What Is This Book About?	p. 1
From Handcrafting to Automated Assembly Lines	p. 1
Generative Programming	p. 5
Benefits and Applicability	p. 13
Analysis and Design Methods and Techniques	p. 17
Domain Engineering	p. 19
Why Is This Chapter Worth Reading?	p. 19
What Is Domain Engineering?	p. 20
Domain Analysis	p. 23
Domain Design and Domain Implementation	p. 24
Application Engineering	p. 30
Product-Line Practices	p. 30
Key Domain Engineering Concepts	p. 32
Domain	p. 32
Domain Scope and Scoping	p. 34
Relationships between Domains	p. 37
Features and Feature Models	p. 38
Method Tailoring and Specialization	p. 43
Survey of Domain Analysis and Domain Engineering Methods	p. 44
Feature-Oriented Domain Analysis (FODA)	p. 44
FODA Process	p. 45
Organization Domain Modeling (ODM)	p. 46
The ODM Process	p. 48
Draco	p. 48
Capture	p. 51
Domain Analysis and Reuse Environment (DARE)	p. 51
Domain-Specific Software Architecture (DSSA) Approach	p. 52
Algebraic Approach	p. 53
Other Approaches	p. 54
Domain Engineering and Related Approaches	p. 55
Historical Notes	p. 56
Summary	p. 57
Domain Engineering and Object-Oriented Analysis and Design	p. 60
Why Is This Chapter Worth Reading?	p. 60
OO Technology and Reuse	p. 60
Solution Space	p. 61
Problem Space	p. 61
Relationship between Domain Engineering and Object-Oriented Analysis and Design (OOA/D) Methods	p. 64
Aspects of Integrating Domain Engineering and OOA/D Methods	p. 64
Horizontal versus Vertical Methods	p. 67
Selected Methods	p. 69

Rational Unified Process 5.0 p. 69
OOram p. 71
Reuse-driven Software Engineering Business (RSEB) p. 73
FeatuRSEB p. 79
Domain Engineering Method for Reusable Algorithmic Libraries (DEMRAL) p. 80
Feature Modeling p. 82
Why Is This Chapter Worth Reading? p. 82
Features Revisited p. 82
Feature Modeling p. 83
Feature Models p. 86
Feature Diagrams p. 87
Mandatory Features p. 88
Optional Features p. 89
Alternative Features p. 90
Or-Features p. 91
Normalized Feature Diagrams p. 93
Expressing Commonality in Feature Diagrams p. 95
Expressing Variability in Feature Diagrams p. 95
Other Information Associated with Feature Diagrams in a Feature Model p. 96
Assigning Priorities to Variable Features p. 99
Availability Sites, Binding Sites, and Binding Modes p. 100
Sites p. 100
Availability Sites p. 100
Binding Sites and Binding Modes p. 101
Relationship between Optimizations and Availability Sites, Binding Sites, and Binding Modes p. 101
Relationship between Feature Diagrams and Other Modeling Notations and Implementation Techniques p. 102
Single Inheritance p. 105
Multiple Inheritance p. 107
Parameterized Inheritance p. 108
Static Parameterization p. 111
Dynamic Parameterization p. 112
Implementing Constraints p. 113
Tool Support for Feature Models p. 114
Frequently Asked Questions about Feature Diagrams p. 116
Feature Modeling Process p. 119
How to Find Features p. 119
Role of Variability in Modeling p. 123
Separation of Concerns p. 123
Decomposition Techniques p. 125
Variability in Modeling p. 128

