# Using Design Patterns and Layers to Support the Early-Stage Design and Prototyping of Cross-Device User Interfaces

by

James Lin

B.S. (California Institute of Technology) 1997
M.S. (University of California, Berkeley) 2000

A dissertation submitted in partial satisfaction of the requirements for the degree of

Doctor of Philosophy

in

Computer Science

in the
GRADUATE DIVISION
of the
UNIVERSITY OF CALIFORNIA, BERKELEY

Fall 2005

The dissertation of James Lin is approved:

| | |
|---|---|
| Co-chair | Date |
| Co-chair | Date |
| | Date |
| | Date |

University of California, Berkeley

Fall 2005

**Using Design Patterns and Layers to Support the Early-Stage Design and Prototyping of Cross-Device User Interfaces**

by

James Lin

ABSTRACT

**Using Design Patterns and Layers to Support the Early-Stage Design and Prototyping of Cross-Device User Interfaces**

by

James Lin

Doctor of Philosophy in Computer Science

University of California, Berkeley

Professor James Anthony Landay, Co-chair

Professor John Canny, Co-chair

People often use a variety of computing devices, such as PCS, PDAS, and cell phones, to access the same information. The user interface to this information needs to be different for each device, due to the different input and output constraints of each device. Currently designers designing such cross-device user interfaces either have to design a UI separately for each device, which is time consuming, or use a program to automatically generate interfaces, which often result in interfaces that are awkward. Each method also discourages iterative design, considered critical for creating good user interfaces.

I have created a system called Damask to support the early-stage design of user interfaces targeted at multiple devices. Within Damask, designers use *layers* to specify which parts of a user interface is common across all devices and which are specific to one device. They use *design patterns* to specify higher-level concepts within a user interface. Design patterns in Damask include pre-built UI fragments that are already optimized for each device. Designers can instantiate patterns in their designs and then customize the instances to fit their particular requirements.

Through a study performed with twelve professional web designers, we have found that, in the early stages of design, designers using patterns and layers in Damask create cross-device user interfaces that are as good as or better than those created without patterns and layers, in less time.

Professor James Anthony Landay
Dissertation Committee Co-chair

Professor John Canny
Dissertation Committee Co-chair

*To my parents, Alice and Bob Lin*

# Table of Contents

**CHAPTER 1 INTRODUCTION 1**

1.1 Creating Cross-device Interfaces 4

1.2 Dissertation Contributions 15

1.3 Design Patterns 16

1.4 Model-Based User Interfaces 17

1.5 Outline 18

**CHAPTER 2 STUDY OF CURRENT PRACTICES IN CROSS-DEVICE UI DESIGN 19**

2.1 Roles and Responsibilities of the Designers 20

2.2 Scope of Cross-Device Projects 21

2.3 Organization of Project Teams 21

2.4 Managing UI Consistency Across Devices 23

2.5 Tools, Documentation, and Patterns 23

2.6 Need for Synchronized Changes Across Devices 24

2.7 Implications for Cross-Device UI Design Tools 25

**CHAPTER 3 PROTOTYPE OF CROSS-DEVICE DESIGN TOOL 27**

3.1 User Interface 27

3.2 Architecture 30

3.3 Evaluation of HopiSketch 32

*3.3.1 Participants 32*

*3.3.2 Methodology 32*

*3.3.3 Results 33*

3.4 Implications for Cross-Device UI Design Tools 35

**CHAPTER 4 DAMASK'S USER INTERFACE 41**

4.1 Damask's User Interface 43

4.2 Designing Desktop and Mobile Phone UIs 44

*4.2.1 Pages 44*

*4.2.2 Adding user interface controls to pages 48*

*4.2.3 Specifying behavior with arrows 52*

*4.2.4 Manipulating objects within a page 54*

*4.2.5 Templates 54*

*4.2.6 Run window 56*

4.3 Designing Voice UIs 57

*4.3.1 Forms 58*

*4.3.2 Prompts 58*

*4.3.3 Responses 59*

*4.3.4 Universals 61*

*4.3.5 Run window 61*

*4.3.6 Exporting to VoiceXML 62*

*4.3.7 Limitations of the VoiceXML export feature and the Run window 63*

4.4 Layers 64

*4.4.1 Relationship between desktop/smartphone and voice UI elements 67*

*4.4.2 Synchronizing text between desktop/smartphone UIs and voice UIs 71*

4.5 Patterns 72

4.6 Summary 78

**CHAPTER 5 ARCHITECTURE AND IMPLEMENTATION OF DAMASK 79**

5.1 Implementation of Model Objects 80

*5.1.1 Interaction graph and interaction elements 80*

*5.1.2 Layers 82*

*5.1.3 Dialogs and templates 83*

*5.1.4 Components 84*

*5.1.5 Connections 86*

*5.1.6 Correspondence between model and voice view 87*

*5.1.7 Patterns and pattern instances 90*

5.2 Implementation of View Objects 91

*5.2.1 Canvas 91*

*5.2.2 Views of interaction elements 92*

*5.2.3 Pages and controls 93*

*5.2.4 Layers 97*

*5.2.5 Templates 97*

*5.2.6 Patterns and pattern instances 98*

5.3 Run mode 98

5.4 VoiceXML Generation 99

5.5 Summary 105

**CHAPTER 6 DAMASK EVALUATION 107**

6.1 Experimental Procedure 108

*6.1.1 Phase 1: Study of desktop and smartphone user interface design 109*

*6.1.2 Phase 2: Study of desktop and voice user interface design 111*

6.2 Participants 112

6.3 Introduction to the Results 116

6.4 Results Concerning All Designs Created by the Participants 116

*6.4.1 Time spent by participants 116*

*6.4.2 Extent of designs created by participants 118*

*6.4.3 Preference of designs created by participants 120*

6.5 Desktop Designs Created by the Participants 120

*6.5.1 Examples of desktop designs 121*

*6.5.2 Quality analysis of desktop designs 124*

*6.5.3 Functional analysis of desktop designs 133*

*6.5.4 Structural analysis of desktop designs 134*

6.6 Smartphone Designs Created by the Participants 143

*6.6.1 Examples of smartphone designs 144*

*6.6.2 Quality analysis of smartphone designs 147*

6.7 Voice Designs Created by the Participants 154

*6.7.1 Examples of voice designs 154*

*6.7.1 Quality analysis of voice designs 157*

6.8 Pattern Usage and Ratings 159

6.9 Layers Usage and Ratings 165

6.10 Usability of Damask 166

6.11 Summary 171

**CHAPTER 7 RELATED WORK 173**

7.1 Model-Based UI Tools 173

*7.1.1 Automatic interface design tools 173*

*7.1.2 Specification-based model-based interface development environments 175*

*7.1.3 Modeling tools 176*

*7.1.4 Design critics and advisors 177*

*7.1.5 Prototyping vs. finished interfaces 178*

7.2 UI Tools and Languages for Multiple Devices 178

7.3 Tool Support for Patterns 181

7.4 Combining Models and Patterns 183

7.5 User Interface Transformation Tools 184

7.6 User Interface Design Tools 185

*7.6.1 Research web and GUI design tools 185*

*7.6.2 Commercial web and GUI design tools 186*

*7.6.3 Commercial mobile UI design tools 188*

*7.6.4 Voice UI design tools 188*

7.7 Summary 189

**CHAPTER 8 FUTURE WORK 191**

8.1 Patterns 191

*8.1.1 Creating and sharing patterns 191*

*8.1.2 Showing previous uses of patterns 192*

*8.1.3 Handling large patterns 193*

*8.1.4 Recognizing patterns 195*

*8.1.5 Versioning of patterns 196*

*8.1.6 Scope of patterns 196*

8.2 Annotations 197

8.3 Layers 198

*8.3.1 The "All" tab 198*

*8.3.2 Device-specific tools 199*

*8.3.3 Trays 200*

8.4 Synchronization Between Voice and Graphical UI Designs 203

8.5 Extending Damask's Visual Language 204

8.6 Summary 204

**CHAPTER 9 CONCLUSION 205**

9.1 Benefits of Damask's Approach 205

9.2 Limitations of Damask's Approach 206

9.3 Contributions 206

9.4 Final Remarks 208

**BIBLIOGRAPHY 209**

**APPENDIX A QUESTIONS FOR STUDY ON CURRENT PRACTICES 231**

**APPENDIX B PROTOTYPE EVALUATION AND QUESTIONNAIRE 235**

B.1 HopiSketch Test Plan 235

*B.1.1 Roles 235*

*B.1.2 Introduction 235*

*B.1.3 Tasks 236*

*B.1.4 Post-test questions 237*

*B.1.5 Debriefing 237*

B.2 Consent Form for HopiSketch User Study 238

B.3 Task 1 239

B.4 Task 2 239

B.5 Post-Test Questionnaire 241

**APPENDIX C DAMASK EVALUATION MATERIALS 255**

C.1 Consent Forms 255

*C.1.1 Consent form for desktop/smartphone participants 255*

*C.1.2 Consent form for desktop/voice participants 257*

*C.1.3 Video records release consent form 259*

C.2 Oral Instructions 260

C.3 Damask Tutorial 261

C.4 Introductory Session for All Participants 262

C.5 Introductory Session for Desktop/Voice Participants 262

*C.5.1 Task 1 262*

*C.5.2 Task 2 263*

C.6 No Patterns or Layers Condition 265

*C.6.1 Desktop/smartphone participants 265*

*C.6.2 Desktop/voice participants 268*

*C.6.3 Questionnaire A 270*

C.7 With Patterns and Layers Condition 286

*C.7.1 Oral introduction to layers 286*

*C.7.2 Task 1, Part 1 287*

*C.7.3 Oral introduction to pattern browser 287*

*C.7.4 Patterns quiz 288*

*C.7.5 Task 1, Part 2 289*

*C.7.6 Task 2 290*

*C.7.7 Questionnaire B 295*

C.8 Summary of Likes and Dislikes of Damask 318

*C.8.1 What did you like about Damask? (Section C.6.3, Question 3 and Section C.7.7, Question 3) 318*

