

Contents

<i>List of Figures</i>	page xi
<i>List of Tables</i>	xv
<i>Preface</i>	xvii
<i>Acknowledgments</i>	xxi
<i>Disclaimer</i>	xxii
PART I THE TAO OF SCIENTIFIC OOP	1
1 Development Costs and Complexity	3
1.1 Introduction	3
1.2 Conventional Scientific Programming Costs	7
1.3 Conventional Programming Complexity	11
1.4 Alternative Programming Paradigms	19
1.5 How Performance Informs Design	24
1.6 How Design Informs Performance	25
1.7 Elements of Style	27
2 The Object-Oriented Way	31
2.1 Nomenclature	31
2.2 Object-Oriented Analysis and Design	33
2.3 Encapsulation and Information Hiding	36
2.4 Wrapping Legacy Software	40
2.5 Composition, Aggregation, and Inheritance	48
2.6 Static and Dynamic Polymorphism	53
2.7 OOP Complexity	54
2.8 More on Style	55
3 Scientific OOP	57
3.1 Abstract Data Type Calculus	57
3.2 Analysis-Driven Design	66
3.2.1 Design Metrics	69
3.2.2 Complexity Theory	71
3.2.3 Information Theory	73

3.3 Still More on Style	77
3.4 The Tao of SOOP	79
PART II SOOP TO NUTS AND BOLTS	83
4 Design Patterns Basics	85
4.1 Essentials	85
4.2 Foundations	86
4.2.1 Building Architecture	87
4.2.2 Software Architecture	93
4.2.3 Scientific Software Architecture	98
4.3 Canonical Contexts	99
4.3.1 The Lorenz Equations: A Chaotic Dynamical System	100
4.3.2 Quantum Vortex Dynamics in a Superfluid	102
4.3.3 Burgers' Equation: Shock Formation and Dissipation	104
5 The Object Pattern	107
5.1 The Problem	107
5.2 The Solution	108
5.2.1 Fortran Implementation	110
5.2.2 C++ Style and Tools	116
5.2.3 C++ Implementation of Vortex	122
5.3 The Consequences	127
5.4 Related Patterns	127
6 The Abstract Calculus Pattern	129
6.1 The Problem	129
6.2 The Solution	130
6.2.1 Fortran Implementation	131
6.2.2 C++ Implementation	137
6.3 The Consequences	140
6.4 Related Patterns	141
7 The Strategy and Surrogate Patterns	143
7.1 The Problem	143
7.2 The Solution	144
7.2.1 Fortran Implementation	146
7.2.2 C++ Implementation	155
7.3 The Consequences	164
7.4 Related Patterns	164
8 The Puppeteer Pattern	167
8.1 The Problem	167
8.2 The Solution	169
8.2.1 Fortran Implementation	170
8.2.2 A C++ Tool: 2D Allocatable Arrays	185
8.2.3 C++ Implementation	191

8.3	The Consequences	199
8.4	Related Patterns	201
9	Factory Patterns	202
9.1	The Problem	202
9.2	The Solution	203
9.2.1	Fortran Implementation	205
9.2.2	C++ Implementation	216
9.3	The Consequences	226
9.4	Related Patterns	227
PART III GUMBO SOOP		229
10	Formal Constraints	231
10.1	Why Be Formal?	231
10.2	Problem Statement	233
10.3	Side Effects in Abstract Calculus	237
10.4	Formal Specification	238
10.4.1	Modeling Arrays with OCL	238
10.4.2	Hermeticity	239
10.4.3	Economy	242
10.5	The Shell Pattern	244
10.6	An Assertion Utility for Fortran	245
10.7	Case Study: A Fluid Turbulence Solver	247
11	Mixed-Language Programming	251
11.1	Automated Interoperability Tools	251
11.2	Manual Interoperability: C++/Fortran 95	254
11.3	Case Study: ForTrilinos and CTrilinos	259
11.3.1	C Interoperability in Fortran 2003	261
11.3.2	Method Invocation	269
11.3.3	Tabulation, Construction, and Destruction	273
11.3.4	Polymorphism and Hierarchy Mirroring	278
11.3.5	Discussion	281
12	Multiphysics Architectures	285
12.1	Toward a Scalable Abstract Calculus	285
12.1.1	Amdahl's Law and Parallel Efficiency	286
12.1.2	Automatic Parallelization: Compiler Technology	292
12.1.3	Directive-Based Parallelization: OpenMP	295
12.1.4	Library-Based Parallelization: ForTrilinos and MPI	297
12.1.5	Intrinsic Parallelization: Fortran 2008 Coarrays	309
12.2	Case Studies: Multiphysics Modeling	317
12.2.1	Quantum Turbulence in Superfluid Liquid Helium	318
12.2.2	Lattice-Boltzman Biofluid Dynamics	320

12.2.3 Particle Dispersion in Magnetohydrodynamics	325
12.2.4 Radar Scattering in the Atmospheric Boundary Layer	326
12.3 The Morfeus Framework	330
Appendix A Mathematical Background	335
A.1 Interpolation	335
A.1.1 Lagrange Interpolation	335
A.2 Linear Solvers	337
A.2.1 Gaussian Elimination	337
A.2.2 LU Decomposition	342
A.3 Nonlinear Solvers	343
A.3.1 Newton's Method in 1D	343
A.3.2 Newton's Method in Multiple Dimensions	344
A.4 Partial Differential Equations	345
A.4.1 The Heat Equation	345
A.4.2 The Burgers Equation	346
A.5 Numerical Analysis	347
A.5.1 Finite Differences	347
A.5.2 Numerical Methods for Differential Equations	350
Appendix B Unified Modeling Language Elements	357
B.1 Use Case Diagrams	357
B.2 Class Diagrams	359
B.2.1 Classes	359
B.2.2 Relationships	362
B.3 Object Diagrams	366
B.4 Sequence Diagrams	367
B.4.1 Interactions and Messages	368
B.4.2 Sequence Diagrams	368
B.5 The Object Constraint Language	370
B.5.1 The Context of OCL Expression	370
B.5.2 Initial Value Rule	371
B.5.3 Specifying Object Invariants	371
B.5.4 Adding Pre- and Postconditions to Operations	371
<i>Bibliography</i>	373
<i>Index</i>	379