The Process of Generative Programming p. 131
Why Is This Chapter Worth Reading? p. 131
Generative Domain Models p. 131
Main Development Steps in Generative Programming p. 134
Adapting Domain Engineering for Generative Programming p. 135
Domain-Specific Languages p. 137
DEMRAL: Example of a Domain Engineering Method for Generative Programming p. 142
Outline of DEMRAL p. 144
Domain Analysis p. 146
Domain Definition p. 146
Domain Modeling p. 148
Identification of Key Concepts p. 149
Feature Modeling p. 151
Feature Starter Set for ADTs p. 151
Feature Starter Set for Algorithms p. 154
Domain Design p. 155
Scope Domain Model for Implementation p. 155
Identify Packages p. 156
Develop Target Architectures and Identify the Implementation Components p. 156
Identify User DSLs p. 156
Identify Interactions between DSLs p. 157
Specify DSLs and Their Translation p. 157
Configuration DSLs p. 158
Expression DSLs p. 162
Domain Implementation p. 164
Implementation Technologies p. 165
Generic Programming p. 167
Why Is This Chapter Worth Reading? p. 167
What Is Generic Programming? p. 167
Generic versus Generative Programming p. 170
Generic Parameters p. 171
Parametric versus Subtype Polymorphism p. 173
Genericity in Java p. 176
Bounded versus Unbounded Polymorphism p. 184
A Fresh Look at Polymorphism p. 186
Parameterized Components p. 190
Parameterized Programming p. 192
Types, Interfaces, and Specifications p. 193
Adapters p. 195
Vertical and Horizontal Parameters p. 197
Module Expressions p. 200

C++ Standard Template Library p. 201
Iterators p. 202
Freestanding Functions versus Member Functions p. 203
Generic Methodology p. 205
Historical Notes p. 209
Component-Oriented Template-Based C++ Programming Techniques p. 211
Why Is This Chapter Worth Reading? p. 211
Types of System Configuration p. 212
C++ Support for Dynamic Configuration p. 213
C++ Support for Static Configuration p. 213
Static Typing p. 214
Static Binding p. 214
Inlining p. 214
Templates p. 218
Parameterized Inheritance p. 221
Typedefs p. 221
Member Types p. 221
Nested Classes p. 222
Prohibiting Certain Template Instantiations p. 222
Static versus Dynamic Parameterization p. 224
Wrappers Based on Parameterized Inheritance p. 231
Template Method Based on Parameterized Inheritance p. 231
Parameterizing Binding Mode p. 234
Consistent Parameterization of Multiple Components p. 236
Static Interactions between Components p. 236
Components with Influence p. 238
Components under Influence p. 240
Structured Configurations p. 243
Recursive Components p. 245
Intelligent Configuration p. 248
Aspect-Oriented Programming p. 251
Why Is This Chapter Worth Reading? p. 251
What Is Aspect-Oriented Programming? p. 252
Aspect-Oriented Decomposition Approaches p. 254
Subject-Oriented Programming p. 254
Composition Filters p. 256
Demeter / Adaptive Programming p. 259
Aspect-Oriented Programming p. 262
How Aspects Arise p. 264
Composition Mechanisms p. 267
Requirements on Composition Mechanisms p. 267

Minimal Coupling p. 267
Different Binding Times and Modes p. 270
Noninvasive Adaptability p. 272
Example: Synchronizing a Bounded Buffer p. 275
"Tangled" Synchronized Stack p. 278
Separating Synchronization Using Design Patterns p. 281
Separating Synchronization Using SOP p. 287
Some Problems with Design Patterns and Some Solutions p. 292
Object Schizophrenia p. 294
Preplarming Problem p. 295
Traceability Problem p. 295
Implementing Noninvasive, Dynamic Composition in Smalltalk p. 296
Model of the Composition p. 296
Composition API p. 297
Instance-Specific Extension Protocol p. 298
Defining Methods in Instances p. 299
Adding Instance Variables to Instances p. 301
Kinds of Crosscutting p. 303
How to Express Aspects in Programming Languages p. 305
Separating Synchronization Using AspectJ Cool p. 306
Implementing Dynamic Cool in Smalltalk p. 310
Example: Synchronizing an Assembly System p. 310
Architecture of the Smalltalk Implementation of Dynamic Cool p. 315
Implementation Technologies for Aspect-Oriented Programming p. 317
Technologies for Implementing Aspect-Specific Abstractions p. 317
Technologies for Implementing Weaving p. 321
AOP and Specialized Language Extensions p. 328
AOP and Active Libraries p. 328
Final Remarks p. 331
Generators p. 332
Why Is This Chapter Worth Reading? p. 332
What Are Generators? p. 333
Transformational Model of Software Development p. 335
Technologies for Building Generators p. 339
Compositional versus Transformational Generators p. 341
Kinds of Transformations p. 344
Compiler Transformations p. 344
Refinements p. 344
Optimizations p. 345
Source-to-Source Transformations p. 348
Transformation Systems p. 350

