 100
repair strategies, 64
repositories, 84–85, 86, 87, 88, 91, 99, 179, 184
and knowledge, 125–130
representation inheritance, 195–203
representation invariants, 196, 200, 201
requirements documentation, 115, 148
requirements specification (RS), 8, 37, 147
RESOLVE, 64–65, 67, 69, 80, 197, 202
modular proof system of, 68
resource category, 130
resources, 48
retrieval by reformulation, 191
return on investment (ROI), 124, 142
in reuse economics, 45–46, 48
reusability procedures, 130–131
reusability support, 59
reusable process descriptions, 25
reusable processes, 24–25
reusable software, 54
reuse
defined, 11
and inheritance, 202
language-independence and, 18
versus reusability, 53
style-based, 84–93
reuse bank, 144
reuse benefits, 14
reuse economics, 20, 41–50, 136–137
reuse histories, 31, 35
reuse investment analysis, 136–144
reuse method, 33
reuse metrics, 136–137
reuse models, 37
reuse scenarios, 121–123
reuse standards, 161–162
reverse engineering, 118–119, 120
reverse tool, 179
rich design elements, 87
risk, 42, 44, 45, 49, 49
and liability, 156
ROI. See return on investment
roles, 25
RPC. See remote procedure calls
RS. See Requirements Specification
rule-based methods, 39, 72
rules, 39, 72, 74, 75, 87, 95, 101, 102, 108,
117–118
run time binding, 106
run time calls, 108
run time compatibility, 56
run time composition, 173
run time reconfiguration, 105
SDIS. See Sustaining Base Information Services scaffolding, 124
scalability, 73
scope, 56–57, 156–163
screen transition diagrams, 37, 38
script rule, 118
segments, 53–54
sentinal nodes, 197, 198n2, 199
sequences, 17–18
service management (SM) subsystem, 108
service objects, 95–96, 98, 102
Set ADT, 65
set implementations, 55
set theory semantics, 2
SILD. See Sustaining Base Information Services (SBIS)
simplicity, 14, 211
Simula, 15
Sleuth, 114, 119, 120, 122–123
Smalltalk, 85, 188–193
software architectures, 85–86
defining process of, 101–102
introduction to, 95–96
for Management Information Systems, 94–103
and parameterized programming, 2–10
and reuse, 104
and style-based reuse, 84–93
software components, 72
behavior of, 62
observability and controllability of, 62–71
reuse of, 64
See also components
software development, 146
design and, 84
for reuse, 104, 147
with reuse, 104, 147
and reuse economics, 43
study of, 125
Software Development Methodology (SWM), 101
Software Engineering Institute, 22
Software Engineering Process Groups (SEPG), 25, 26
Software Productivity Consortium, 22, 139
software reuse, 22, 85
large scale, 104–111
Software Shelf, 84–85, 87, 88–91
software system generators
See generators
SOM, 174
sort algorithms, 17
source code inheritance, 206
specialization, 188–193
specification customization, 39
specification databases, 34, 35
specification design, 70, 195
folklore and, 62
specification language, 68
specification patterns, 37
domain-specific, 32, 34, 35
elements of, 35, 36
specification reuse model, 32–35, 147
use of, 35
SQL*Loader utilities, 99
SQL*Net, 100
standardized interfaces, 167–168, 174
STARS project, 22
state space, 63–64, 66
static analysis, 147
statics and dynamics, 132
StorageTek, 114, 116–118
style-based queries, 91
subclasses, 105, 195–196, 196, 199, 202, 203
subjective interfaces, 174
subjectivity, 166–175, 173–174, 174
subjectivity model, 170–173
superclasses, 105, 189, 195–196, 199, 202, 203
supporting tools, 147
Sustaining Base Information Services (SBIS), 94
installation database (SID) of, 99
switching applications, 104
switching systems, 107–108, 177
switch statements, 169
symmetric components, 73–74, 167
symmetry, 66
Synervision, 22
syntax, 114, 118–119
synthesized attributes, 77
system architectures, 3, 6–7, 148
system binding, 176
system browser, 188
system building, 182–183
system classes, 16
system configuration, 182
system decomposition, 84
system documentation, 213
system engineering, 148
system evolution, 179
system instantiation, 176, 179
System Interface Agreements (SIA), 99
system modelling, 179, 182
systems
versus components, 76
and type equations, 73
system specification phase, 119
system testing, 114–123
system variability, 178, 180–182
table spaces, 99
TAFIM. See Technical Architecture for Information Management
Technical Architecture for Information Management (TAFIM), 97
Technical Reference Model (TRM), 98
Technology Product Managers (TPM), 128
telecommunication domain, 106, 107–108, 114
telecommunication switching system (tss), 104, 107
template parameters, 17
test case reuse, 121–123
test objectives, 114
textbook model, 166
type specifications, 3, 4–5
3Cs model, 23, 52
time horizons, 137
time value of money, 42, 44, 45, 48–49, 49
ToolBook, 151
toolkits, 25, 28, 72, 84, 85, 86, 88–91, 101, 180
in reuse economics, 48
Sleuth, 114, 119, 120, 122–123
tools, 146, 179, 182, 192
queries and, 129
and system modelling, 178
TOOR, 8
trade secrets, 159–160
Train Control System (TCS), 128, 129
training, 24–25
transaction processing systems, 114
transformations, 73
trays, 85
tree traversal, 78
TRIV, 4
TRM. See Technical Reference Model
TST, 177, 178, 184, 185
two-way list, 197–198, 198, 199–201
type checking, 56–57, 58
type equations, 172–173
type parameters, 68–69
understandability, 62–63
Union Pacific Railroad, 124, 128
units and documentation, 212
UNIX, 23, 87, 91, 95, 121, 170, 206
up-layer-call interface, 105
252
usability guidelines, 127
user defined types, 57
user interfaces. See graphical user interface; interfaces
user manuals, 114, 118–119, 205, 206–207
user semantic interpretation, 115

VAG, 79–80
validation, 72, 74–75
value-based management, 137, 144
variable prefixes, 16
Venn diagram, 67, 69
version selection, 179
vertical components, 5–6, 7, 95, 140, 143
views, 4, 166, 192, 205, 209, 213, 214
virtual machines, 5, 73
visibility, 178
vocabulary, 129, 189
  architectural description and, 85, 87
  in models, 52
  See also grammars; lexicon
voice communication systems, 177

warranties, 163
Watchdog project, 128
Weaves, 86
web pages
  and development process, 27
dreck, 80
white box code inheritance, 195–203
workstations, 23, 98, 177
wrappers, 174

Yacc, 85