

Contents

Preface xiii

Acknowledgments xxi

About the Author xxiii

Chapter 1: Introduction to Context-Driven Design 1

 Designing Requirements 2

 What Is Design? 9

 Ad Hoc Design 12

 Planned Design 14

 Engineered Design 14

 Summary of Design Approaches 18

 Making IT Application Development More of
 an Engineering Discipline 19

 Taking IT Architecture into Account 20

 Concluding Remarks 21

Chapter 2: A Hierarchy of Designs 23

 Justifying the Hierarchy of Designs 23

 Context Design 28

 Tasks 28

 User Groups 30

 Data Tables 30

 Messages between Tasks 31

 Task Dependencies 31

 Putting It All Together 32

 Analysis of the Context Design 34

 Integration Design 35

 Technical Design 41

 User Interface Design 44

Database Design	46
Implementation	47
Is It Really Engineering?	48
Concluding Remarks	51
Chapter 3: Reusing Existing Methods and Practices	53
Agile	54
Individuals and Interactions over Processes and Tools	55
Working Software over Comprehensive Documentation	56
Customer Collaboration over Contract Negotiations	58
Responding to Change over Following a Plan	59
Conclusion	60
Upside-Down Design	60
Use Cases	62
Atomicity	63
Confusion of Design Layers	64
Use Cases Are Confusing	66
Large Use Case Documents Are Hard to Understand	67
Use Cases Do Not Assist Engineered Design	67
Conclusion	68
The Problem with Estimating Cost	68
Why Is BDUF Big?	72
Iterations	74
Quality	75
Testing and Inspection	76
Using Existing Practices in Context-Driven Design	78
Learning Organizations	80
Concluding Remarks	80
Chapter 4: The Problem of Large Applications	83
The Dimensions of Size	84
Problems with Large Projects	88
Requirements Problems	88
Lack of End User Support	91
Technical Design Problems	93
Procurement and Outsourcing	96
Can Large Projects Be Avoided?	100
Concluding Remarks	103

Chapter 5: The Relationship with the Business	105
Understanding Business Processes	106
When It's Not a Process	112
Business Services	112
Resource Management	113
Reviewing and Monitoring	115
The Need for a Wider View	115
Applying the Business Strategy to Application Development	118
Speed of Development	119
Cost versus Performance and Availability	119
Experimental Business Programs	120
How Long Before the Benefits	120
The Need for Security	120
Designing for the Existing Culture	120
Design for a Culture to Which the Organization Aspires	121
Allow for Changing Plans	122
Support a Learning Organization	122
Non-Business Applications	122
Analysis	123
Is the Process Well Formed?	123
Dependency Analysis	123
Objectives Analysis	127
Concluding Remarks	128
Chapter 6: The Relationship with the Users	129
Adding the Detail	129
Task Details	131
Task Fragments	135
Common Purpose Groups	135
Data Tables	136
Messages	137
Nonfunctional Requirements	138
Who Uses the Context Design?	140
Who Are the Users?	141
Business Process Operations	142
Monitoring by Management	143
Data Used by Other Applications	147
Data Analysis	148

Application Administration	149
Analyzing the Context Design	151
Process Layer	151
Task Details	153
Data Table Details	154
User Group Details	155
Message Details	155
Reviewing the Context Design	156
Concluding Remarks	158
Chapter 7: The Relationship to Other IT Projects	159
Integration Design	161
Applications	161
Services	162
Databases	165
Services Interface Design	170
Service Interface Definition	172
Designing Reusable Services	176
Existing Applications	178
Knowing What Is There	178
Replacing Existing Applications	180
Fashioning Services from Existing Applications	184
Looking Back at the Design Process	186
Concluding Remarks	188
Chapter 8: User Interface Design and Ease of Use	189
Logical User Interfaces	191
From Tasks to Clicks	194
Ease of Use	199
Function	200
Information	201
Navigation	202
Text	202
Help	203
Intuitive and Likable Applications	203
Ease-of-Use Design	205
Monitoring Ease of Use	208
Transaction and Task Integrity	208

