

# Index

**Abstraction**, 27  
Acceptance testing, 130, 133-34  
Access functions, 84  
Active reviews, 119  
Ada programs, 38, 82  
Adaptive maintenance, 149  
**admin** command, 154-55  
APL language, 18, 101  
**Append Element**, 86, 192  
Application-specific values,  
parameterization of, 67-68  
Architectural model, 237  
**ccount**, 178-82  
definition, 42  
**ar** command, 104  
**argc**, 183-84  
**argv**, 183-84  
Assertion-driven debugging, 141  
Assumptions:  
**ccount** module, 183  
**classify** module, 187  
**counter** module, 185  
and design document, 43, 237  
error module, 191  
**list** module, 191  
**params** module, 183  
**report** module, 186  
AT&T UNIX System V, 2  
Attributes, of software quality, 6-8  
**avien** system, 104  
**awk** program, 13, 20-23, 105

**Banner comments**, 56, 60  
Baseline, definition, 201-33  
BASIC, 101  
Beta test, 133-34  
Bit masks, parameterization of, 67  
Bitwise operations, 74  
Black box test cases, 134-36  
Black box testing, focus of, 142  
Booch diagrams, 38-39  
Bottom-up integration and testing,  
133  
Boundary value analysis, 135  
Braces, placement of, 53-54  
Branch, definition, 153  
Branch coverage, 136  
Branch direction, 136  
Build control, 158-59

**C++**, 101-3, 136  
**calloc**, 99  
CASE data repository problem, 164  
**cat** tool, 18-19  
**cb**, 96-97  
C compilers, 100-1  
**ccount**:  
coding standards, 193-97  
concept exploration, 169-71  
conceptual model, 172  
design, 178-93  
development/operation/maintenance  
environments, 172

error handling, 177-78  
function requirements, 173-77  
manual page, 199-200  
modules, 182-93  
nonfunctional requirements, 177  
project documents, 169-200  
rationale, 193  
regression test script/test cases,  
197-99  
requirements, 171-78  
source code, 201-33  
testing, 142-46  
equivalence partition for, 142-46  
user documentation, 178  
user interface specifications,  
172-73  
**ccount** module, 183  
**cf**, 98  
**cflow** tool, 96, 152  
C function libraries, 73-74  
documentation, 73  
programming practice rules, 242  
Changeability, of requirements  
documents, 16  
Change control, 155-58  
activities/processes, 156-57  
definition, 155-56  
MR classification system, 157-58  
MR review board, 157  
Change control authority, 157  
Change tracking authority, 156

## Index

**char** type, 70, 76, 78, 194  
**chelp** tool, 111-12  
Child MRs, 157  
**cia** tool, 105-7, 152  
**cin**, 101  
C language:  
disguising as another language,  
70-71  
expressions, 74-76  
bitwise operations, 74  
complex boolean expressions,  
74-75  
loop termination expressions, 76  
parenthesization, 74  
relational expressions, 75  
side-effects, 74  
and UNIX, 3  
Classes and object-oriented methods,  
38  
**Classify Line**, 182  
**classify** module, 187-90, 193  
**Clean Command Line**, 181  
C metric program prototypes, 20-24  
CMET tool, 107-8  
metric values reported by, 107  
COCOMO, 107  
Code [See Source Code Control  
System (SCCS)]  
Code inspections, 121-24  
**ccount**, 196  
checklist, 244-48  
error checklist for C, 123-24  
follow-up, 122-23  
guidelines, 123  
inspection readiness, 121-22  
overview, 122  
preparation for, 122  
rework, 122  
team inspection, 122  
(See also Source Code Control  
System)  
Code optimization techniques, 78-80  
Coding standards, **ccount**, 193-97  
code checks/inspections, 196  
comments, 195  
control structures, 194  
error checking, 195-96  
expressions, 194  
formatting, 195  
function header comment  
template, 197  
module header comment template,  
196  
modules/access to program  
objects, 196  
naming conventions, 193-94  
preprocessor, use of, 195  
target metric values, 196  
types, 194