Scheduling Transformations p. 352
Existing Transformation Systems and Their Applications p. 355
Selected Approaches to Generation p. 357
Draco p. 357
GenVoca p. 363
Example: Applying GenVoca to the Domain of Data Containers p. 364
GenVoca Model p. 366
Defining Realms and Layers in P++ p. 372
Implementing GenVoca Layers in C++ p. 375
Composition Validation p. 385
Transformations and GenVoca p. 387
Frameworks and GenVoca p. 388
Approaches Based on Algebraic Specifications p. 389
Static Metaprogramming in C++ p. 397
Why Is This Chapter Worth Reading? p. 397
What Is Metaprogramming? p. 398
A Quick Tour of Metaprogramming p. 399
Static Metaprogramming p. 405
C++ As a Two-Level Language p. 406
Functional Flavor of the Static Level p. 410
Class Templates As Functions p. 410
Integers and Types As Data p. 411
Symbolic Names Instead of Variables p. 411
Constant Initialization and typedef-Statements Instead of Assignment p. 411
Template Recursion Instead of Loops p. 412
Conditional Operator and Template Specialization As Conditional Constructs p. 413
Template Metaprogramming p. 413
Template Metafunctions p. 414
Metafunctions As Arguments and Return Values of Other Metafunctions p. 416
Representing Metainformation p. 418
Member Traits p. 419
Traits Classes p. 420
Traits Templates p. 421
Example: Using Template Metafunctions and Traits Templates to Implement Type Promotions p. 425
Compile-Time Lists and Trees As Nested Templates p. 427
Compile-Time Control Structures p. 433
Explicit Selection Constructs (? : , IF[], and SWITCH[]) p. 433
Implementing IF[] without Partial Template Specialization p. 434
Switch Construct (SWITCH[]) p. 436
Taking Advantage of Lazy Behavior p. 438
SWITCH[] without Partial Template Specialization p. 444

Template Recursion As a Looping Construct p. 444
Explicit Looping Constructs (WHILE[], DO[], and FOR[]) p. 451
While-Loop (WHILE[]) p. 451
Do-Loop (DO[]) p. 455
For-Loop (FOR[]) p. 459
Code Generation p. 462
Simple Code Selection p. 462
Composing Templates p. 464
Generators Based on Expression Templates p. 464
Recursive Code Expansion p. 465
Explicit Loops for Generating Code (EWHILE[], EDO[], and EFOR[]) p. 473
EWHILE[] p. 473
EDO[] p. 476
EFOR[] p. 477
Nesting Static E-Loops p. 479
Example: Using Static Execute Loops to Test Metafunctions p. 487
Partial Evaluation in C++ p. 493
Workarounds for Partial Template Specialization p. 497
Problems of Template Metaprogramming p. 499
Historical Notes p. 499
Intentional Programming p. 503
Why Is This Chapter Worth Reading? p. 503
What Is Intentional Programming? p. 504
Technology behind IP p. 510
System Architecture p. 510
Representing Programs in IP: The Source Graph p. 512
Treelike Structures p. 513
Graphlike Structures p. 513
Source Graphs: The Big Picture p. 516
The Essence of Source Graphs: Abstraction Sharing and Parameterization p. 518
Source Graph + Methods = Active Source p. 518
Kinds of Methods p. 519
Working with the IP Programming Environment p. 522
Editing p. 524
Further Capabilities of the IP Editor p. 528
Extending the IP System with New Intentions p. 537
Advanced Topics p. 541
Questions, Methods, and a Frameworklike Organization p. 542
Source-Pattem-Based Polymorphism p. 543
Methods As Visitors p. 545
Asking Questions Synchronously and Asynchronously p. 546