*C.8.2 What do you not like about Damask? What additional features would you like to see? (Section C.6.3, Questions 4 and 5, and Section C.7.7, Questions 4 and 7) 318*

C.9 Debriefing 319

**APPENDIX D DESIGNS CREATED IN DAMASK EVALUATION 321**

D.1 Key for Sitemaps 321

D.2 Designer 4 322

*D.2.1 TotalMusic (no layers or patterns) 322*

*D.2.2 TotalBooks (with layers and patterns) 325*

D.3 Designer 5 332

*D.3.1 TotalMusic (no layers or patterns) 332*

*D.3.2 TotalBooks (with layers and patterns) 334*

D.4 Designer 6 339

*D.4.1 TotalMusic (no layers or patterns) 339*

*D.4.2 TotalBooks (with layers and patterns) 342*

D.5 Designer 7 349

*D.5.1 TotalMusic (no layers or patterns) 349*

*D.5.2 TotalBooks (with layers and patterns) 352*

D.6 Designer 8 358

*D.6.1 TotalMusic (no layers or patterns) 358*

*D.6.2 TotalBooks (with layers and patterns) 361*

D.7 Designer 9 365

*D.7.1 TotalMusic (no layers or patterns) 365*

*D.7.2 TotalBooks (with layers and patterns) 368*

D.8 Designer 10 375

*D.8.1 TotalMusic (no layers or patterns) 375*

*D.8.2 TotalBooks (with layers and patterns) 378*

D.9 Designer 11 383

*D.9.1 TotalMusic (no layers or patterns) 383*

*D.9.2 TotalBooks (with layers and patterns) 387*

D.10 Designer 13 392

*D.10.1 TotalMusic (no layers or patterns) 392*

*D.10.2 TotalBooks (with layers and patterns) 394*

D.11 Designer 15 397

*D.11.1 TotalMusic (no layers or patterns) 397*

*D.11.2 TotalBooks (with layers and patterns) 399*

D.12 Designer 16 401

*D.12.1 TotalMusic (no layers or patterns) 401*

*D.12.2 TotalBooks (with layers and patterns) 403*

D.13 Designer 17 406

*D.13.1 TotalMusic (no layers or patterns) 406*

*D.13.2 TotalBooks (with layers and patterns) 408*

**APPENDIX E DESIGNS AND SITEMAPS OF AMAZON.COM AND BARNESANDNOBLE.COM 411**

E.1 Amazon.com — as of April 26, 2005 411

*E.1.1 Sitemap 411*

*E.1.2 Pages 412*

E.2 BarnesAndNoble.com — as of April 27, 2005 420

*E.2.1 Sitemap 420*

*E.2.2 Pages 421*

**APPENDIX F MATERIALS FOR JUDGING DAMASK DESKTOP AND SMARTPHONE USER INTERFACES 431**

F.1 Consent Form 431

F.2 Introduction 433

F.3 Questionnaire 435

**APPENDIX G MATERIALS FOR JUDGING DAMASK VOICE USER INTERFACES 541**

G.1 Consent Form 541

G.2 Directions 543

G.3 Tasks 543

G.4 Questionnaire 546

*G.4.1 Questions 546*

*G.4.2 Design 1: TotalMusic (Designer 13, no patterns or layers) 546*

*G.4.3 Design 2: TotalBooks (Designer 13, with patterns and layers) 547*

*G.4.4 Design 3: TotalBooks (Designer 15, with patterns and layers) 549*

*G.4.5 Design 4: TotalMusic (Designer 15, no patterns or layers) 550*

*G.4.6 Design 5: TotalBooks (Designer 16, with patterns and layers) 552*
*G.4.7 Design 6: TotalMusic (Designer 16, no patterns or layers) 553*
*G.4.8 Design 7: TotalMusic (Designer 17, no patterns or layers) 554*
*G.4.9 Design 8: TotalBooks (Designer 17, with patterns and layers) 556*

# List of Figures

FIGURE 1-1 Damask's user interface 3

FIGURE 1-2 *Top* A desktop web page sketched by a designer in Damask. *Bottom* The smartphone (left) and voice (right) versions of that page generated by Damask. 5

FIGURE 1-3 Adding objects to the *This Device* layer in Desktop view, which does not affect the Smartphone and Voice views. The objects previously drawn in the *All Devices* layer are grayed out, since the *This Device* layer is the current layer. 6

FIGURE 1-4 Smoothing out the voice interface. 7

FIGURE 1-5 The Pattern Explorer with the SHOPPING CART pattern selected. 8

FIGURE 1-6 The generalized solutions for SHOPPING CART. *Top* The desktop and smartphone versions. *Bottom* The voice version. 9

FIGURE 1-7 The shopping cart added to the main design. 10

FIGURE 1-8 The desktop version of the shopping cart, with the contents, prices, and quantities changed by the designer. 11

FIGURE 1-9 The smartphone (top) and voice (bottom) versions of the shopping cart, customized by Damask. Note that the contents of the cart have changed. 12

FIGURE 1-10 *Top* A designer adding an "Add to Cart" button with a link to the "Shopping Cart" page in *Desktop* view. *Bottom* An "Add to Cart" button linked to the Shopping Cart page in *Smartphone* view that was automatically created by Damask. 13

FIGURE 1-11 An Add to Cart response linked to the Shopping Cart form that was automatically created by Damask. 14

FIGURE 1-12 The Add to Cart response after it was fixed up by the designer. 15

FIGURE 3-1 DENIM showing a typical design. 28

FIGURE 3-2 a) A page with the label "Home" b) An arrow, whose source is a blue hyperlink, "Business." 29

FIGURE 4-1 Damask's user interface. 42

**FIGURE 4-2** Damask's pattern browser. 44

**FIGURE 4-3** Creating a page. 45

**FIGURE 4-4** A page, with gray bars for dragging and resizing regions and a red dotted "fold" line. 46

**FIGURE 4-5** The tools for splitting a page (above) and merging pages (below). 47

**FIGURE 4-6** *Left* Using the split tool to split a page. *Right* The result of splitting a page.48

**FIGURE 4-7** The toolbox buttons used for creating and erasing controls. 48

**FIGURE 4-8** Adding an ink stroke to a page. 49

**FIGURE 4-9** Grouping labels together into one label, by clicking the Group button in the toolbar. 49

**FIGURE 4-10** Changing the display mode of a label. 50

**FIGURE 4-11** Changing the state of a radio button. 51

**FIGURE 4-12** A panel containing a label and radio buttons in the canvas. The dotted border is invisible at run time. 52

**FIGURE 4-13** An organizational arrow (gray) between the "Sign in" and "Incorrect login or password" pages, and a navigational arrow (blue) between the "Create a new account" button and the "New account" page. 53

**FIGURE 4-14** The pointer tool. 54

**FIGURE 4-15** The template pane. Clearing the check box next to the default template would remove it from the currently selected page, in this case, the Wine Country page. 55

**FIGURE 4-16** The toolbar buttons for *Run from Home Page* and *Run from Selected Page*. 56

**FIGURE 4-17** Run windows for desktop (left) and smartphone (right) designs. 57

**FIGURE 4-18** A voice UI design, including a form titled "Wine Country," orange computer prompts and green user responses. 58

**FIGURE 4-19** Inserting a prompt. 59

**FIGURE 4-20** Creating a response. 59

**FIGURE 4-21** A response with more than one possible phrase. 60

**FIGURE 4-22** The prompt *What is your favorite vegetable?* is the initial prompt in this form.60

**FIGURE 4-23** A universal which goes to the Home form whenever a user says “Start Over.” 61

**FIGURE 4-24** Run mode for voice user interfaces, with the same form in design mode in the background. 62

**FIGURE 4-25** The radio buttons used for changing the active layer. 64

**FIGURE 4-26** The same page with *All Devices* as the current layer (above) and *This Device* as the current layer (below). The phrase “any accounts in our file. Please try again” is in the *This Device* layer, and all of the other objects are in the *All Devices* layer. 65