Cohesion, 28-29  
and coupling, 30  
Command line format, **ccount**, 173  
Commentary source lines (CSLs),  
20, 62, 170, 174, 199  
Comment errors, 123, 246-47  
Comments, 47, 55-62  
**ccount**, 195  
C programming practice rules,  
241  
definition, 47  
program goals, 55  
program plans, 55  
purpose of program, 56  
role of program object, 55  
size/complexity/density, 241  
strategy, 61  
Comment-to-code ratio  
(CSL/NCSL), 62, 170  
Comparison errors, 123, 245  
Compile modules, 82, 85-86  
categories of, 85  
Compilers, 100-1  
Completeness, of requirements  
documents, 16  
Complex boolean expressions, 74-75  
Comprehension process, debugging,  
140  
Computation errors, 123, 245  
Computer Aided Software  
Engineering (CASE), 40,  
162-66  
Concept exploration document:  
preparation of, 13  
template, 234-35  
Concept exploration phase, 10-13  
documentation, preparation of, 13  
feasibility analysis, 12-13  
process, 11-13  
tools/techniques, 13  
Configuration management, 152-59  
build control, 158-59  
change control, 155-58  
and project monitoring, 159  
version control, 153-55  
Consistency, of requirements  
documents, 16  
Constraints/limitations:  
**ccount** module, 183  
**classify** module, 187  
**counter** module, 185  
and design document, 238  
error module, 191  
**list** module, 192  
**params** module, 183  
**report** module, 186  
Constructor functions, 102  
Content coupling, 30  
Control coupling, 29-30

Control flow editor (CFE), StP, 41  
Control-flow errors, 123, 245-46  
Control structures:  
**ccount**, 194  
C programming practice rules,  
240  
guidelines for choosing, 52  
using appropriate, 51-52  
Correction process, debugging,  
140-41  
Corrective maintenance, 149  
Correctness, of software, 7  
Costs, maintenance, 149-51  
**counter** module, 185-86, 193  
**count\_list**, 86-89  
Count list data type, 86-89  
Coupling, 29-30  
and cohesion, 30  
content, 30  
control, 29-30  
data definition, 29  
data element, 29  
global, 30  
C preprocessor, 70-73, 242  
C programming practice rules,  
240-43  
about revealing program structure,  
240-41  
coding efficiency, 242-43  
comments, 241  
comment size/complexity/density,  
241  
control structures, 240  
error checking, 243  
expressions, 242  
function libraries, 242  
modules/access to program  
objects, 243  
names/declarations, 240  
parameterization, 241-42  
preprocessor, 242  
types, 242  
**cpr\_** tool, 98  
**Create List**, 86, 192  
**creat** function, 112  
**cscope** tool, 94-95, 152  
**cslawk** program, 22  
constraints/limitations:  
**ccount** module, 183  
**classify** module, 187  
**counter** module, 185  
and design document, 238  
error module, 191  
**list** module, 192  
**params** module, 183  
**report** module, 186  
Constructor functions, 102  
Content coupling, 30  
Control coupling, 29-30  
Control flow editor (CFE), StP, 41  
Control-flow errors, 123, 245-46  
Control structures:  
**ccount**, 194  
C programming practice rules,  
240  
guidelines for choosing, 52  
using appropriate, 51-52  
Correction process, debugging,  
140-41  
Corrective maintenance, 149  
Correctness, of software, 7  
Costs, maintenance, 149-51  
**counter** module, 185-86, 193  
**count\_list**, 86-89  
Count list data type, 86-89  
Coupling, 29-30  
and cohesion, 30  
content, 30  
control, 29-30  
data definition, 29  
data element, 29  
global, 30  
C preprocessor, 70-73, 242  
C programming practice rules,  
240-43  
about revealing program structure,  
240-41  
coding efficiency, 242-43  
comments, 241  
comment size/complexity/density,  
241  
control structures, 240  
error checking, 243  
expressions, 242  
function libraries, 242  
modules/access to program  
objects, 243  
names/declarations, 240  
parameterization, 241-42  
preprocessor, 242  
types, 242  
**cpr\_** tool, 98  
**Create List**, 86, 192  
**creat** function, 112  
**cscope** tool, 94-95, 152  
**cslawk** program, 22  
constraints/limitations:  
**ccount** module, 183  
**classify** module, 187  
**counter** module, 185  
and design document, 238  
error module, 191  
**list** module, 192  
**params** module, 183  
**report** module, 186  
Constructor functions, 102  
Content coupling, 30  
Control coupling, 29-30

Data-driven tests (See Black box test cases)

