Synthesis Lectures on
Human-Centered Informatics

SYNTHESIS
COLLECTION OF TECHNOLOGY

Jeff Johnson · Austin Henderson

Conceptual Models

Core to the Design of Interactive Applications

Second Edition

Springer

Synthesis Lectures on Human-Centered Informatics

Series Editor

John M. Carroll, College of Information Sciences and Technology, Penn State University, University Park, PA, USA

Jeff Johnson · Austin Henderson

Conceptual Models

Core to the Design of Interactive Applications

Second Edition

Springer

Jeff Johnson
Department of Computer Science
University of San Francisco
San Francisco, CA, USA

Austin Henderson
Rivendel Consulting and Design
Berkeley, CA, USA

ISSN 1946-7680 ISSN 1946-7699 (electronic)
Synthesis Lectures on Human-Centered Informatics
ISBN 978-3-031-50851-6 ISBN 978-3-031-50852-3 (eBook)
https://doi.org/10.1007/978-3-031-50852-3

Preface

We have been interested in Conceptual Models for many years. We both lived through the rough and tumble days of inventing the future at Xerox, and understand just how hard it has been (very) for the world to develop interactive applications that work even as well as they do. A continuing subject of discussion for many designers of interactive applications has been the "conceptual model of the system"—the view of the application that the designers hope people will adopt when using it.

Yet, there was a lack of clarity about exactly what conceptual models are. This is not entirely surprising, since so many different kinds of models and modeling are employed in the design and development of interactive applications.

For that reason, in 2002 we wrote an article for *Interactions* magazine—a 2500 word attempt to encourage designers to "begin by designing what to design". After it was published, we received feedback that the article was helpful to some designers and developers. Nonetheless, the use of conceptual models in software design and development remained fairly uncommon.

In the wake of the *Interactions* article, our own interest in conceptual models continued to evolve. We realized that there was more to say than we said in the article. We felt that conceptual models should be discussed more thoroughly and in a place that was readily available to all engaged in developing interactive applications or studying how to develop them. That was our motivation for the first edition of this book, which was published in 2011.

In the twelve years since then, the book has been used by many designers of user interfaces (UI), user experiences (UX), and even software. It has also served as the basis for many tutorials and courses on conceptual modeling that we presented to software design professionals. Several college professors, including one of us, also used the first edition as a textbook to teach conceptual modeling to college students—both undergraduates and graduates. Using the book in professional tutorials and college courses provided insight into where the book worked well, and where it did not.

Many who have used the first edition have provided us with feedback, either indirectly via ratings and comments at online booksellers, or directly via email and face-to-face at conferences. Some of the feedback was positive (see the first edition's ratings and comments at Amazon.com), and some of it pointed out deficiencies and ways in which the book could be improved. After Springer-Nature invited us to create a second edition, we compiled a list of improvements we wanted to make and solicited suggestions for improvement from others who had used the first edition.

In contrast to the first edition, the second edition:

- Summarizes early in the book the benefits of using conceptual models and how they fit into the design/development process.
- Expands the discussion of how conceptual modeling is related to other software design methods, and the explanations of what conceptual models are and are *not*.
- Defines a vocabulary for describing and discussing conceptual models, and adheres to that vocabulary throughout the book.
- Expands the explanation of "concept", "object", "attribute", and "operation" to try to overcome difficulties some designers—and design students—have in understanding those ideas.
- Updates outdated examples.
- Adds a second fully-worked-out realistic example of a conceptual model.
- Describes the user research methods that provide input for conceptual modeling.
- Clarifies the important components of a conceptual model.
- Adds a chapter on how to present conceptual models, specifically the results of an Objects/Operations analysis.
- Reorganizes and extends the discussion of conceptual modeling practices, discussing those practices that we consider essential and some that we consider optional and/or advanced.
- Expands the discussion of how conceptual models support collaboration between members of a design/development team.
- Briefly discusses what comes after conceptual modeling: proceeding from a conceptual model to the design of user interfaces.
- Adds an Appendix outlining the history of conceptual modeling and identifying subfields that have contributed to the practice.
- Provides chapter titles that are more self-explanatory than the first edition's cryptic one-word titles.

There are many people we would like to thank for their help in producing the two editions of this book.