**FIGURE 4-27** The "Move Object to This Device" button in the toolbar. 67

**FIGURE 4-28** A set of radio buttons and a button that links to another page, and the equivalent voice design. Note that the text of the radio buttons is the same as the text of the response, while the destination of the arrow from the OK button is the same as the destination of the response. 69

**FIGURE 4-29** The “synchronize text” indicator, circled on this prompt, is dimmed to indicate the prompt’s text is not to be synchronized with other devices. 72

**FIGURE 4-30** The pattern browser. 73

**FIGURE 4-31** The pattern browser with the Illustration and Background sections collapsed. 74

**FIGURE 4-32** The pattern browser showing solutions of the SIGN-IN/NEW ACCOUNT pattern for the desktop and smartphone, which can be added to a Damask design. 76

**FIGURE 4-33** An instance of the SIGN-IN/NEW ACCOUNT pattern in a UI design. 77

**FIGURE 5-1** The class hierarchy for `InteractionElement`. For implementation purposes, `InteractionElement` is actually an interface, but it has only one implementation, `AbstractInteractionElement`. 81

**FIGURE 5-2** The model (left) for a typical smartphone user interface design (right). 82

**FIGURE 5-3** A page that is on the All Devices layer (“Home”) and a page that is on the This Device layer in Desktop view (“Settings”), along with the associated model. 83

**FIGURE 5-4** A panel with a label and three radio buttons, and the backing model. Note how the group only points to controls; it does not contain them. 86

**FIGURE 5-5** A voice UI in Damask and its corresponding model. Empty page regions are omitted for brevity. 89

**FIGURE 5-6** Instantiating a pattern. 91

**FIGURE 5-7** The class hierarchy for `InteractionElementView`. 93

**FIGURE 5-8** The scenegraph (top left and bottom left) for a smartphone UI (top right) and its equivalent voice UI (bottom right). 94

**FIGURE 5-9** An example voice UI for generating a VoiceXML file. This is the same as Figure 5-5. 99

**FIGURE 5-10** The generated VoiceXML file for the user interface in Figure 5-9. 105

**FIGURE 6-1** The Fujitsu Tablet PC used in the experiments, in tablet mode (left) and laptop mode (right). 108

**FIGURE 6-2** Participant 6's desktop design for TotalMusic.com, created without patterns or layers. 122

**FIGURE 6-3** Participant 6's desktop design for TotalBooks.com, created with patterns and layers. 123

**FIGURE 6-4** A desktop Damask design (above) and its cleaned-up HTML version (below) used in the online evaluation. 128

**FIGURE 6-5** The key to the symbols used in the abstract sitemaps. 135

**FIGURE 6-6** The abstract sitemap for Participant 6's TotalMusic.com design for the desktop, shown in Figure 6-2, which was created without layers or patterns.Figure **6-2** Participant 6's desktop design for TotalMusic.com, created without patterns or layers. 135

**FIGURE 6-7** The abstract sitemap for Participant 6's TotalBooks.com design for the desktop, shown in Figure 6-3. 136

**FIGURE 6-8** The abstract sitemaps for Amazon.com (top) and BarnesAndNoble.com (bottom). 136

**FIGURE 6-9** Transforming the design shown in Participant 6's TotalMusic.com to the Amazon.com sitemap. 138

**FIGURE 6-10** Participant 7's smartphone design for TotalMusic.com, made without patterns or layers. 145

**FIGURE 6-11** Participant 7's smartphone design for TotalBooks.com, made with patterns and layers. 146

**FIGURE 6-12** A smartphone Damask design (left) and its cleaned-up HTML version (right) used in the online evaluation. 151

**FIGURE 6-13** Participant 15's voice design for TotalMusic.com, made without patterns or layers. 155

**FIGURE 6-14** Participant 15's voice design for TotalBooks.com, made with patterns and layers. 156

**FIGURE 8-1** Mockup of creating a solution for a new pattern. 192

**FIGURE 8-2** Mockup of an Examples section in the Pattern Browser. 193

**FIGURE 8-3** Mockup of a checklist for choosing features within a pattern solution. 195

**FIGURE 8-4** Mockup of annotations (in green) 197

**FIGURE 8-5** Mockup of an "All" tab in Damask. 199

**FIGURE 8-6** Mockup of device-specific tools within the lighter-colored toolbox in the Desktop tab. 200

**FIGURE 8-7** *Top* A tray for the Album page, containing desktop objects to add. *Bottom* The Album page with objects added from the tray. 201

# List of Tables

TABLE 2.1 The participants for our study, their companies, and the platforms they designed for. 19
TABLE 2.2 The job role, educational background, and design experience of our participants. 21
TABLE 4.1 Visual controls and their corresponding voice controls. 68
TABLE 4.2 Voice controls and their corresponding visual controls. 70
TABLE 4.3 The patterns for which Damask solutions were implemented. 75
TABLE 5.1 Views of visual controls and their corresponding model objects. 85
TABLE 5.2 Views of voice controls and their corresponding model objects. 87
TABLE 5.3 The model objects for pages and controls, along with their corresponding view objects. 95
TABLE 6.1 The details and task ordering of Phase 1 (desktop/smartphone) 111
TABLE 6.2 The details of and task ordering of Phase 2 (desktop/voice) 112
TABLE 6.3 Demographics of Desktop/Smartphone phase (Phase 1) participants 114
TABLE 6.4 Demographics of Desktop/Voice phase (Phase 2) participants 115
TABLE 6.5 Mean time spent by desktop/smartphone participants creating UI design, in hours and minutes (gray background signifies $p < 0.05$, 2-tailed t-test) 117
TABLE 6.6 Mean time spent by desktop/voice participants creating UI design, in hours and minutes ($p > 0.30$, 2-tailed t-test) 117
TABLE 6.7 Mean time spent by desktop/voice participants creating UI design, without Designer 16, in hours and minutes 118
TABLE 6.8 Completeness of designs (gray background signifies $p < 0.05$, 2-tailed t-test)119
TABLE 6.9 Completeness of designs among designers who used more than 1 pattern in the patterns condition (gray background signifies $p < 0.05$, 2-tailed t-test)119
TABLE 6.10 Demographics of the evaluators of the desktop designs. The demographics of the designers are included for comparison. 125

**TABLE 6.11** Average ratings for the desktop designs (1 = low, 5 = high), along with *F* and *p* values from ANOVAs. 130

**TABLE 6.12** Average ratings for the desktop designs (1 = low, 5 = high), excluding Designers 8 and 16, along with *F* and *p* values from ANOVAs. 131

**TABLE 6.13** Common features in the checkout process. 134

**TABLE 6.14** Results of MANOVAs on edit distances for desktop designs. 139

**TABLE 6.15** Average edit distances for the desktop designs, along with *F* and *p* values from ANOVAs (NP = no patterns, P = patterns). 140

**TABLE 6.16** Average edit distances for the desktop designs, without the outliers from Designers 8 and 16, along with *F* and *p* values from ANOVAs (NP = no patterns, P = patterns). 140

**TABLE 6.17** Average edit distances for the desktop designs, excluding optional features, along with *F* and *p* values from ANOVAs (NP = no patterns, P = patterns).143

**TABLE 6.18** Average edit distances for the desktop designs, excluding optional features, without the outliers from Designers 8 and 16, along with *F* and *p* values from ANOVAs (NP = no patterns, P = patterns). 143

**TABLE 6.19** Demographics of the evaluators of the smartphone designs. The demographics of the smartphone designers (Designers 4–11) are included for comparison. 148

**TABLE 6.20** Average ratings for the smartphone designs (1 = low, 5 = high), along with *F* and *p* values from ANOVAs. 152

**TABLE 6.21** Average ratings for the smartphone designs (1 = low, 5 = high), excluding Designer 8, along with *F* and *p* values from ANOVAs. 153

**TABLE 6.22** Average ratings of all eight voice designs (four non-patterns and four patterns) and the voice designs except those from Designer 16, who only used one pattern in the Patterns condition (1 = did not like, 7 = liked very much). 157

**TABLE 6.23** Patterns used by the participants 159

**TABLE 6.24** Participants' ratings on the usefulness of the patterns for the TotalBooks.com design task (1 = not useful, 7 = very useful) 161

**TABLE 6.25** Participants' ratings on the general usefulness of the patterns (1 = not useful, 7 = very useful) 162

**TABLE 6.26** Patterns that participants tried to use but had no Damask solutions. The highlighted columns indicate the patterns most frequently tried. 163

**TABLE 6.27** Participants' ratings on how easy to understand layers are (1 = not easy, 7 = very easy). 165

**TABLE 6.28** Participants' ratings on how much they like layers (1 = do not like, 7 = like very much). 165

## Acknowledgments

No one, including me, could have written a dissertation like this without the help, support, and encouragement of a veritable army of colleagues, family, and friends.