Data element coupling, 29

Dataflow design, 35-37

Dataflow diagrams, 35

Dataflow editor (DFE), StP, 40-41

Dataflows, 35

Data-reference errors, 123, 244

Data repositories, future of, 163-65

Data-structure design, 32-35

Data-structure editor (DSE), StP, 40

Data-structure limits, parameterization of, 67

Data-type cohesion, 28

Data-type limits, parameterization of, 67-68

dbx debugger, 110, 152, 163

dbxtool, 110, 152, 163

Debuggers, 109-10

Debugging, 139-42
 

- debugging advisers, 111-12
- definition, 139
- goal of, 139-40
- strategy, 140-41
- tactics, 141-42

Decision coverage, 136

Declarations, 82-83, 85
 

- C programming practice rules, 240

#define'd constants, comments about roles/purposes, 58

Defining declaration, definition, 83

Delete\_Element, 86, 192

delta command, 154-55

DeMorgan's Laws, simplifying boolean expressions based on, 74-75

Derived items, 159

Design, 26-45
 

- concepts/principles:
  - abstraction, 27
  - cohesion, 28-29
  - coupling, 29-30
  - information hiding, 30-31
  - modularity, 27-28
- design document, contents of, 42-43
- graphical design tools, 39-41
- methods, 31-39
  - dataflow design, 35-37
  - data structure design, 32-35
  - object-oriented design, 37-39
  - structured design, 32
- program design languages, 41-42

Design document:
 

- contents of, 42-43
- template, 237-38

Designer, source code, 121

Design methods, 26

definition, 31

Design notations, 26

Design tools, 26

Destroy\_Node, 90

Destructor functions, 102

Development environment, ccount, 172

diff program, 99-100, 138-39, 152

Documentation, 8
 

- ccount, 178
- C function libraries, 73
- external, 63-65
- internal, 47-63
- quality of, and maintenance costs, 150
  - and software project size, 5

Dogs, habits, 2

Domain Software Engineering Environment (DSEE), 164

-d option, ccount, 173

Dot-c files, 82

Dot-h files, 84

double values, 77-78, 194, 242

Dynamic analysis tools, 108-14
 

- debuggers, 109-10
- debugging advisers, 111-12
- performance monitoring tools, 112-14
- tracing tools, 110-11

Editors/browsers, 94-96

ed text editor, 94

Efficiency, of software, 7

Efficient code, 78-80
 

- C programming practice rules, 242-43

egrep program, 20-22

emacs text editor, 94, 101

Engineering document templates, 234-43
 

- concept exploration document, 234-35
- design document, 237-38
- project summary document, 239
- requirements document, 235-37

Entity-relationship editor (ERE), StP, 41

Enumeration types, 51

Environment-specific values, parameterization of, 67

eqn, 166

Equivalence partitioning, 135

Error checking:
 

- ccount, 195-96
- C programming practice rules, 243

Error handling:
 

- ccount, 177-78
- ccount module, 183

classify module, 188

counter module, 185

and design document, 43, 238

error module, 191

list module, 192

params module, 184

report module, 186

error module, 191

Execution paths, tracing, 52-54

Execution tracing, 141

Expected changes, and design document, 43, 238

Expressions, 74-76 (See also Documentation; Internal documentation)
 

- bitwise operations, 74
- ccount, 194
- complex boolean expressions, 74-75
- loop termination expressions, 76
- parenthesization, 74
- programming practice rules, 242
- relational expressions, 75
- side-effects, 74

extern, 90, 243

External documentation, 63-65
 

- implementation notes, 63-64
- program unit notebooks, 63

External names, 83-84

Failures, software projects, 3

FDELIM variable, 177, 184, 199

Feasibility of requirements documents, 16

Feasibility analysis:
 

- ccount metrics tool, 171
- concept exploration phase, 12-13

Find\_Function\_Name, ccount, 182

Flags, 30, 32

Floating point types, 77

float values, 77-78, 194, 242

Follow-up, to code inspections, 122-23

Formatting, ccount, 195

Formatting C code, 96-100

fspar function, 99

FrameMaker (Frame Technology Corp.), 166

Functional decomposition (See Structured design)

Function arguments, type bug classes, 77-78

Function delimiter string:
 

- ccount, 174
- determination, 176-77
- /\*FN, 197
- recognition, 177
- restrictions, 176

Function header comments, 56-57, 60-61

Function header recognition, ccount, 174

Function return value, type/purpose, 61

Functions:
 

- comments about roles/purposes, 58
- compared to macros, 71-72
- values/parameters, 84

Future trends:
 

