

Essential Software Architecture

Ian Gorton

Essential Software Architecture

Second Edition

 Springer

Ian Gorton
Laboratory Fellow
Pacific Northwest National Laboratory
PO Box 999
MSIN: K7-90
Richland, WA 99352
USA
ian.gorton@pnl.gov

ACM Computing Classification (1998): D.2

ISBN 978-3-642-19175-6 e-ISBN 978-3-642-19176-3
DOI 10.1007/978-3-642-19176-3
Springer Heidelberg Dordrecht London New York

Library of Congress Control Number: 2011926871

© Springer-Verlag Berlin Heidelberg 2006, 2011

This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilm or in any other way, and storage in data banks. Duplication of this publication or parts thereof is permitted only under the provisions of the German Copyright Law of September 9, 1965, in its current version, and permission for use must always be obtained from Springer. Violations are liable to prosecution under the German Copyright Law.

The use of general descriptive names, registered names, trademarks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

Cover design: KuenkelLopka GmbH

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Preface

Welcome to the second edition of Essential Software Architecture. It is 5 years since the first edition was published, and in the software architecture world, 5 years is a long time. Hence this updated version, with refreshed chapters to capture new developments in methods and technologies, and to relate relevant experiences from practise. There's new material covering enterprise architecture, agile development, enterprise service bus technologies and RESTful Web services. All chapters have an updated and more extensive list of recommended reading, capturing many of the best new books, papers, web sites and blogs that I know of.

Most notably, the completely new Chap. 10 provides a case study on the design of the MeDICi technology, which extends an open source enterprise service bus with a component-based programming model. The MeDICi technology is open source and freely downloadable (<http://www.medic.pnl.gov>), making it a highly suitable tool for teaching the advanced concepts of middleware and architecture described in this text.

At its heart however, this remains a book that aims to succinctly impart a broad sweep of software architecture knowledge relating to systems built from mainstream middleware technologies. This includes a large, diverse spectrum of systems, ranging from Web-based ecommerce sites to scientific data management and high performance financial data analysis systems.

Motivation

What hasn't changed in the last 5 years is that many projects I work with or review lack an explicit notion of an architectural design. Functional requirements are usually captured using traditional or agile techniques, agreed with stakeholders, and addressed through highly iterative or traditional waterfall methods. But the architectural issues, the "how" the application achieves its purpose, the "what" happens when things change and evolve or fail, are frequently implicit (this means they are in somebody's head, maybe) at best. At worst, they are simply not addressed in any way that can be described in terms other than accidental. Frequently, when I ask for an overview of the application architecture and the driving nonfunctional

requirements at the first technical meeting, people start drawing on a whiteboard. Or they show me code and dive into the internals of the implementation based around their favorite, trendy technology. Either of these is rarely a good sign.

The problems and risks of poor architectural practices are well known and documented within the software engineering profession. A large body of excellent architectural knowledge is captured in broadly accessible books, journals and reports from members of the Software Engineering Institute (SEI), Siemens and various other renowned industrial and academic institutions.

While the focus of much of this literature is highly technical systems such as avionics, flight simulation, and telecommunications switching, this book leans more to the mainstream world of software applications. In a sense, it bridges the gap between the needs of the vast majority of software professionals and the current body of knowledge in software architecture. Specifically:

- It provides clear and concise discussions about the issues, techniques and methods that are at the heart of sound architectural practices.
- It describes and analyzes the general purpose component and middleware technologies that support many of the fundamental architectural patterns used in applications.
- It looks forward to how changes in technologies and practices may affect the next generation of business information systems.
- It uses familiar information systems as examples, taken from the author's experiences in banking, e-commerce and government information systems.
- It provides many pointers and references to existing work on software architecture.

If you work as an architect or senior designer, or you want to 1 day, this book should be of value to you. And if you're a student who is studying software engineering and need an overview of the field of software architecture, this book should be an approachable and useful first source of information. It certainly won't tell you everything you need to know – that will take a lot more than can be included in a book of such modest length. But it aims to convey the essence of architectural thinking, practices and supporting technologies, and to position the reader to delve more deeply into areas that are pertinent to their professional life and interests.

Outline

The book is structured into three basic sections. The first is introductory in nature, and approachable by a relatively nontechnical reader wanting an overview of software architecture.

The second section is the most technical in nature. It describes the essential skills and technical knowledge that an IT architect needs.

The third is forward looking. Six chapters each introduce an emerging area of software practice or technology. These are suitable for existing architects and

designers, as well as people who've read the first two sections, and who wish to gain insights into the future influences on their profession.