The User Interface Design and the Other Detailed Designs	212
Concluding Remarks	212
Chapter 9: Database Design	215
Database Design	215
Database Design Theory	223
Programmers versus the Database Designer	233
Database Access Services	236
NoSQL	238
Concluding Remarks	242
Chapter 10: Technical Design—Principles	243
Principles of High Performance on a Single Machine	244
Cache	245
Multithreading and Multiprocessing	248
Principles of High Performance on Many Servers	252
Front-End Parallelism	252
Back-End Parallelism	256
Principles of High Resiliency	260
The Need for Testing and Benchmarking	263
The Technical Design Process	265
Concluding Remarks	268
Chapter 11: Technical Design—Structure	271
Program Structure	272
What Is a Framework?	276
The Variety of Programming Languages	281
Choosing a Programming Language and Framework	286
Choose a Language that Fits Your Organization's Skill Set ..	287
Choose a Language that Is Appropriate for	
Your Application's Performance Goals	287
Choose a Language that Can Meet Your	
Integration Requirements	287
Choose a Language that Supports Group	
Working If Needed	287
Choose Version Control Software and Project Management	
Software as Well as a Language	288
Choose a Language that Chimes with Your Development	
Methodology	289

Extending the Framework	290
Implementing Common Functionality	293
Concluding Remarks	295
Chapter 12: Security Design	297
IT Application Security Principles	299
Authentication	300
Access Control	302
User Administration	303
Security Protection	304
Security Monitoring	306
The Security Elements of Each Design	307
Context Design	307
Integration Design	311
User Interface Design	312
Database Design	312
Technical Design	314
Security Programming	316
Concluding Remarks	319
Chapter 13: The Future of Application Development	323
How Context-Driven Design Changes Application Development	323
Context-Driven Design Opportunities	325
New Tools	326
Context and Integration Design	328
User Interface and Database Design	328
Technical Design	329
The Application Development Challenges	332
Flexibility	332
Operations	334
Correctness	335
Quality	336
Professionalism	337
Concluding Remarks	339
Appendix A: Context Design Checklist	341
Description	341
Elaboration	344
Analysis	344
References	349
Index	353

Preface

This book is the fruit of about 15 years of thinking about IT application development. It started when I was working on IT architecture in the late 1990s. At that time I wrote a book called *IT Architecture and Middleware: Strategies for Building Large, Scalable Systems* (a second edition coauthored with Peter Bye in 2004 is still available). This was about the technology for building integrated applications and about how to make the applications scalable, highly available, and secure. Other people were thinking along similar lines at the time, and the kinds of solutions Peter and I were proposing came to be called Service Oriented Architectures (SOAs) because the basic idea was to have reusable services with which you can rapidly assemble new applications using integration technology. In spite of the advantages of SOA, which we thought were obvious, very little happened. IT managers liked the SOA story but didn't get around to implementing anything. Something was missing, and almost from the beginning I had the suspicion that the missing something was application development. In other words, we didn't have a good answer to the question, "How do you develop an SOA application?" Or perhaps the question is better expressed as, "I have a bunch of requirements; what do I have to do to ensure that I end up with an SOA solution rather than a stand-alone application?" Over the next few years I did less and less thinking about architecture and more and more thinking about application development.

I first did application programming in the late 1970s. Since that time I have mostly worked in the system and environmental software arena, fixing and designing. I spent a lot of time mending data management software and occasionally would be thrown a compiler bug or an operating system bug to fix. I have done a fair bit of designing and programming system software. Later on I worked on database design and repository design (I could give a long discourse on version control—strangely, very few people wanted to hear it). By the year 2000, I was experienced in many aspects of computer technology, but I hadn't done a lot of straightforward application design and programming, so I couldn't in all honesty go to the application developers and tell them they were doing it all wrong.

At that time, application development gurus showed little interest in architecture. Instead they were at war among themselves. In one corner was the "big design up front" (BDUF) crowd who promoted designs based on Unified Modeling Language (UML) modeling. The designs were structured, well documented, and full of quality