- data repositories, 163-65
- document production support, 166
- front-end support, 166
- project management support, 165
- UNIX software development environments (SDEs), 161-62
- user interfaces, 162-63

General Object-Oriented Development (GOOD), 38

Generate-Counts, ccount, 182

get command, 154-55

Get\_Program\_Parameters, 181

Get\_Source\_Line, 182

Glass box tests (See White box test cases)

Global coupling, 30

Glossary, design document, 43, 238

Goal statements, 56-57
 

- definition, 55

grap, 39

Graphical design tools, 39-41

Graphical output, StP, 41

grep, 18-21, 99

Header files, 84-85

Hierarchical Object-Oriented Design (HOOD), 38

Hierarchy charts, 32

High-level programming, 66, 81-92 (See also Low-level programming)
 

- implementation of modules in C, 81-86
- modularization example, 86-90
- module size/complexity standards, 90-91

howrule variable, 111

Hypertext systems, and information storage, 164-65

Implementation notes, 63-64

Implementation structure, 81-86
 

- module contents/organization, 85-86
- program object access control, 82-84

Implementer, code inspections, 121

Inappropriate control structures, 51-52

Incidental cohesion, 29

#include directives, 21, 70, 84-85

Indent statements/substatements, 53

indent tool, 97

Information hiding, 30-31 (See also Secrets)
 

- advantages of, 31
- and object-oriented design, 38

Input file size, ccount, 177

Input file specification, ccount, 173

Input-output errors, 123, 246

Input-output functions, C function libraries, 73

Input-output tests (See Black box test cases)
 

- Inspections (See Code inspections)

Integration testing, 131-33

Interface checking, 141

Interface errors, 246

Internal codes, parameterization of, 67
 

- Internal documentation, 47-63 (See also Comments; Documentation; Non-role-based operations; Self-documenting code)
  - comments, 55-62
  - guidelines, 62-63
  - metrics, 62
  - self-documenting code, 47-55
  - side-effect information, 60

Internal documentation metrics, definition, 62

Internal module, 237

Internal structure, 238
 

- ccount module, 183
- classify module, 188-90
- counter module, 185-86
- error module, 191
- list module, 192
- params module, 184-85
- report module, 186-87

Interpreters, 101

\_int suffix, 49

int variables, 67, 242

Is\_Empty\_List, 86, 90, 192

is\_ prefix, 194

ISTAR, 165

Jackson System Development (JSD), 33

JAD methodology, 18, 143

Kernighan and Ritchie style, of brace placement, 54

Korn shell programming language, 20n

Language usage errors, 124

lcc, 112

Leaf module, 237

lex, 100-1

Library functions (See C function libraries)

Life-cycle models, 5-6
 

- and prototypes, 17-18

Line of code, 4n

lint tool, 78, 94, 100, 104-5, 111, 141, 196
 

- problems reported by, 105

Lisp language, 101

list module, 191-92

Localization process, debugging, 140

Local scope, 82

Logical cohesion, 28

Logical path coverage, 136

Logical phases, of life-cycle models, 5

long type, 77, 85-86

Loop termination expressions, 76

Low-level programming, 66-80 (See also High-level programming)
 

- C function libraries, 73-74
- C preprocessor, 70-73
- definition, 66
- expressions, 74-76
- optimization, 78-80
- parameterization, 66-70
- types in C, 76-78

lprof tool, 108, 112-14, 137

McCabe's cyclomatic complexity metric, 90

Macros:
 

- comments about roles/purposes, 58
- compared to functions, 71-72
- with large definitions, 72
- "Magic numbers," and parameterization, 67

Maintainability, 7, 148

Maintenance, 148-52
 

- costs, 149-51
- cost factors, 149-51
- lowering, 151
- UNIX tools for, 151-52

Maintenance environment, ccount, 172

Maintenance errors, 124, 248

makefile, 103, 152

make utility, 103-4, 151-52, 159, 162

malloc procedure, 144

Manual page, ccount:
 