More specifically:

- *Chapters 1–3*: These chapters provide the introductory material for the rest of the book, and the area of software architecture itself. Chapter 1 discusses the key elements of software architecture, and describes the roles of a software architect. Chapter 2 introduces the requirements for a case study problem, a design for which is presented in Chap. 9. This demonstrates the type of problem and associated description that a software architect typically works on. Chapter 3 analyzes the elements of some key quality attributes like scalability, performance and availability. Architects spend a lot of time addressing the quality attribute requirements for applications. It's therefore essential that these quality attributes are well understood, as they are fundamental elements of the knowledge of an architect.
- *Chapters 4–10*: These chapters are the technical backbone of the book. Chapter 4 introduces a range of fundamental middleware technologies that architects commonly leverage in application solutions. Chapter 5 is devoted to describing Web services, including both SOAP and REST-based approaches. Chapter 6 builds on the previous chapters to explain advanced middleware platforms such as enterprise service bus technologies. Chapter 7 presents a three stage iterative software architecture process that can be tailored to be as agile as a project requires. It describes the essential tasks and documents that involve an architect. Chapter 8 discusses architecture documentation, and focuses on the new notations available in the UML version 2.0. Chapter 9 brings together the information in the first 6 chapters, showing how middleware technologies can be used to address the quality attribute requirements for the case study. It also demonstrates the use of the documentation template described in Chap. 8 for describing an application architecture. Chapter 10 provides another practical case study describing the design of the open source MeDICi Integration Framework, which is a specialized API for building applications structured as pipelines of components.
- *Chapters 11–15*: These chapters each focus on an emerging technique or technology that will likely influence the futures of software architects. These include software product lines, model-driven architecture, aspect-oriented architecture and the Semantic Web. Each chapter introduces the essential elements of the method or technology, describes the state-of-the-art and speculates about how increasing adoption is likely to affect the required skills and practices of a software architect. Each chapter also relates its approach to an extension of the ICDE case study in Chap. 9.

Acknowledgments

First, thanks to the chapter contributors who have helped provide the content on software product lines (Mark Staples), aspect-oriented programming (Jenny Liu), model-driven development (Liming Zhu), Web services (Paul Greenfield) and the Semantic Web (Judi Thomson). Adam Wynne also coauthored the chapter on MeDICi. Your collective efforts and patience are greatly appreciated.

Contact details for the contributing authors are as follows:

Dr Mark Staples, National ICT Australia, email: mark.staples@nicta.com.au

Dr Liming Zhu, National ICT Australia, email: liming.zhu@nicta.com.au

Dr Yan Liu, Pacific Northwest National Lab, USA, email: jenny.liu@nicta.com.au

Adam Wynne, Pacific Northwest National Lab, USA, email: adam.wynne@pnl.gov

Paul Greenfield, School of IT, CSIRO, Australia, email: paul.greenfield@csiro.au

Dr Judi McCuaig, University of Guelph, Canada, email: judi@cis.uoguelph.ca

I'd also like to thank everyone at Springer who has helped make this book a reality, especially the editor, Ralf Gerstner.

I'd also like to acknowledge the many talented software architects, engineers and researchers who I've worked closely with recently and/or who have helped shape my thinking and experience through long and entertaining geeky discussions. In no particular order these are Anna Liu, Paul Greenfield, Shiping Chen, Paul Brebner, Jenny Liu, John Colton, Karen Schhardt, Gary Black, Dave Thurman, Jerome Haack, Sven Overhage, John Grundy, Muhammad Ali Babar, Justin Almqvist, Rik Littlefield, Kevin Dorow, Steffen Becker, Ranata Johnson, Len Bass, Lei Hu, Jim Thomas, Deb Gracio, Nihar Trivedi, Paula Cowley, Jim Webber, Adrienne Andrew, Dan Adams, Dean Kuo, John Hoskins, Shuping Ran, Doug Palmer, Nick Cramer, Liming Zhu, Ralf Reussner, Mark Hoza, Shijian Lu, Andrew Cowell, Tariq Al Naeem, Wendy Cowley and Alan Fekete.