First of all, thanks to all of those who participated in my interviews and usability studies (preserving anonymity prevents me for naming them personally): the nine user interface designers whom I interviewed at the beginning of this project, the six user interface designers who used HopiSketch, the seventeen user interface designers who used Damask, and the thirty-four user interface designers who *evaluated* the designs made with Damask by the previous group.

Thanks to the undergrads that worked with me on Damask: Wei (Michelle) Xue and Qing Li on the coding, and Madhu Prabaker on the evaluation. DENIM was an important predecessor to Damask, so I would also like to extend my thanks to the researchers who worked with Jason Hong, Mark Newman, and me on that project—Yang Li, Marc Ringuette, and Michael Thomsen—as well as the throngs of undergrads: Liane Beckman, Michael Bina, Lisa Chan, Eric Chung, Andrew Cuneo, Carol Hu, Peter Khooshabeh, John Brian Kirby, Jenny Lee, Robert Lee, William Lee, Boxin Li, Benson Limketkai, Nahush Mahajan, Michael Pow, Lifeng Shelley Shen, Quinn Solomon, Orna Tarshish, Eric Tse, and Juan Valencia.

The design patterns in the book *The Design of Sites* were a crucial part of Damask. Thanks to Doug van Duyne who, along with James Landay and Jason Hong, cataloged the patterns and wrote the book.

The PIMA and Hopi projects at IBM Watson Research Center scared me at first, because I thought I had been scooped. But once it was clear that hadn't quite happened, I realized what a great opportunity it would be to work with those researchers. Thanks to Lawrence

Bergman, Danny Soroker, Richard Cardone, and Guruduth Banavar for their guidance on HopiSketch and helping me run the user study; John Karat, Noi Sukaviriya, and Tracee Wolf for helping us with the design of our study and questionnaire; Pauline Ores and Kate Swann for helping us recruit participants for our user study; and Frederique Giraud, Ashish Kundu, Yves Gaeremynck, and Vianney Chevalier for their contributions to the implementation of HopiSketch.

As a member of the HCI research community, you get to know some wonderful people. Thanks to Beverly Harrison, Mave Houston, Liz Goodman, and Wendy Ju for helping me with Damask's evaluation (Wendy, I still owe you a bowl of ramen).

My colleagues at IBM Almaden Research Center have shown remarkable support and encouragement in my quest to finish my dissertation, including Alex Cozzi, Clemens Drews, Steve Farrell, Tessa Lau, Barton Smith, Steve Cousins, and Tom Moran. A big thanks to those who helped me with the evaluation of Damask: Allen Cypher, Alison Sue, John Tang, and Shumin Zhai.

Being part of the Group for User Interface Research at Berkeley has been a great experience. Thanks to Jason Hong, my officemate of seven years whose conversations about design patterns helped spark Damask, and co-discoverer of the Hong-Lin Line dividing sweet-tea and non-sweet-tea country in the U.S. (as of this writing, it's too bad we don't know where the line actually is). Thanks to those GUIR members and alumni who directly contributed to Damask and its related projects: Jonathan Huang, Francis Li, Mark Newman, Anoop Sinha, Leila Takayama, and Scott Klemmer (I also thank Scott for letting me swipe his Microsoft Word dissertation template). Also thanks to the rest of the past and present of GUIR, especially Chris Beckmann, Richard Davis (who was also my CSGSA Big Brother when I first entered grad school), Kate Everitt, Jeff Heer, Xiaodong Jiang, Hesham Kamel, Alan Newberger, Wai-ling So (née Ho-Ching), Sarah Waterson, and the other two members of the Great Scott Troika, Scott Carter and Scott Lederer. It has been a privilege, and a lot of fun, working with all of you. I look forward to seeing you again at conferences, seminars, and reunions.

Of course, life at Berkeley wasn't just about GUIR. Andy Begel and David Oppenheimer joined me in wondering whether we would ever graduate from Berkeley—we made it! Rich Vuduc and Ben Horowitz accompanied me on all sorts of wacky adventures, like the Silicon Valley Tour (Classic, .com, and Redux Editions), Rail Around the Bay, and a tour of the Jelly Belly Factory. And I can't forget my fellow Caltech undergrad/Berkeley grad friends, including Vito Dai (who also went to my high school!), Frank Ling, Michael Ru, and Chinh Doan, my roommate for 7½ years.

Thanks to the members of my quals and dissertation committees, Jennifer Mankoff, Marti Hearst, John Canny, and Warren Sack, both for their research insights and in helping me navigate the Ph.D. process. My advisor, James Landay, was instrumental in shaping my research career. He showed me by example how to be a successful researcher, he was not afraid to challenge my half-baked ideas in private, but he always defended and promoted my research in public. Thank you very much.

And finally, thanks to Mom and Dad, who saw the potential of computers for me at an early age, and helped plant the crazy idea in my head to pursue a Ph.D. in the first place. I appreciate your love, support, and encouragement.

This research was funded by in part by Hewlett-Packard, a summer internship at IBM Watson Research Center, and the National Science Foundation under Grant No. 9985111.

# 1 Introduction

"Using a computer" is no longer limited to using a personal computer. Interacting with a PC in a home or office is now augmented with a variety of devices, such as handheld personal digital assistants (PDAs), mobile phones, pagers, and even telematics systems in cars. Companies as varied as Amazon [1], the BBC [28], and Yahoo [32] are starting to allow their customers to access their services through such a variety of devices. For example, you can find out which theaters are playing a particular movie and at what time through a voice-based phone interface, a PDA web site, or a desktop web site. However, due to the attributes and limitations of each device, the interfaces across devices are often drastically different. This makes the task of designing a user interface (UI) for a service that targets several devices difficult, because you essentially need a distinct UI for each device.

If UI designers want to target several devices for an application, they generally face two alternatives. One option is to design a user interface for each targeted device. This process results in interfaces that are optimized for each device, but it has several drawbacks. Designing several user interfaces is very time consuming, and the more devices the designer targets, the more time and effort the designer must spend. It is also hard for designers to keep the designs coordinated across devices. A designer could add a feature to one device-specific UI, and then easily forget to at least investigate the possibility of adding that feature to another device-specific UI. Also, a different person often designs each device-specific UI, exacerbating this problem.

The other option is to design an interface for only one device and let special-purpose programs automatically generate the interfaces for other devices. This cuts down development time but leads to interfaces that are awkward to use. Consequently, they are only used as a last resort by end-users who have no other way to access the information or perform the task provided by that UI.

The difficulty of designing for multiple devices discourages designers from iteratively refining and prototyping their designs. One of the best ways to create a good user interface is to continually design, test, and analyze a user interface idea [60, 103]. If creating a design in the first place is difficult, designers will not want to try multiple designs or drastically change their initial design, which may impact the quality of the final design. Tools that make early-stage design, prototyping, and testing of cross-device user interfaces easier could dramatically improve the usability and usefulness of those interfaces.

There is a way that allows designers to design and prototype cross-device UIs that are appropriate for each device, yet takes much less time than designing each design-specific UI separately. Specifically, my thesis is:

> **A tool that uses design patterns to bridge the gap between device-specific UIs will enable designers to create cross-device UIs with at least the same functionality as if the designer designed each device-specific UI separately, but in less time.**

To show this, we have created a tool called Damask aimed at designers who want to design and prototype a UI targeted at three types of interfaces: the web accessed through a desktop, mobile phone displays, and prompt-and-response style voice interfaces (see Figure 1-1). We have picked these three because they represent the "extremes" of the range of devices that are widely used. For example, simply shrinking a screen designed for a desktop PC will not result in a good mobile phone interface.