- bugs, 200
- description, 199-200

- Manual page, **ccount** (*continued*):
  - name, 199
  - synopsis, 199
- Mathematical functions, C function libraries, 73
- Memory allocation/manipulation functions, C function libraries, 73
- message operation, 35
- Metrics, 107-8
  - internal documentation, 62
- Metric values, **ccount**, 196
- Minimal useful system prototypes, 17
- Mini-specs (*See* Pseudocode)
- Missing function delimiter, **ccount**, 174
- Moderator, code inspections, 121
- Modification request form (MRF), 156
- Modification requests [*See* MRs (modification requests)]
- Modified Kernighan and Ritchie style, of brace placement, 54
- Modified Pascal style, of brace placement, 54
- Modula, 82
- Modularity, 27-28
- Modularity errors, 124, 247
- Modularization criteria, 38
- Module behavior, and design document, 43, 237
- Module cohesion, 28-29
- Module descriptions, in design document, 42
- Module header comments, 56-57, 60-61
- Module independence, and maintenance costs, 150
- Modules:
  - ccount**, 182-93
    - descriptions, 183-92
    - overview, 182
    - uses relationships, 192-93
  - contents/organization of, 85-86
  - C programming practice rules, 243
  - definition, 237
  - internal structure, 43
  - packaging, 43, 238
    - ccount** module, 183
    - classify** module, 187-88
    - counter** module, 185
    - error** module, 191
    - list** module, 192
    - params** module, 183-84
    - report** module, 186
  - size/complexity standards, 90-91
  - monitor function, 112
- Motif graphical user interface
  - standard, 163
- MR classification system, 157-58
- MR review board, 157
- MRs (modification requests), 156-57
  - child MRs, 157
  - classification system, 157-58
  - closing, 158
  - MR review board, 157-58
  - opening, 156
- Multiword identifiers, as naming convention, 49, 193
- Naming, 47-50, 193-94**
  - conventions, 49
  - C programming practice rules, 240
    - importance to self-documenting code, 49
  - informative names, 49-50
- Noncommentary source lines (NCSLs), 20, 62, 90, 169-72, 174, 199
  - estimating in C source files, 20-21
- Nonderived items, 158
- Nonlocal scope, 82
- Non-role-based operations,
  - explanation of, 59-60
- Nontabbed format header/body layout, **ccount**, 176
- Nontabbed format output example, **ccount**, 176
- Notes, and function header comments, 61
- Null loop bodies, 54
- nvcc tool, 108, 137, 142
  - limitations of, 146
  - used on **ccount**, 144-45
- Object-oriented databases and information storage, 164**
- Object-oriented design, 37-39
- Object-Oriented Structured Design (OOSD), 38
- Offsets, parameterization of, 67
- Omitted **break** statements, as control strategy, 54
- OPEN LOOK graphical user interface standard, 163
- Operating environment, **ccount**, 172
- Operating system service functions, C function libraries, 73
- Operations, parameterization of, 67-68
- Operator overloading, 102
- Optimization, 78-80
- Option actions/format, **ccount**, 173
- Output report contents/formats, **ccount**, 174-75

- Packaging, modules, 238**
- Packaging information, 43
- Parameterization, 66-70, 84
  - C programming practice rules, 241-42
    - of operations, 67-68
    - ways to parameterize, 68-70
    - what to parameterize, 66-68
- Parameters:
  - comments about roles/purposes of, 59
  - definition, 66
- params** module, 183-84
- Parenthesization, 74
- Pascal style, of brace placement, 54
- PDL (program design language), 41-42
- Perfective maintenance, 149
- Performance errors, 124, 247-48
- Performance monitoring tools, 112-14
- Performance requirements, **ccount**, 177
- pic**, 39, 166
- Picture editor, StP, 41
- Pipes, 18-19
- Political feasibility (*See also* Feasibility analysis):
  - of **ccount** metrics tool, 171
  - of a system, 12-13
- Portability, of software, 7
- Portable Common Tool Environment (PCTE), 165
- Predecessors, definition, 153
- Prefixes, as naming convention, 49, 193
- Printing C code, 96-100
- Process audits, 124-25
  - audit team recommendations, implementation of, 125
  - development process, characterization of, 125
  - findings,
    - formulation/communication of, 125
    - plan of, 124
- Process quality, 118
- Producer-consumer model, and information hiding, 31
- Product quality, 118
- Profilers, 79
- prof** tool, 108, 112
- Program design languages, 41-42
- Program generation tools, 94-104
  - C++, 101-3
  - compilers/interpreters, 100-1
  - editors/browsers, 94-96
  - make** tool, 103-4
  - searching/printing code, tools for, 96-100