Contents

- 1 Understanding Software Architecture** 1
 - 1.1 What is Software Architecture? 1
 - 1.2 Definitions of Software Architecture 2
 - 1.2.1 Architecture Defines Structure 3
 - 1.2.2 Architecture Specifies Component Communication 4
 - 1.3 Architecture Addresses Nonfunctional Requirements 5
 - 1.3.1 Architecture Is an Abstraction 6
 - 1.3.2 Architecture Views 7
 - 1.4 What Does a Software Architect Do? 8
 - 1.5 Architectures and Technologies 9
 - 1.6 Architect Title Soup 11
 - 1.7 Summary 12
 - 1.8 Further Reading 13
 - 1.8.1 General Architecture 13
 - 1.8.2 Architecture Requirements 13
 - 1.8.3 Architecture Patterns 14
 - 1.8.4 Technology Comparisons 14
 - 1.8.5 Enterprise Architecture 15
- 2 Introducing the Case Study** 17
 - 2.1 Overview 17
 - 2.2 The ICDE System 17
 - 2.3 Project Context 19
 - 2.4 Business Goals 21
 - 2.5 Constraints 22
 - 2.6 Summary 22
- 3 Software Quality Attributes** 23
 - 3.1 Quality Attributes 23
 - 3.2 Performance 24
 - 3.2.1 Throughput 24
 - 3.2.2 Response Time 25

3.2.3	Deadlines	25
3.2.4	Performance for the ICDE System	26
3.3	Scalability	27
3.3.1	Request Load	27
3.3.2	Simultaneous Connections	29
3.3.3	Data Size	29
3.3.4	Deployment	30
3.3.5	Some Thoughts on Scalability	30
3.3.6	Scalability for the ICDE Application	30
3.4	Modifiability	30
3.4.1	Modifiability for the ICDE Application	33
3.5	Security	33
3.5.1	Security for the ICDE Application	34
3.6	Availability	34
3.6.1	Availability for the ICDE Application	35
3.7	Integration	35
3.7.1	Integration for the ICDE Application	36
3.8	Other Quality Attributes	36
3.9	Design Trade-Offs	37
3.10	Summary	37
3.11	Further Reading	38

4 An Introduction to Middleware Architectures and Technologies

		39
4.1	Introduction	39
4.2	Middleware Technology Classification	40
4.3	Distributed Objects	41
4.4	Message-Oriented Middleware	43
4.4.1	MOM Basics	44
4.4.2	Exploiting MOM Advanced Features	45
4.4.3	Publish-Subscribe	50
4.5	Application Servers	54
4.5.1	Enterprise JavaBeans	55
4.5.2	EJB Component Model	56
4.5.3	Stateless Session Bean Programming Example	57
4.5.4	Message-Driven Bean Programming Example	58
4.5.5	Responsibilities of the EJB Container	59
4.5.6	Some Thoughts	60
4.6	Summary	61
4.7	Further Reading	62
4.7.1	CORBA	62
4.7.2	Message-Oriented Middleware	62
4.7.3	Application Servers	63

5	Service-Oriented Architectures and Technologies	65
5.1	Background	65
5.2	Service-Oriented Systems	66
5.2.1	Boundaries Are Explicit	68
5.2.2	Services Are Autonomous	69
5.2.3	Share Schemas and Contracts, Not Implementations	69
5.2.4	Service Compatibility Is Based on Policy	70
5.3	Web Services	71
5.4	SOAP and Messaging	73
5.5	UDDI, WSDL, and Metadata	74
5.6	Security, Transactions, and Reliability	77
5.7	RESTful Web Services	78
5.8	Conclusion and Further Reading	79
6	Advanced Middleware Technologies	81
6.1	Introduction	81
6.2	Message Brokers	81
6.3	Business Process Orchestration	87
6.4	Integration Architecture Issues	91
6.5	What Is an Enterprise Service Bus	95
6.6	Further Reading	95
7	A Software Architecture Process	97
7.1	Process Outline	97
7.1.1	Determine Architectural Requirements	98
7.1.2	Identifying Architecture Requirements	98
7.1.3	Prioritizing Architecture Requirements	99
7.2	Architecture Design	101
7.2.1	Choosing the Architecture Framework	102
7.2.2	Allocate Components	108
7.3	Validation	110
7.3.1	Using Scenarios	111
7.3.2	Prototyping	113
7.4	Summary and Further Reading	114
8	Documenting a Software Architecture	117
8.1	Introduction	117
8.2	What to Document	118
8.3	UML 2.0	119
8.4	Architecture Views	120
8.5	More on Component Diagrams	123
8.6	Architecture Documentation Template	126
8.7	Summary and Further Reading	127