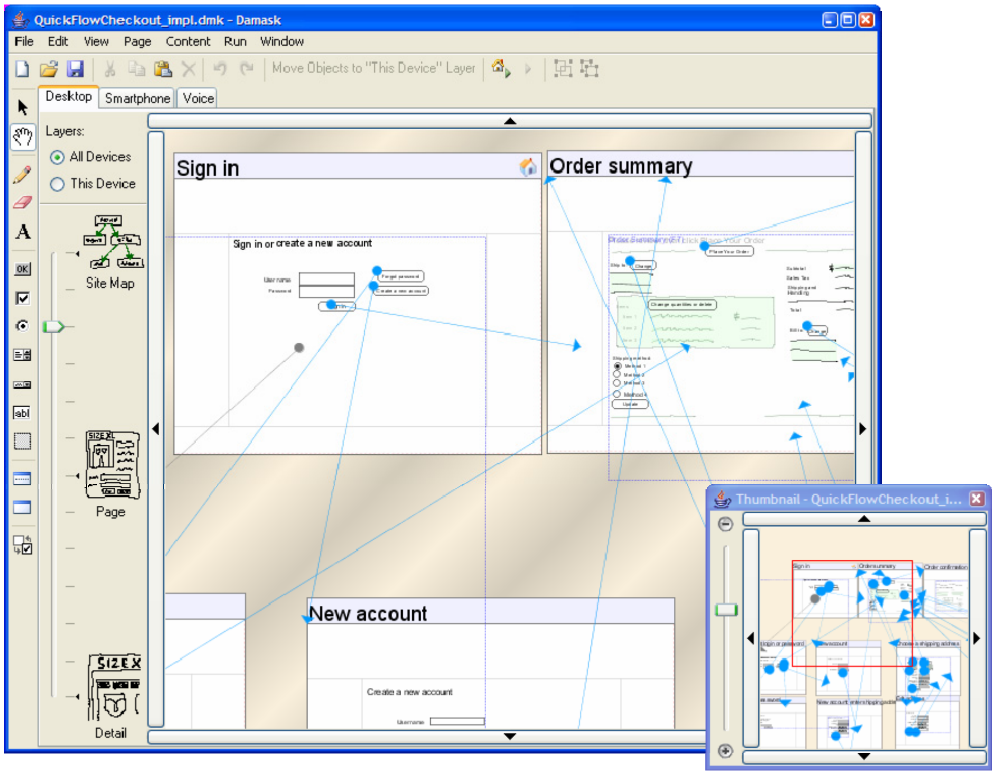


**FIGURE 1-1** Damask's user interface

With Damask, the designer designs a user interface for one device, by sketching the design and by specifying which *design patterns* [4, 188] the interface uses. As the designer sketches an interface, Damask constructs an *abstract model* [49], which captures aspects of the UI design at a high level of abstraction. Damask uses the abstract model and the patterns to generate the other device-specific interfaces in real time, which the designer can refine if he or she wishes. The generated interfaces are good enough so that it is more convenient to use the tool than to design each of the other interfaces separately.

Damask also provides a Run mode in which designers interact with their design sketches in a browser that roughly simulates the devices they are targeting. This allows designers to get quick feedback about their design from other team members or even their target users, which will inform any modifications they want to make to their design.

## 1.1 Creating Cross-device Interfaces

Here is an example of a designer using Damask to design a UI, for example, an e-commerce web site for the PC, mobile phone, and voice. The designer decides to first target the PC, so he sketches out some web pages for the PC version of the web site in the main canvas area of Damask. As he is sketching his site, Damask creates mobile phone and voice versions of the site. Figure 1-2 shows one such page, along with the smartphone and voice versions.

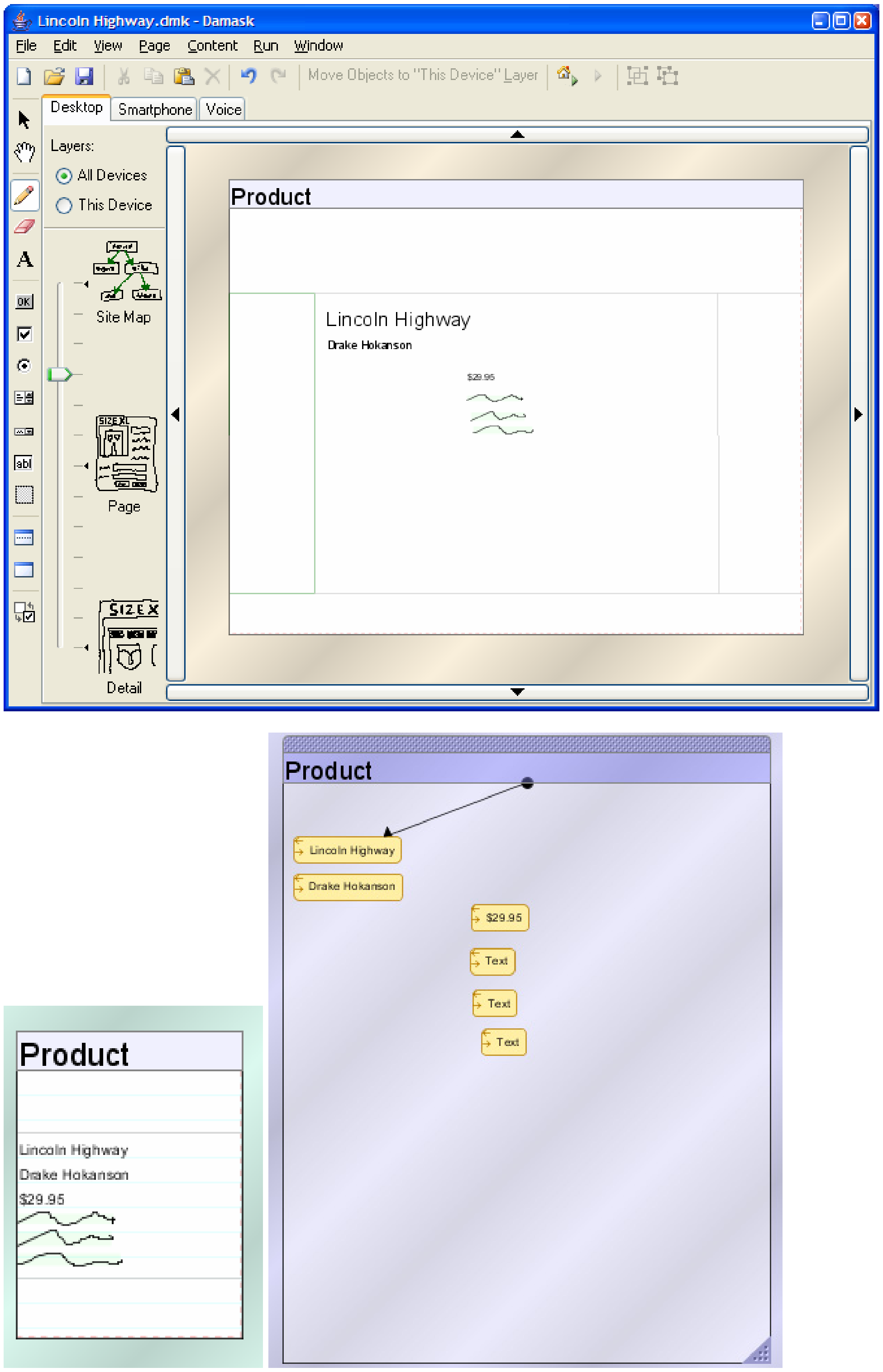


**FIGURE 1-2** *Top* A desktop web page sketched by a designer in Damask. *Bottom* The smartphone (left) and voice (right) versions of that page generated by Damask.

When the designer creates, edits, or deletes an object in the *Desktop* view, the same action is performed in the *Smartphone* and *Voice* views. However, moving or resizing an object in one view does not affect the other views, since the displays of mobile phones are so limited that the designer would most likely have to move the object again.

While he is sketching, he knows that he does not want the product image and the "Related Products" section of the desktop page to also be in the other two versions. So he switches from the *All Devices* layer to the *This Device* layer, and then sketches those items (see Figure 1-3).

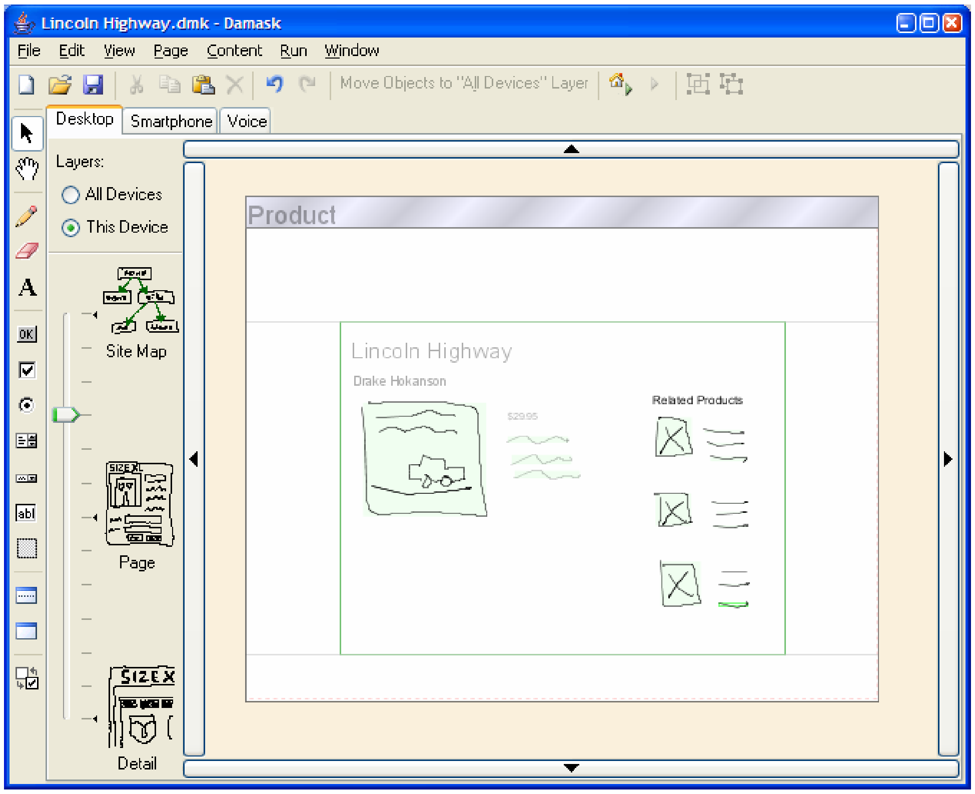


**FIGURE 1-3** Adding objects to the *This Device* layer in Desktop view, which does not affect the Smartphone and Voice views. The objects previously drawn in the *All Devices* layer are grayed out, since the *This Device* layer is the current layer.