- Program goals, 55
  - statement of, 56-57
- Program lifetime and maintenance costs, 149
- Programming language/style and maintenance costs, 150
- Program object access control, 82-84
  - C programming practice rules, 243
    - external names, 83-84
    - function values/parameters, 84
    - header files, 84
    - scope rules, 82-83
- Program object roles, 47-48, 55, 59
  - statement of, 58-59
- Program plans:
  - definition, 55, 57
  - and function header comments, 61
  - statement of, 57-58
- Program readability, 46-65 (*See also* Readability):
  - external documentation, 63-65
  - internal documentation, 47-63
- Program unit, definition, 63
- Program unit notebooks, 63
- Project methodology, 5
- Project monitoring, and configuration management, 159
- Project summary document,
  - template, 239
- Prolog, 18, 101
- Prototyping, 17-18
  - ccount**, 170
  - C metric program prototypes, 20-24
    - definition, 17
    - prototyping languages, 18
    - types of, 17
    - UNIX shell language, 18-19
- Pseudocode, 37
- \_ptr** suffix, 49, 194
- Quality, attributes of, 6-8**
- Quality assurance, 117-25
  - code inspections, 121-24
  - process audits, 124-25
  - reviews, 118-21
- Rational and design document, 43, 238**
- Readability:
  - of requirements documents, 16
  - of work products, 7
- read** function, 112
- Regression testing, 137-39
  - ccount**, 142-44
- Relational expressions, 75
- Reliability of software, 8
- Report\_Errors**, 181
- Report\_Metrics**, 181
- report** module, 186-87
- Report section header/body contents, **ccount**, 175
- Requirements document, template, 235-37
- Requirements specification phase, 13-17
  - requirements specification document, 14-15
  - information contained in, 14-15
  - quality criteria for, 16
  - tools/techniques, 16-17
- Return value, 84, 184
  - purpose of, 61
- Reusability, of software, 8
- Reviews, 118-21
  - active review process, 119
  - completion criteria, 121
  - meetings, 121
  - performance of, 120-21
  - reporting results of, 121
  - review readiness, 119-20
  - start-up meeting, 120
  - traditional process, problems with, 118-19
- Revision Control System (RCS), 153-54
- Revisions, definition, 153
- Rework, 122
- Ritchie, Dennis, 2
- Robustness of software, 8
- Role-based operations and comments, 59
- Role of program object, 47-48, 55, 59
  - statement of, 58-59
- Sandwich integration and testing, 133**
  - and documentation, 5
- Scope rules, C language, 82-83
- sdb** debugger, 109, 152
- SDEs [*See* UNIX software development environments (SDEs)]
- Secrets, 30-31 (*See also* Information hiding)
  - ccount** module, 183
  - classify** module, 187
  - counter** module, 185
  - definition, 30
  - and design document, 43, 238
  - error** module, 191
  - list** module, 192
  - params** module, 183
  - report** module, 186
- sed**, 21, 94
- Self-documenting code, 47-55 (*See also* Source code)
  - appropriate code structures, use of, 51-52
  - definition, 47
  - names, as expression of pertinent information, 47-50
  - program structure, display of, 52-55
    - types, careful use of, 50-51
- Sequential cohesion, 28-29
- short** type, 67, 77-78
- showmatch** feature, vi editor, 94
- SID, 154-55
- Side-effects, 60, 74, 84
- Signed types, 76, 194
- Sinks, 35
- Skipping code, 142
- Software, definition, 3
- Software configuration item, 153
- Software configuration management (*See* Configuration management)
- Software design (*See* Design)
- Software development environments (SDEs) [*See* UNIX software development environments (SDEs)]
- Software engineering, 1
  - primary purpose of, 169
  - state of the art in, 2-3
- Software life cycle, 5-6
- Software maintenance (*See* Maintenance)
- Software metrics, 107-8
- Software product, definition, 3-4
- Software product requirement, 13-14
- Software projects:
  - definition, 4
  - failures, 3
  - project methodology, 5
  - size/type, 4-5
- Software quality assurance (*See* Quality assurance)
- Software testing (*See* Testing)
- Software Through Pictures (StP), 40-41
- sort** tool, 18, 103
- Source code (*See also* Code inspections; Self-documenting code):
  - ccount**, 201-33
  - code inspection tester, 121
  - designer, 121
  - dividing into several files, 85
  - searching/printing tools, 96-100
  - skipping, 142
- Source Code Control System (SCCS), 13, 153-54
- services, 154
- version number, 154-55