9 Case Study Design	129
9.1 Overview	129
9.2 ICDE Technical Issues	129
9.2.1 Large Data	129
9.2.2 Notification	131
9.2.3 Data Abstraction	131
9.2.4 Platform and Distribution Issues	131
9.2.5 API Issues	132
9.2.6 Discussion	133
9.3 ICDE Architecture Requirements	133
9.3.1 Overview of Key Objectives	133
9.3.2 Architecture Use Cases	134
9.3.3 Stakeholder Architecture Requirements	134
9.3.4 Constraints	136
9.3.5 Nonfunctional Requirements	136
9.3.6 Risks	137
9.4 ICDE Solution	137
9.4.1 Architecture Patterns	137
9.4.2 Architecture Overview	138
9.4.3 Structural Views	139
9.4.4 Behavioral Views	142
9.4.5 Implementation Issues	145
9.5 Architecture Analysis	145
9.5.1 Scenario Analysis	145
9.5.2 Risks	146
9.6 Summary	146
10 Middleware Case Study: MeDICI	147
10.1 MeDICI Background	147
10.2 MeDICI Hello World	148
10.3 Implementing Modules	151
10.3.1 MifProcessor	151
10.3.2 MifObjectProcessor	151
10.3.3 MifMessageProcessor	152
10.3.4 Module Properties	152
10.4 Endpoints and Transports	153
10.4.1 Connectors	153
10.4.2 Supported Transports	154
10.5 MeDICI Example	157
10.5.1 Initialize Pipeline	158
10.5.2 Chat Component	159
10.5.3 Implementation code	161
10.6 Component Builder	161
10.7 Summary	163
10.8 Further Reading	163

11	Looking Forward	165
11.1	Introduction	165
11.2	The Challenges of Complexity	165
11.2.1	Business Process Complexity	166
11.3	Agility	167
11.4	Reduced Costs	168
11.5	What Next	169
12	The Semantic Web	171
12.1	ICDE and the Semantic Web	171
12.2	Automated, Distributed Integration and Collaboration	172
12.3	The Semantic Web	173
12.4	Creating and Using Metadata for the Semantic Web	174
12.5	Putting Semantics in the Web	176
12.6	Semantics for ICDE	178
12.7	Semantic Web Services	180
12.8	Continued Optimism	181
12.9	Further Reading	182
13	Aspect Oriented Architectures	185
13.1	Aspects for ICDE Development	185
13.2	Introduction to Aspect-Oriented Programming	186
13.2.1	Crosscutting Concerns	186
13.2.2	Managing Concerns with Aspects	187
13.2.3	AOP Syntax and Programming Model	188
13.2.4	Weaving	189
13.3	Example of a Cache Aspect	190
13.4	Aspect-Oriented Architectures	191
13.5	Architectural Aspects and Middleware	192
13.6	State-of-the-Art	193
13.6.1	Aspect Oriented Modeling in UML	193
13.6.2	AOP Tools	193
13.6.3	Annotations and AOP	194
13.7	Performance Monitoring of ICDE with AspectWerkz	195
13.8	Conclusions	197
13.9	Further Reading	198
14	Model-Driven Architecture	201
14.1	Model-Driven Development for ICDE	201
14.2	What is MDA?	203
14.3	Why MDA?	205
14.3.1	Portability	205
14.3.2	Interoperability	206
14.3.3	Reusability	207

14.4	State-of-Art Practices and Tools	208
14.4.1	AndroMDA	208
14.4.2	ArcStyler	209
14.4.3	Eclipse Modeling Framework	209
14.5	MDA and Software Architecture	210
14.5.1	MDA and Nonfunctional Requirements	211
14.5.2	Model Transformation and Software Architecture	211
14.5.3	SOA and MDA	212
14.5.4	Analytical Models are Models Too	212
14.6	MDA for ICDE Capacity Planning	214
14.7	Summary and Further Reading	216
15	Software Product Lines	219
15.1	Product Lines for ICDE	219
15.2	Software Product Lines	220
15.2.1	Benefiting from SPL Development	222
15.2.2	Product Lines for ICDE	223
15.3	Product Line Architecture	223
15.3.1	Find and Understand Software	224
15.3.2	Bring Software into the Development Context	225
15.3.3	Invoke Software	225
15.3.4	Software Configuration Management for Reuse	225
15.4	Variation Mechanisms	227
15.4.1	Architecture-Level Variation Points	227
15.4.2	Design-Level Variation	227
15.4.3	File-Level Variation	228
15.4.4	Variation by Software Configuration Management	228
15.4.5	Product Line Architecture for ICDE	228
15.5	Adopting Software Product Line Development	229
15.5.1	Product Line Adoption Practice Areas	231
15.5.2	Product Line Adoption for ICDE	231
15.6	Ongoing Software Product Line Development	232
15.6.1	Change Control	232
15.6.2	Architectural Evolution for SPL Development	233
15.6.3	Product Line Development Practice Areas	234
15.6.4	Product Lines with ICDE	234
15.7	Conclusions	235
15.8	Further Reading	236
Index	239	