
Contents

Foreword	v
Preface	vii
Contents	xi
1 Introduction	1
1.1 Partitioning, knowledge, and abstractions	2
1.2 Three examples of software architecture	3
1.3 Reflections	5
1.4 Perspective shift	6
1.5 Architects architecting architectures	7
1.6 Risk-driven software architecture	8
1.7 Architecture for agile developers	9
1.8 About this book	10
 I Risk-Driven Software Architecture	 13
2 Software Architecture	15
2.1 What is software architecture?	16
2.2 Why is software architecture important?	18
2.3 When is architecture important?	22
2.4 Presumptive architectures	23
2.5 How should software architecture be used?	24
2.6 Architecture-indifferent design	25
2.7 Architecture-focused design	26

CONTENTS

2.8	Architecture hoisting	27
2.9	Architecture in large organizations	30
2.10	Conclusion	31
2.11	Further reading	32
Risk-Driven Model		35
3.1	What is the risk-driven model?	37
3.2	Are you risk-driven now?	38
3.3	Risks	39
3.4	Techniques	42
3.5	Guidance on choosing techniques	44
3.6	When to stop	47
3.7	Planned and evolutionary design	48
3.8	Software development process	51
3.9	Understanding process variations	53
3.10	The risk-driven model and software processes	55
3.11	Application to an agile processes	56
3.12	Risk and architecture refactoring	58
3.13	Alternatives to the risk-driven model	58
3.14	Conclusion	60
3.15	Further reading	61
Example: Home Media Player		65
4.1	Team communication	67
4.2	Integration of COTS components	75
4.3	Metadata consistency	81
4.4	Conclusion	86
Modeling Advice		89
5.1	Focus on risks	89
5.2	Understand your architecture	90
5.3	Distribute architecture skills	91
5.4	Make rational architecture choices	92
5.5	Avoid Big Design Up Front	93
5.6	Avoid top-down design	95
5.7	Remaining challenges	95
5.8	Features and risk: a story	97

CONTENTS

II Architecture Modeling		101
6 Engineers Use Models		103
6.1	Scale and complexity require abstraction	104
6.2	Abstractions provide insight and leverage	105
6.3	Reasoning about system qualities	105
6.4	Models elide details	106
6.5	Models can amplify reasoning	107
6.6	Question first and model second	108
6.7	Conclusion	108
6.8	Further reading	109
7 Conceptual Model of Software Architecture		111
7.1	Canonical model structure	114
7.2	Domain, design, and code models	115
7.3	Designation and refinement relationships	116
7.4	Views of a master model	118
7.5	Other ways to organize models	121
7.6	Business modeling	121
7.7	Use of UML	122
7.8	Conclusion	123
7.9	Further reading	123
8 The Domain Model		127
8.1	How the domain relates to architecture	128
8.2	Information model	131
8.3	Navigation and invariants	133
8.4	Snapshots	134
8.5	Functionality scenarios	135
8.6	Conclusion	136
8.7	Further reading	137
9 The Design Model		139
9.1	Design model	140
9.2	Boundary model	141
9.3	Internals model	141
9.4	Quality attributes	142
9.5	Walkthrough of Yinzer design	143
9.6	Viewtypes	157
9.7	Dynamic architecture models	161
9.8	Architecture description languages	162

CONTENTS

9 Conclusion	163
10 Further reading	164
the Code Model	167
0.1 Model-code gap	167
0.2 Managing consistency	171
0.3 Architecturally-evident coding style	174
0.4 Expressing design intent in code	175
0.5 Model-in-code principle	177
0.6 What to express	178
0.7 Patterns for expressing design intent in code	180
0.8 Walkthrough of an email processing system	187
0.9 Conclusion	193
Encapsulation and Partitioning	195
1 Story at many levels	195
2 Hierarchy and partitioning	197
3 Decomposition strategies	199
4 Effective encapsulation	203
5 Building an encapsulated interface	206
6 Conclusion	210
7 Further reading	210
Model Elements	213
1 Allocation elements	214
2 Components	215
3 Component assemblies	219
4 Connectors	223
5 Design decisions	233
6 Functionality scenarios	234
7 Invariants (constraints)	239
8 Modules	239
9 Ports	241
0 Quality attributes	246
1 Quality attribute scenarios	249
2 Responsibilities	251
3 Tradeoffs	252
4 Conclusion	253
Model Relationships	255
Projection (view) relationship	256

CONTENTS

xv

13.2 Partition relationship	261
13.3 Composition relationship	261
13.4 Classification relationship	261
13.5 Generalization relationship	262
13.6 Designation relationship	263
13.7 Refinement relationship	264
13.8 Binding relationship	268
13.9 Dependency relationship	269
13.10 Using the relationships	269
13.11 Conclusion	270
13.12 Further reading	271
14 Architectural Styles	273
14.1 Advantages	274
14.2 Platonic vs. embodied styles	275
14.3 Constraints and architecture-focused design	276
14.4 Patterns vs. styles	277
14.5 A catalog of styles	277
14.6 Layered style	277
14.7 Big ball of mud style	280
14.8 Pipe-and-filter style	281
14.9 Batch-sequential style	283
14.10 Model-centered style	285
14.11 Publish-subscribe style	286
14.12 Client-server style & N-tier	288
14.13 Peer-to-peer style	290
14.14 Map-reduce style	291
14.15 Mirrored, rack, and farm styles	293
14.16 Conclusion	294
14.17 Further reading	295
15 Using Architecture Models	297
15.1 Desirable model traits	297
15.2 Working with views	303
15.3 Improving view quality	306
15.4 Improving diagram quality	310
15.5 Testing and proving	312
15.6 Analyzing architecture models	312
15.7 Architectural mismatch	318
15.8 Choose your abstraction level	319
15.9 Planning for the user interface	320

15.10 Prescriptive vs. descriptive models	320
15.11 Modeling existing systems	320
15.12 Conclusion	322
15.13 Further reading	323
16 Conclusion	325
16.1 Challenges	326
16.2 Focus on quality attributes	330
16.3 Solve problems, not just model them	331
16.4 Use constraints as guide rails	332
16.5 Use standard architectural abstractions	333
Glossary	335
Bibliography	347
Index	355