The designer also goes to the *Voice* view, switches to the *This Device* layer, and adds connecting phrases between the disconnected bits of text to make a smooth sentence. He also changes the prompts labeled "Text," which were generated from the squiggly lines he drew in *Desktop* view, into actual text (see Figure 1-4). This allows the designer to create an interface that is tailored to voice while still synchronized with the other devices. For example, if the designer changed the name of the product in *Smartphone* view, it would also be changed in *Voice* view.

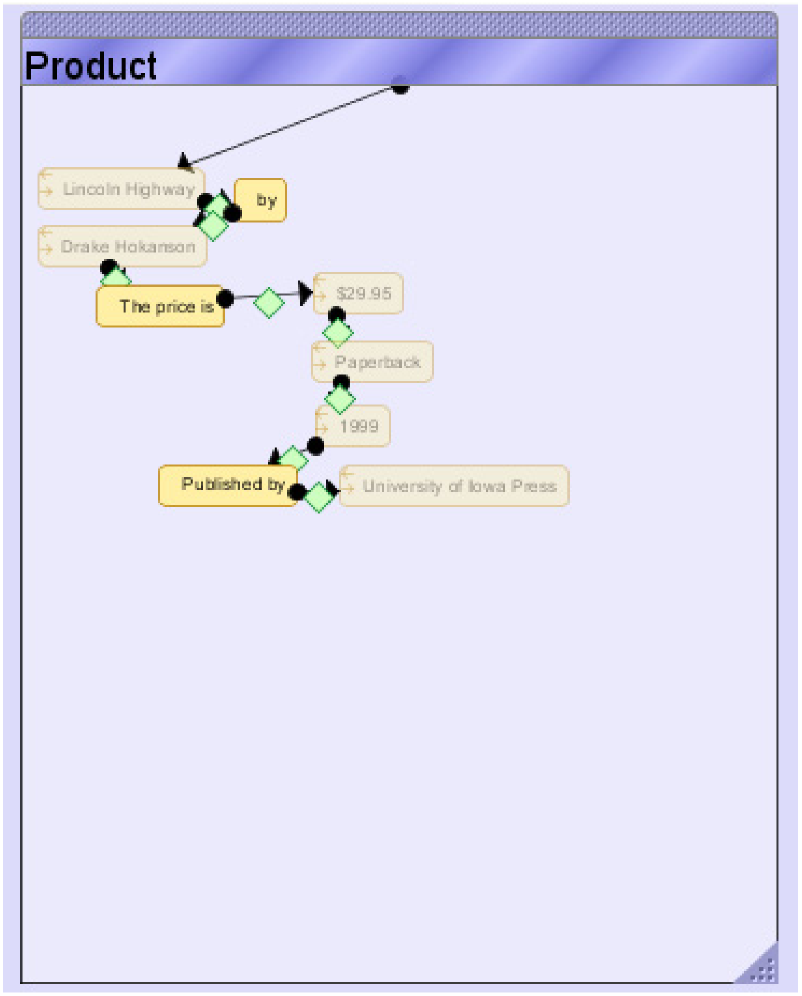
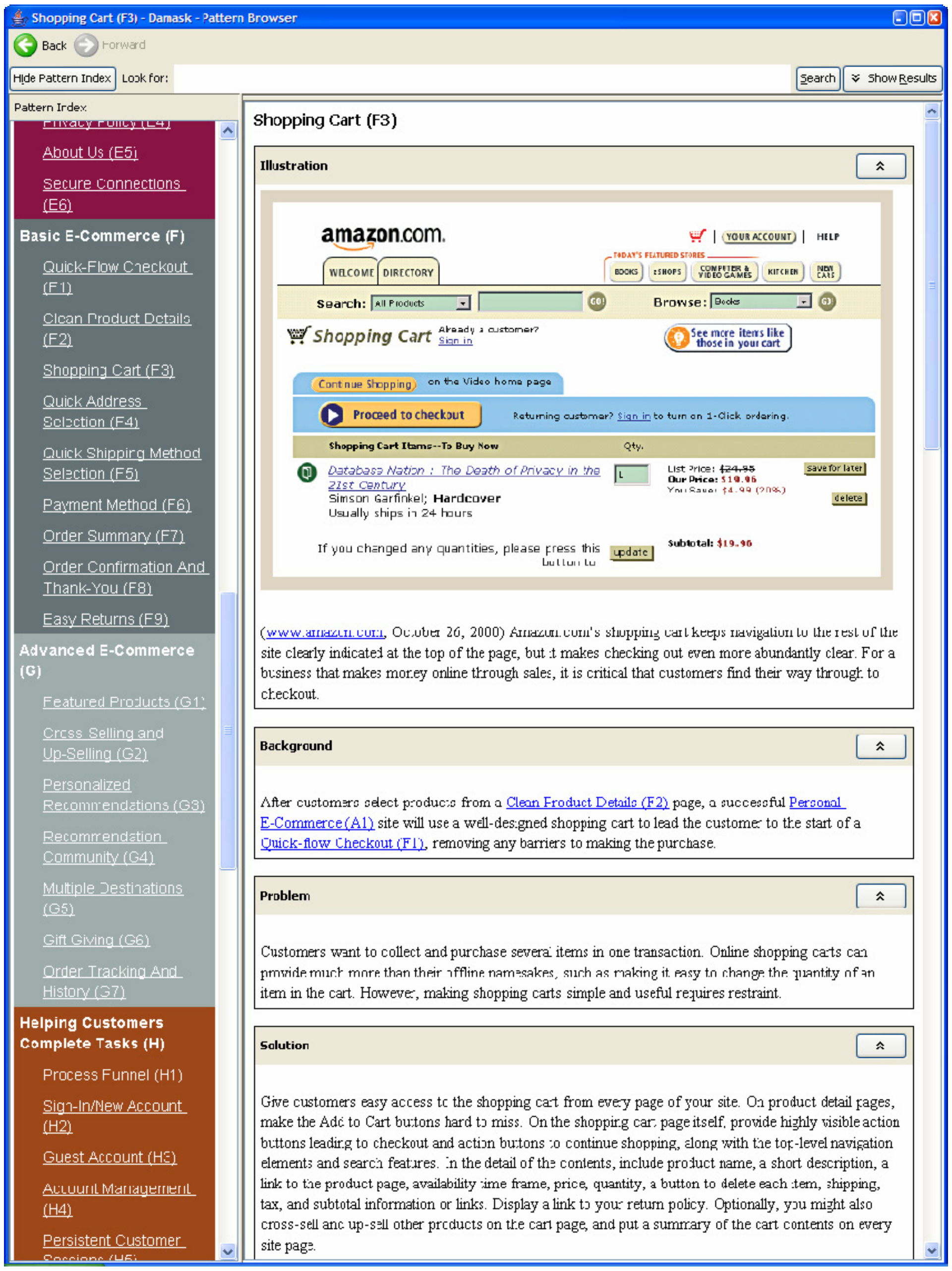


**FIGURE 1-4** Smoothing out the voice interface.

Later, the designer decides to add a shopping cart to the site. Instead of sketching it out from scratch, the designer takes advantage of the patterns built into Damask. He brings up the Pattern Explorer to browse through the patterns, and comes across the SHOPPING CART pattern (see Figure 1-5).


