Conceptual Models

Core to Good Design

Synthesis Lectures on Human-Centered Informatics

Editor
John M. Carroll, ***Penn State University***

Human-Centered Informatics (HCI) is the intersection of the cultural, the social, the cognitive, and the aesthetic with computing and information technology. It encompasses a huge range of issues, theories, technologies, designs, tools, environments and human experiences in knowledge work, recreation and leisure activity, teaching and learning, and the potpourri of everyday life. The series will publish state-of-the-art syntheses, case studies, and tutorials in key areas. It will share the focus of leading international conferences in HCI.

Conceptual Models: Core to Good Design
Jeff Johnson and Austin Henderson
2011

Geographical Design: Spatial Cognition and Geographical Information Science
Stephen C. Hirtle
2011

User-Centered Agile Methods
Hugh Beyer
2010

Experience-Centered Design: Designers, Users, and Communities in Dialogue
Peter Wright and John McCarthy
2010

Experience Design: Technology for All the Right Reasons
Marc Hassenzahl
2010

Designing and Evaluating Usable Technology in Industrial Research: Three Case Studies
Clare-Marie Karat and John Karat
2010

Interacting with Information
Ann Blandford and Simon Attfield
2010

Designing for User Engagement: Aesthetic and Attractive User Interfaces
Alistair Sutcliffe
2009

Context-Aware Mobile Computing: Affordances of Space, Social Awareness, and Social Influence
Geri Gay
2009

Studies of Work and the Workplace in HCI: Concepts and Techniques
Graham Button and Wes Sharrock
2009

Semiotic Engineering Methods for Scientific Research in HCI
Clarisse Sieckenius de Souza and Carla Faria Leitão
2009

Common Ground in Electronically Mediated Conversation
Andrew Monk
2008

© Springer Nature Switzerland AG 2022

Reprint of original edition © Morgan & Claypool 2012

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means—electronic, mechanical, photocopy, recording, or any other except for brief quotations in printed reviews, without the prior permission of the publisher.

Conceptual Models: Core to Good Design

Jeff Johnson and Austin Henderson

ISBN: 978-3-031-01067-5 paperback
ISBN:978-3-031-02195-4 ebook

DOI 10.1007/978-3-031-02195-4

A Publication in the Springer series
SYNTHESIS LECTURES ON HUMAN-CENTERED INFORMATICS

Lecture #12
Series Editor: John M. Carroll, *Penn State University*
Series ISSN
Synthesis Lectures on Human-Centered Informatics
Print 1946-7680 Electronic 1946-7699

Conceptual Models

Core to Good Design

Jeff Johnson
UI Wizards, Inc.

Austin Henderson
Rivendel Consulting & Design

SYNTHESIS LECTURES ON HUMAN-CENTERED INFORMATICS #12

ABSTRACT

People make use of software applications in their activities, applying them as tools in carrying out tasks. That this use should be good for people – easy, effective, efficient, and enjoyable – is a principal goal of design. In this book, we present the notion of Conceptual Models, and argue that Conceptual Models are core to achieving good design. From years of helping companies create software applications, we have come to believe that building applications without Conceptual Models is just asking for designs that will be confusing and difficult to learn, remember, and use.

We show how Conceptual Models are the central link between the elements involved in application use: people's tasks (task domains), the use of tools to perform the tasks, the conceptual structure of those tools, the presentation of the conceptual model (i.e., the user interface), the language used to describe it, its implementation, and the learning that people must do to use the application. We further show that putting a Conceptual Model at the center of the design and development process can pay rich dividends: designs that are simpler and mesh better with users' tasks, avoidance of unnecessary features, easier documentation, faster development, improved customer uptake, and decreased need for training and customer support.

Supplemental Website, including figures:
`http://sites.google.com/a/morganclaypool.com/conceptual-models`

KEYWORDS

conceptual model, conceptual design, concepts, task-domain, tasks, tools, task-to-tool mapping, object/operations analysis, interaction design, user interface design, application design, software design, design, usability, software development method, software development process

Contents

Preface . . . . . **xi**

Acknowledgments . . . . . **xiii**

Introduction **1**

1 Using Tools **5**

1.1 Activity 5
1.2 Tasks 5
1.3 Task Domains 5
1.4 Tools 7
1.5 Function 7
1.6 Task-to-tool Mapping 7
1.7 Describing Tools 8
1.7.1 Task-Based Descriptions of Tools 8
1.7.2 Model-Based Descriptions 8
1.8 Seeing the Model Through the User Interface 9
1.9 Implementation Architecture and Implementations 9
1.10 Mental Models 9
1.11 Conceptual Models 10

2 Start with the Conceptual Model **13**

2.1 Design (and Implement) Functionality? 13
2.2 Duplicate the Existing Activity? 14
2.3 Design (Sketch) the User Interface? 14
2.4 Start with *Conceptual* Design 14
2.5 Important Decisions 15
2.6 Conceptual Design's Place in Software Development 15

3 Definition **17**

3.1 What a Conceptual Model Is 17
3.1.1 High-Level Description of an Application 17

3.1.2 Basis for Users' Understanding of the Application 18
3.1.3 Close Relative: Information Architecture 19
3.1.4 Summary: What a Conceptual Model Is 20
3.2 What a Conceptual Model Is *Not* 20
3.2.1 Not Task-Level Scenarios or Use Cases 20
3.2.2 Not Users' Mental Model 21
3.2.3 Not a Design Metaphor 21
3.2.4 Not the User Interface 22
3.2.5 Not the Implementation Architecture 22
3.2.6 Not Product Designer's "Concept Design" 23
3.3 Design Goals for Conceptual Models 24
3.3.1 Simple 24
3.3.2 Task-Focused 24