Sources, 35  
 Stabilization process, debugging, 140  
 Staff stability and maintenance costs, 149  
**state0**, 99  
 Statemate (I-Logix, Inc.), 166  
 Statement coverage, 136  
 Stages in, 130-34  
 Static analysis tools, 105-8  
   software metrics, 107-8  
**static functions**, 85, 90, 243  
**stderr**, 35, 142, 172  
**stdin**, 35, 162, 173, 175-76  
**stdout**, 35, 142, 172  
 Storage usage errors, 247  
 String manipulation functions, C  
   function libraries, 73  
 Strongly typed language, 50-51  
 Stroustrup, Bjarne, 101  
   **\_str** suffix, 49, 73  
 Structural tests (*See* White box test cases)  
 Structure chart editor (SCE), StP, 140  
 Structure charts, 32  
 Structured design, 32  
 Successors, definition, 153  
 Suffixes, as naming convention, 49, 193  
**switch** statement, 140  
   controlling, 54  
   using **if-else** statements instead of, 124  
**syned**, 96  
 Syntax-directed editors, 95-96  
 System and acceptance testing, 130, 133-34  
  
**Tabbed format header/body layout, ccount, 175**  
 Tabbed format output example, **ccount**, 175-76  
**tcov**, 114, 137  
 Team inspection, 122  
 Technical feasibility (*See also* Feasibility analysis):  
   of **ccount** metrics tool, 171  
   of a system, 12  
 Technical Publishing Software system (Interleaf), 166  
 Temporal phases, of life-cycle models, 5  
 Testability:  
   of requirements documents, 16  
   of software, 8

Testing, 125-46  
   debugging, 139-42  
   definitions, 125-26  
   and maintenance costs, 150  
   in parts, 85  
   regression testing, 137-39  
   resource allocation, 127-30  
   stages in, 130-34  
     integration testing, 131-33  
     system and acceptance testing, 133-34  
   unit testing, 130-31  
   test cases, 134-37  
     black box, 134-36  
     white box, 136-37  
   testing of **ccount**, 142-46  
   test team formation, 126-27  
 Thompson, Ken, 2  
 Throwaway prototypes, 17  
**time** command, 24, 112, 177  
**Tokenize\_Line, ccount**, 182  
 Top-down integration and testing, 131-33  
**-t** option, **ccount**, 173, 175, 199  
 Totally Automated Regression Test System (TARTS), 139  
 Traceability of requirements documents, 16  
 Traceability errors, 124, 248  
 Tracing tools, 110-11  
 Transition diagram editor (TDE), StP, 41  
**troff**, 39, 166  
 Tuning values, parameterization of, 67  
**typedef** utility, 49-51, 70, 174  
 Typed language, 50  
 Type-rich language, 50-51  
 Types, 76-78  
   **ccount**, 194  
   **char** type, 76  
   C programming practice rules, 242  
   definition, 50  
   enumeration types, 51  
   floating point types, 77  
   function arguments, type bug classes, 77-78  
   **long** type, 77  
   principles guiding selection/use of, 51  
   **short** type, 77  
   signed types, 76  
   **\_type** suffix, 194

**Underscores, as naming convention, 49, 193**  
 Unit testing, 130-31  
 UNIX shell language, 13, 18, 163  
   control structures, 19  
   as prototyping tools, 18-19  
   punctuation symbols, 161-62  
 UNIX software development environments (SDEs), 161-62  
   data repositories, 163-65  
   document production support, 166  
   front-end support, 166  
   and project management support, 165  
   user interfaces, 162-63  
 UNIX system, C compilers, 100  
 UNIX tools:  
   for coding phase of life cycle, 93-116  
   dynamic analysis tools, 108-14  
   program generation tools, 94-104  
   static analysis tools, 105-8  
**unsigned** types, 67, 194  
**Update\_Counters, ccount**, 182  
 Up-down design (*See* Structured design)  
 User documentation, **ccount**, 178  
 User interfaces, future of, 162-63  
**userprmt** function, 95  
 Uses relationships and design document, 43, 238  
  
**Validation and maintenance costs, 150**  
 Variables, comments about roles/purposes, 58-59  
 Variations, definition, 153  
 Version control, 153-55  
**vi** text editor, 94  
**vmake** tool, 144  
**void** function, 86, 242  
  
**Waterfall model of software life cycle, 6-7**  
 White box test cases, 134, 136-37  
   testing tools, 136-37  
 Work product, definition, 4  
 WYSIWYG (what you see is what you get), and UNIX SDEs, 161, 166  
  
**yacc**, 101