We thank Jack Carroll for offering to publish the first edition in his *Synthesis Lectures on Human-Centered Informatics* series at Morgan & Claypool. His encouragement was supportive. We thank Diane Cerra for her help in setting direction and managing the mechanics of the first edition. We thank our colleagues Robin Jeffries, Jon Meads, Susan Fowler, Stuart Card, and Nigel Bevan for their helpful feedback on the first edition's first draft.

We thank Christine Killerich at Springer-Nature for inviting us to produce a second edition. We thank Hugh Dubberly, Prof. Ann Blandford, Jorge Arango, and Carola F. Thompson, for responding to our request for suggestions on how to improve the book, and Steven Pemberton for providing input on the Appendix. We also thank those who helped edit and design this book.

Finally, we each have people we would like to acknowledge:

Austin Henderson: I want to acknowledge many people for conversations over the years that have touched on conceptual models, particularly, Tom Moran, Stuart Card, John Rheinfrank, Shelley Evenson, Don Norman, David Asano, Jed Harris, and, of course, most centrally, Jeff Johnson. I want to thank my wife Lynne for her invaluable support and encouragement while this work has been underway, for two editions now.

Jeff Johnson: I acknowledge the many insights I have gained as a result of discussions—and arguments—about interaction design and conceptual models over the years with many colleagues with whom I have worked: Robin Jeffries, Bonnie Nardi, Steve Whittaker, Terry Roberts, Chuck Clanton, and of course, Austin Henderson. I also acknowledge the support and patience of my wife Karen Ande as both editions of this book were being written.

San Francisco, USA Jeff Johnson
Berkeley, USA Austin Henderson

Contents

1 Introduction 1
- 1.1 Where Does a Design Team Begin? 2
  - 1.1.1 Design (and Implement) the Functionality? 2
  - 1.1.2 Design (Sketch) the User Interface? 3
  - 1.1.3 Create a Conceptual Design 3
- 1.2 Conceptual Models 4
  - 1.2.1 An Example: Calendar 4
  - 1.2.2 Contrasting Conceptual Models 4
- 1.3 Important Decisions 5
- 1.4 Conceptual Design's Place in the Application Development Process 6
- 1.5 The Benefits of Using Conceptual Models 6
- 1.6 Organization of Book 7

2 Framework and Terminology 9
- 2.1 Framework for Interactive Applications and Their Use 9
  - 2.1.1 People and Domains 9
  - 2.1.2 Tasks 11
  - 2.1.3 Plans and Activities 11
  - 2.1.4 Applications and Uses 11
  - 2.1.5 Conceptual Models 12
- 2.2 Beyond Conceptual Models 12
  - 2.2.1 User Interfaces 13
  - 2.2.2 Implementation 14
- 2.3 Thinking About Applications 15
  - 2.3.1 User's Mental Model 15
  - 2.3.2 Designers' Conceptual Model 16
  - 2.3.3 Terms for Conceptual Modeling 16
- 2.4 Descriptions of Applications 16
- 2.5 Designer Practice: Designing with Conceptual Models 18

3 What Conceptual Models Are and Are Not 21
3.1 What are Models? 21
3.2 What Are *Conceptual* Models? 22
3.2.1 Unsatisfactory Descriptions of Applications 22
3.2.2 Better: Task-Level Descriptions 22
3.2.3 Best: Object-Operations Model Descriptions 23
3.2.4 Our Approach to Conceptual Modeling 23
3.2.5 High-Level Description of an Application 24
3.2.6 Basis for Users' Mental Model of the Application 25
3.2.7 Design Goals for Conceptual Models 26
3.2.8 Summary: What a Conceptual Model is 28
3.2.9 Close Relative: Information Architecture 28
3.3 What Conceptual Models Are *Not* 29
3.3.1 Not Users' Mental Model 29
3.3.2 Not a Design Metaphor 29
3.3.3 Not Just Task Models, Task-Level Scenarios, or Use Cases 30
3.3.4 Not Based on the User Interface 31
3.3.5 Not Formal Specifications to Automatically Generate or Simulate Applications 32
3.3.6 Not the Implementation Architecture or an Implementation-Based Description 32
3.3.7 Not Just the Information or Content Architecture 33
3.3.8 Not Product Designer's "Concept Design" 33
3.4 Summary: Our View of Conceptual Models 34
3.4.1 Conceptual Models *Are* 34
3.4.2 Conceptual Models Are *Not* 34