Shopping Cart (F3) - Damask - Pattern Browser
Back
Forward
Hide Pattern Index
Look for:
Search
Show Results
Pattern Index
amazon.com.
YOUR ACCOUNT
HELP
TODAY'S FEATURED STORES
WELCOME
DIRECTORY
BOOKS
zSHOPS
COMPUTER & VIDEO GAMES
KITCHEN
NEW CARS
Search: All Products GO!
Browse: Books GO!
Shopping Cart
Already a customer? Sign in
See more items like those in your cart
Continue Shopping on the Video home page
Proceed to checkout
Returning customer? Sign in to turn on 1-Click ordering.
Shopping Cart Items--To Buy Now
Qty.
Database Nation : The Death of Privacy in the 21st Century
Simson Garfinkel; Hardcover
Usually ships in 24 hours
List Price: ~~$24.95~~
Our Price: $19.96
You Save: $4.99 (20%)
save for later
delete
If you changed any quantities, please press this button to
update
Subtotal: $19.96

Privacy Policy (E4)

About Us (E5)

Secure Connections (E6)

**Basic E-Commerce (F)**

Quick-Flow Checkout (F1)

Clean Product Details (F2)

Shopping Cart (F3)

Quick Address Selection (F4)

Quick Shipping Method Selection (F5)

Payment Method (F6)

Order Summary (F7)

Order Confirmation And Thank-You (F8)

Easy Returns (F9)

**Advanced E-Commerce (G)**

Featured Products (G1)

Cross Selling and Up-Selling (G2)

Personalized Recommendations (G3)

Recommendation Community (G4)

Multiple Destinations (G5)

Gift Giving (G6)

Order Tracking And History (G7)

**Helping Customers Complete Tasks (H)**

Process Funnel (H1)

Sign-In/New Account (H2)

Guest Account (H3)

Account Management (H4)

Persistent Customer Sessions (H5)

Shopping Cart (F3)

**Illustration**

(www.amazon.com, October 26, 2000) Amazon.com's shopping cart keeps navigation to the rest of the site clearly indicated at the top of the page, but it makes checking out even more abundantly clear. For a business that makes money online through sales, it is critical that customers find their way through to checkout.

**Background**

After customers select products from a Clean Product Details (F2) page, a successful Personal E-Commerce (A1) site will use a well-designed shopping cart to lead the customer to the start of a Quick-flow Checkout (F1), removing any barriers to making the purchase.

**Problem**

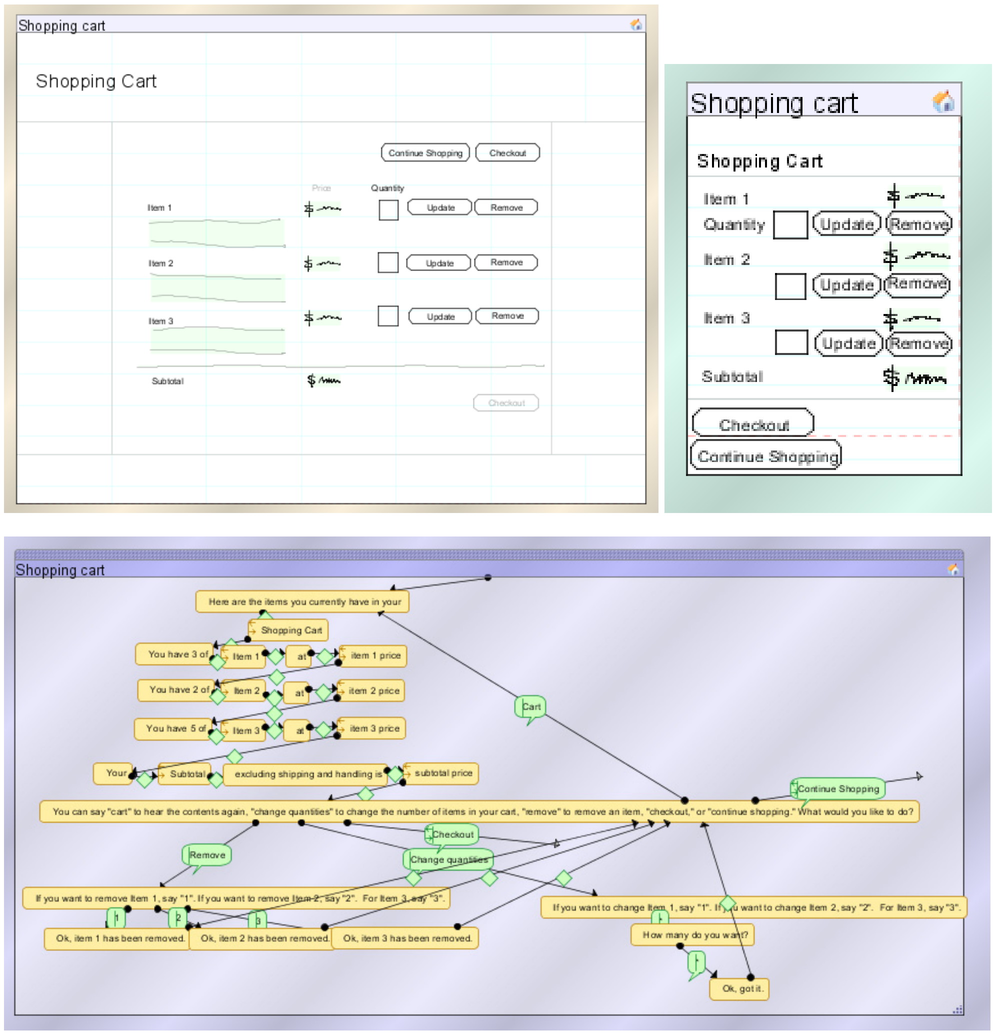
Customers want to collect and purchase several items in one transaction. Online shopping carts can provide much more than their offline namesakes, such as making it easy to change the quantity of an item in the cart. However, making shopping carts simple and useful requires restraint.

**Solution**

Give customers easy access to the shopping cart from every page of your site. On product detail pages, make the Add to Cart buttons hard to miss. On the shopping cart page itself, provide highly visible action buttons leading to checkout and action buttons to continue shopping, along with the top-level navigation elements and search features. In the detail of the contents, include product name, a short description, a link to the product page, availability time frame, price, quantity, a button to delete each item, shipping, tax, and subtotal information or links. Display a link to your return policy. Optionally, you might also cross-sell and up-sell other products on the cart page, and put a summary of the cart contents on every site page.

**FIGURE 1-5** The Pattern Explorer with the SHOPPING CART pattern selected.

He sees that there are three generalized solutions for SHOPPING CART—for desktop, smartphone, and voice (see Figure 1-6).



Here are the items you currently have in your

Shopping Cart

You have 3 of | Item 1 | at | item 1 price

You have 2 of | Item 2 | at | item 2 price

You have 5 of | Item 3 | at | item 3 price

Your | Subtotal | excluding shipping and handling is | subtotal price

You can say "cart" to hear the contents again, "change quantities" to change the number of items in your cart, "remove" to remove an item, "checkout," or "continue shopping." What would you like to do?

If you want to remove Item 1, say "1". If you want to remove Item 2, say "2". For Item 3, say "3".

Ok, item 1 has been removed.

Ok, item 2 has been removed.

Ok, item 3 has been removed.

If you want to change Item 1, say "1". If you want to change Item 2, say "2". For Item 3, say "3".

How many do you want?

Ok, got it.

**FIGURE 1-6** The generalized solutions for SHOPPING CART. *Top* The desktop and smartphone versions. *Bottom* The voice version.

The designer drags the solution of the SHOPPING CART into the canvas. This adds an *instance* of the pattern to the desktop, mobile phone, and voice versions of the designer's design (see Figure 1-7). The instance is surrounded by a light blue dotted box and the name of the instance's pattern.

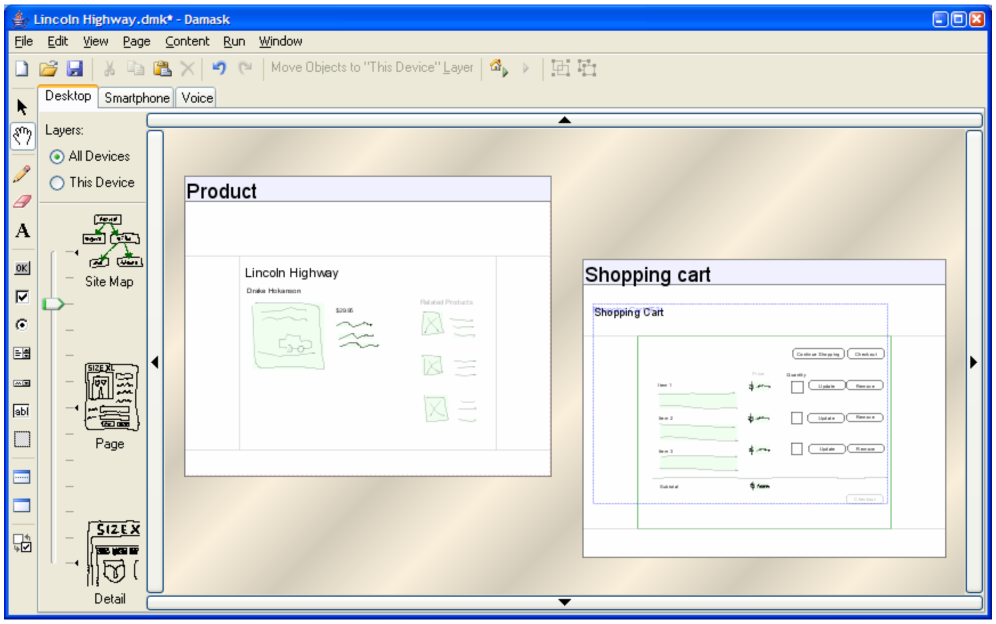


FIGURE 1-7 The shopping cart added to the main design.