4 Structure 27

4.1 Purpose & High-Level Functionality 27
4.2 Major Concepts and Vocabulary 27
4.3 Objects/Operations Analysis 30
4.3.1 Declares Concepts that the Application will Expose 31
4.3.2 Introduces New Concepts, if Needed 31
4.3.3 Hides Concepts that Once Were Necessary but No Longer Are 32
4.3.4 Shows Relationships Between Concepts 33
4.4 Example of Objects/Operation Analysis 34
4.5 Conceptual Design Issues 35
4.6 Mapping from Task-Domain 36

5 Example 39

5.1 Overall Purpose 39
5.2 High-Level Functionality 40
5.3 Major Concepts and Vocabulary 40
5.4 Object-Operations Analysis 42
5.4.1 Formatting Conventions 42
5.4.2 Object-Operations Hierarchy 42
5.5 Mapping: Task Hierarchy Enumeration 46
5.6 Resolved Conceptual Design Issues 47
5.7 Open Conceptual Design Issues 48

6 Essential Modeling . . . 49

6.1 Basic Object/Operations Analysis . . . 49
6.1.1 Assign Operations and Attributes to Objects . . . 49
6.1.2 Assign Operations to the Appropriate Object(s) . . . 49
6.1.3 Decide How to Model Similar Objects . . . 50
6.1.4 Decide Whether to Include the Generic Object Type . . . 51
6.1.5 Decide What Type of Values an Attribute Has . . . 51
6.1.6 Decide How Detailed to be in Modeling Common Operations . . . 52
6.1.7 Include All Task-Relevant Operations . . . 52
6.1.8 Part-of and Containment Relationships Need Not Always be Distinguished . . . 53
6.2 Supporting Learning . . . 53
6.2.1 Metaphors . . . 53
6.2.2 Progressive Disclosure . . . 54
6.2.3 Component Models . . . 54
6.2.4 Surrounding Models . . . 54
6.2.5 Object-Oriented vs. Task-Oriented User Interfaces . . . 56
6.3 Conceptual Model vs. User Interface . . . 58
6.3.1 View vs. Focus . . . 58
6.3.2 Interactive Concepts . . . 58
6.4 Object Identity . . . 59
6.4.1 Containment . . . 59
6.4.2 Synchronizing Objects . . . 59
6.4.3 Inheriting Attributes . . . 60
6.4.4 What Work is Saved, When? And Can I Reverse It? . . . 60

7 Optional Modeling . . . 61

7.1 Going Meta: Activity as Objects . . . 61
7.1.1 Managing Trouble . . . 61
7.1.2 Anticipating Trouble . . . 63
7.1.3 Macros: Capturing Work Activity . . . 63
7.2 Evolving the Application . . . 64
7.2.1 Managed Growth . . . 64
7.2.2 Anticipated Growth . . . 64
7.2.3 Versioning . . . 66
7.2.4 Unanticipated Growth . . . 67
7.2.5 Embedding in Social Domains . . . 69

8 Process **71**
8.1 First Step of Design 71
8.2 Start with User Research 71
8.3 The Conceptual Model Needs a Place at the Project Table 72
8.3.1 Coordination is Required 73
8.3.2 One Team-member Should Drive the Conceptual Design 73
8.3.3 Include Developers, but Keep the Conceptual Model Focused on Tasks . . 74
8.4 Use the Conceptual Model to Coordinate Development 74
8.5 Representing Conceptual Models 74
8.6 Iterate, Iterate, Iterate 75
8.7 Including Conceptual Models in Agile Development 77
8.8 Testing Conceptual Models 79

9 Value **81**
9.1 Produces a Vocabulary 81
9.2 Facilitates Creation of High-Level Task Scenarios 84
9.3 Facilitates Creation of User Documentation, Training, and Support 85
9.4 Focuses the User Interface Design: Gives Designers a Clear Target 86
9.5 Jump-Starts and Focuses the Implementation 87
9.6 Saves Time and Money 87

10 Epilogue **89**

Bibliography **91**

Authors' Biographies **95**

Preface

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

Yet in all this time, there has been a lack of clarity about exactly what these models are. This is not entirely surprising, seeing as there are so many different kinds of models, and modeling is such an endemic effort in the development of systems.

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". In the intervening decade, we have received sporadic reports that that article has been helpful to some designers and developers. However, conceptual models do not seem to have become common in accepted practice.

We also noticed that our own interest in conceptual models continued to evolve. We realized that there was much more to say than we said in the Interactions article. We felt that conceptual models should be discussed more thoroughly and in a place that was readily available to all engaged in developing applications or studying how to develop them.

This book is the result.

Jeff Johnson and Austin Henderson
November 2011

Acknowledgments

We thank Jack Carroll for offering to publish this in his "Human-Centered Informatics" series at Morgan & Claypool. His encouragement has been supportive. We thank Diane Cerra for her help in setting direction and managing the mechanics of this undertaking. We thank our colleagues Robin Jeffries, Jon Meads, Susan Fowler, and Nigel Bevan for their helpful feedback on our first draft. We also thank Jon Meads for allowing us to use two of his diagrams (Fig. 8.3 courtesy of Jon Meads, copyright © 2011 Jon Meads). We 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, Hugh Dubberly, Jed Harris and, of course, most centrally, Jeff Johnson. I want to thank my wife Lynne for invaluable support and encouragement while this work has been underway.

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, Stuart Card, and of course, Austin Henderson. I also acknowledge the support and patience of my wife Karen Ande as this book was being written.

Jeff Johnson and Austin Henderson
November 2011