4 Components of a Conceptual Model 37
4.1 Domain, Purpose, and High-Level Functionality 37
4.2 Concepts and Vocabulary 38
4.3 Structure: Objects/Operations Analysis 38
4.3.1 Declares Concepts that the Application Will Expose 39
4.3.2 Introduces New Concepts, if Needed 40
4.3.3 Shows Relationships Between Concepts 41
4.4 Conceptual Scenarios 44
4.5 Conceptual Model Issues 45
4.6 Resolved Issues 46
4.7 Open Issues 46

5 Representing Conceptual Models . . . 47
5.1 Introduction . . . 47
5.2 Outline . . . 48
5.3 Table or Spreadsheet . . . 48
5.4 Unified Modeling Language (UML) Diagram . . . 49
5.5 Entity-Relationship (ER) Diagram or Concept Map . . . 51
5.6 Conceptual Scenario . . . 51

6 Two Complete Examples . . . 55
6.1 Example 1: Server Management Console . . . 55
6.1.1 Purpose . . . 56
6.1.2 High-Level Functionality . . . 56
6.1.3 Major Concepts and Vocabulary . . . 56
6.1.4 Conceptual-Level Scenarios . . . 57
6.1.5 Resolved Conceptual Design Issues . . . 58
6.1.6 Open Conceptual Design Issues . . . 58
6.2 Example 2: Web-Search Service for Professionals . . . 59
6.2.1 Domain and Purpose . . . 59
6.2.2 High-Level Functionality . . . 59
6.2.3 Major Concepts and Vocabulary . . . 60
6.2.4 Objects/Operations Structure . . . 61
6.2.5 Task Hierarchy Enumeration (from Task Analysis) . . . 65
6.2.6 Conceptual-Level Scenarios . . . 66
6.2.7 Resolved Conceptual Design Issues . . . 67
6.2.8 Open Conceptual Design Issues . . . 67

7 Essential Conceptual Modeling . . . 69
7.1 Basic Objects/Operations Analysis . . . 69
7.1.1 First Step: Identify the Objects, Attributes, Attribute Values, and Operations . . . 70
7.1.2 Objects Are UI-Independent User-Manipulable Content . . . 71
7.1.3 Assign Operations and Attributes to Objects . . . 71
7.1.4 Assign Operations to the Appropriate Object(S) . . . 71
7.1.5 Decide How to Model Similar Objects . . . 72
7.1.6 Decide Whether to Include and Expose a Parent Object as Well as Its Subtypes . . . 72
7.1.7 Identify the Attributes of Each Object-Type . . . 74
7.1.8 Decide What Type of Values an Attribute Has . . . 74
7.1.9 Watch Out for Hidden Objects . . . 74
7.1.10 Decide How Detailed to Be in Modeling Common Operations . . . 75
7.1.11 Include All Task-Relevant Operations . . . 76

7.1.12 Remove Obsolete Concepts 76
7.1.13 Part-of and Containment Relationships Need Not Always Be Distinguished 77
7.1.14 Include Software Developers Who Have Experience with Object-Oriented Programming 77
7.2 Supporting Learning 77
7.2.1 Metaphors 78
7.2.2 Consistent Terminology 78
7.3 Conceptual Model Versus User Interface 79
7.3.1 UI Terms Versus Conceptual Terms 79
7.3.2 Interactive Concepts 80
7.4 Object Identity 80
7.4.1 Containment 80
7.4.2 Synchronizing Objects 81
7.4.3 Inheriting Attributes 81
7.4.4 What Work is Saved, When? And Can I Reverse It? 82

8 Enhanced Conceptual Modeling 83
8.1 Using Companion Models 84
8.1.1 Progressive Disclosure 84
8.1.2 Component Models 84
8.1.3 Surrounding Models 85
8.2 Modeling Interactions 87
8.2.1 Managing Errors 87
8.2.2 Anticipating Trouble 89
8.2.3 Macros: Capturing Activity 89
8.3 Evolving the Application 90
8.3.1 Managed Growth 90
8.3.2 Anticipated Growth 90
8.3.3 Unanticipated Growth 93
8.3.4 Embedding in Social Domains 96