Reduction p. 547
The Philosophy behind IP p. 551
Why Do We Need Extendible Programming Environments? or What Is the Problem with Fixed Programming Languages? p. 551
Problems with General-Purpose Languages and Conventional Libraries p. 551
Problems with Comprehensive Application-Specific Languages p. 553
The IP Solution: An Extendible Programming Environment p. 555
Moving Focus from Fixed Languages to Language Features and the Emergence of an Intention Market p. 556
Intentional Programming and Component-Based Development p. 557
Frequently Asked Questions p. 559
Summary p. 566
Application Examples p. 569
List Container p. 571
Why Is This Chapter Worth Reading? p. 571
Overview p. 571
Domain Analysis p. 572
Domain Design p. 573
Implementation Components p. 577
Manual Assembly p. 583
Specifying Lists p. 586
The Generator p. 586
Extensions p. 590
Bank Account p. 593
Why Is This Chapter Worth Reading? p. 593
The Successful Programming Shop p. 593
Design Pattems, Frameworks, and Components p. 596
Domain Engineering and Generative Programming p. 597
Feature Modeling p. 599
Architecture Design p. 600
Implementation Components p. 603
Configurable Class Hierarchies p. 610
Designing a Domain-Specific Language p. 612
Bank Account Generator p. 619
Testing Generators and Their Products p. 623
Generative Matrix Computation Library (GMCL) p. 624
Why Is This Chapter Worth Reading? p. 624
Why Matrix Computations? p. 625
Domain Analysis p. 626
Domain Definition p. 626
Domain Modeling p. 628
Domain Design and Implementation p. 634

Matrix Type Generation p. 638
Target Architecture and Matrix Implementation Components p. 639
Matrix Configuration DSL p. 654
Matrix Configuration Generator p. 661
Configuration DSL Parser p. 664
Assigning Defaults to DSL Features p. 666
Matrix Component Assembler p. 672
Generating Code for Matrix Expressions p. 678
Evaluating Matrix Expressions p. 678
Modeling the Elementwise Expression Evaluation Using Objects p. 680
Overview of the Implementation p. 685
Matrix Operator Templates p. 686
Matrix Expression Templates p. 690
Matrix Cache p. 696
Generating getElement p. 698
Metafunctions for Computing Result Types of Expressions p. 701
Generating Matrix Assignment p. 706
Lessons Learned p. 708
Problems with Template Metaprogrammning p. 710
Implementing the Matrix Component in IP p. 713
Appendices p. 719
Conceptual Modeling p. 721
What Are Concepts? p. 721
Theories of Concepts p. 722
Basic Terminology p. 723
The Classical View p. 724
The Probabilistic View p. 727
The Exemplar View p. 728
Summary of the Three Views p. 728
Important Issues Concerning Concepts p. 730
Stability of Concepts p. 730
Concept Core p. 731
Informational Contents of Features p. 732
Feature Composition and Relationships between Features p. 732
Quality of Features p. 733
Abstraction and Generalization p. 733
Conceptual Modeling, Object-Orientation, and Software Reuse p. 736
Instance-Specific Extension Protocol for Smalltalk p. 738
Protocol for Attaching Listener Objects in Smalltalk p. 741
Glossary of Matrix Computation Terms p. 746
Metafunction for Evaluating Dependency Tables p. 749

Glossary of Generative Programming Terms p. 754
References p. 757
Index p. 799
Table of Contents provided by Blackwell's Book Services and R.R. Bowker. Used with permission.