9 Process of Designing with Conceptual Models 97
9.1 Overview of the Process 97
9.2 Start with User Research to Understand Users and Tasks 98
9.2.1 User Profiles 98
9.2.2 Task Analysis 100
9.3 Based on User Research, Start Designing a Conceptual Model 102
9.3.1 The Conceptual Model is Central to the Design Process 102
9.3.2 Coordination is Required 103
9.3.3 One Team Member Should Drive the Conceptual Design 103

9.3.4 Include Developers, but Keep the Conceptual Model Focused on User-Facing Objects and Tasks . . . 104
9.4 Use the Conceptual Model to Coordinate Development . . . 104
9.5 Iterate, Iterate, Iterate . . . 105
9.5.1 Including Conceptual Models in Agile Development . . . 107
9.5.2 Testing Conceptual Models . . . 108
9.6 Getting from CM to UI . . . 109
9.6.1 Object-Based Versus Task-Based UIs . . . 110

10 Benefits of Designing with Conceptual Models . . . 113
10.1 Produces a Vocabulary . . . 113
10.2 Facilitates Creation of Conceptual Scenarios . . . 117
10.3 Facilitates Creation of User Documentation, Training, and Support . . . 118
10.4 Focuses User Interface Design: Gives Designers a Clear Target . . . 118
10.5 Jump-Starts and Focuses the Implementation . . . 119
10.6 Supports Further Task Analysis . . . 120
10.7 Supports Controlled Iteration of a Design . . . 120
10.8 Facilitates Communication Between Team Members . . . 121
10.9 Saves Time and Money . . . 121
10.10 Conclusion . . . 122

Glossary of Terms Used in This Book . . . 123

Appendix: Origins, History, and Contributing Fields . . . 125

Bibliography . . . 137

About the Authors

Jeff Johnson is an Adjunct Professor of Computer Science at the University of San Francisco. He is also President and Principal Consultant at UI Wizards, Inc., a product usability consulting firm. After earning B.A. and Ph.D. degrees from Yale and Stanford Universities, he worked as a UI designer and implementer, engineer manager, usability tester, and researcher at Cromemco, Xerox, US West, Hewlett-Packard Labs, and Sun Microsystems. In the late 1980s and early 1990s, he was Chair of Computer Professionals for Social Responsibility. In 1990, he co-chaired the first Participatory Design conference, PDC'90. He has taught at Stanford University and Mills College, and in 2006 and 2013 taught as an Erskine Fellow at the University of Canterbury in New Zealand. Since 2004 he has served on the SIGCHI Public Policy Committee. He has spoken twice in the prestigious Authors@Google talk series, in 2013 and 2017. He is a member of the ACM SIGCHI Academy, a recipient of SIGCHI's Lifetime Achievement in Practice Award, and an ACM Distinguished Member. He has authored or co-authored many articles and chapters on Human-Computer Interaction, as well as the books *GUI Bloopers, Web Bloopers, GUI Bloopers 2.0, Designing with the Mind in Mind, Conceptual Models: Core to Good Design* (with Austin Henderson), *Designing with the Mind in Mind, 2nd edition, Designing User Interfaces for an Aging Population* (with Kate Finn), and *Designing with the Mind in Mind, 3rd edition, and Conceptual Models: Core to Good Design, 2nd edition* (with Austin Henderson).

Austin Henderson 50-year career in Human-Computer Interaction includes user interface research and architecture at MIT's Lincoln Laboratory, Bolt Beranek and Newman, Xerox Research (both PARC and EuroPARC), Apple Computer, and Pitney Bowes, as well as strategic industrial design with Fitch, and his own Rivendel Consulting & Design. Austin has built both commercial and research applications in many domains including manufacturing, programming languages, air traffic control, electronic mail (Hermes), user interface design tools (Trillium), workspace management (Rooms, Buttons), distributed collaboration (MediaSpace), and user-evolvable systems (tailorable—"design continued in use", pliant—"designing for the unanticipated" and ontologically extensible—"scalable conversations"). These applications, and their development with users, have grounded his analytical work, which has included the nature of computation-based socio-technical systems, the interaction of people with the technology in those systems, and the practices and tools of their development. The primary goals of his work have been to better meet user needs, both by improving system development to better anticipate those needs, and by broadening system capability to enable users themselves to better respond to new unanticipated needs when they arise in a rich and changing world.