The pattern that the designer has just instantiated is very generic, for example, having mostly text placeholders instead of actual text. The designer now customizes the pattern instance to fit his own project. He replaces the text placeholders with actual text, moves widgets around, and adds his own images. He could even add pages and arrows if he decides that is appropriate (see Figure 1-8).

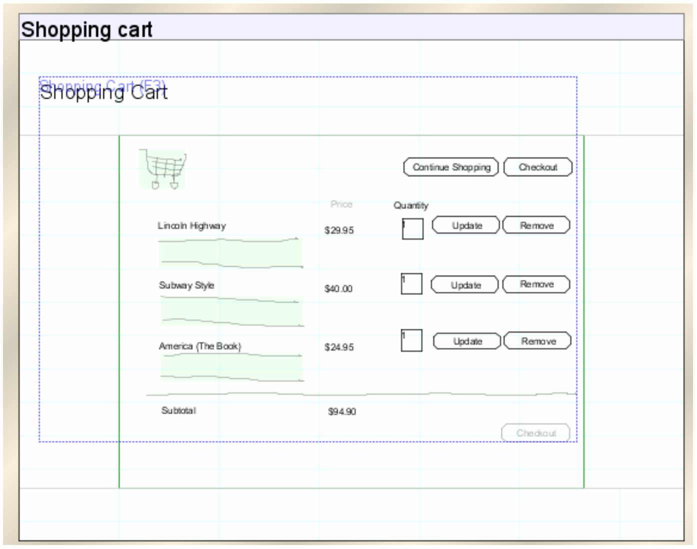
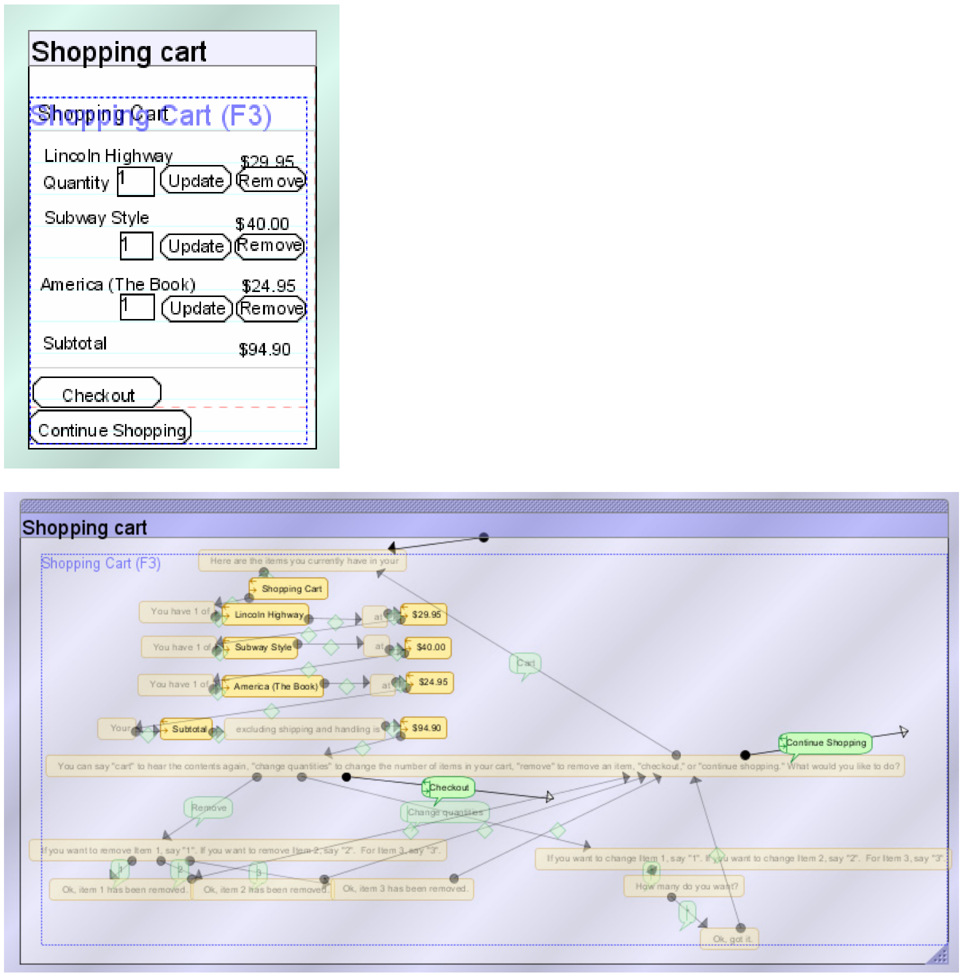


**FIGURE 1-8** The desktop version of the shopping cart, with the contents, prices, and quantities changed by the designer.

As the designer edits the pattern instance, Damask applies some of the edits to the designs for the other devices, according the same rules as when the designer edited his own design: adding, removing, and editing an object in the All Devices layer apply across all devices, while moving and resizing it do not (see Figure 1-9).



Here are the items you currently have in your

You have 1 of Lincoln Highway at $29.95

You have 1 of Subway Style at $40.00

You have 1 of America (The Book) at $24.95

Your Subtotal excluding shipping and handling is $94.90

You can say "cart" to hear the contents again, "change quantities" to change the number of items in your cart, "remove" to remove an item, "checkout," or "continue shopping." What would you like to do?

If you want to remove Item 1, say "1". If you want to remove Item 2, say "2". For Item 3, say "3".

Ok, item 1 has been removed.

Ok, item 2 has been removed.

Ok, item 3 has been removed.

If you want to change Item 1, say "1". If you want to change Item 2, say "2". For Item 3, say "3".

How many do you want?

Ok, got it.

**FIGURE 1-9** The smartphone (top) and voice (bottom) versions of the shopping cart, customized by Damask. Note that the contents of the cart have changed.

The designer now adds an Add to Cart button to the Product page in Desktop view and links the button to the Shopping Cart page. This creates and links a button in the Smartphone view (see Figure 1-10).

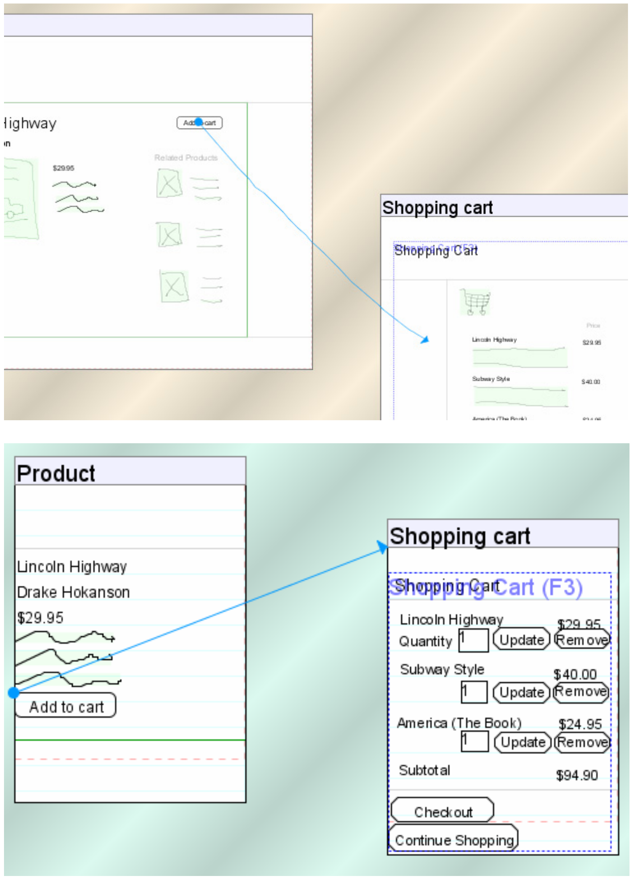


**FIGURE 1-10** *Top* A designer adding an "Add to Cart" button with a link to the "Shopping Cart" page in *Desktop* view. *Bottom* An "Add to Cart" button linked to the Shopping Cart page in *Smartphone* view that was automatically created by Damask.

This also results in a voice response labeled "Add to Cart" in the *Voice* view, pointing to the first prompt in the "Shopping Cart" form. The response represents the end-user saying "Add to Cart," which results in the "Shopping Cart" form being read to the user. However, the source of the response is not quite right; as designed, the voice interface will pause after saying "University of Iowa Press," waiting for the user to say, "Add to Cart" (see Figure 1-11).

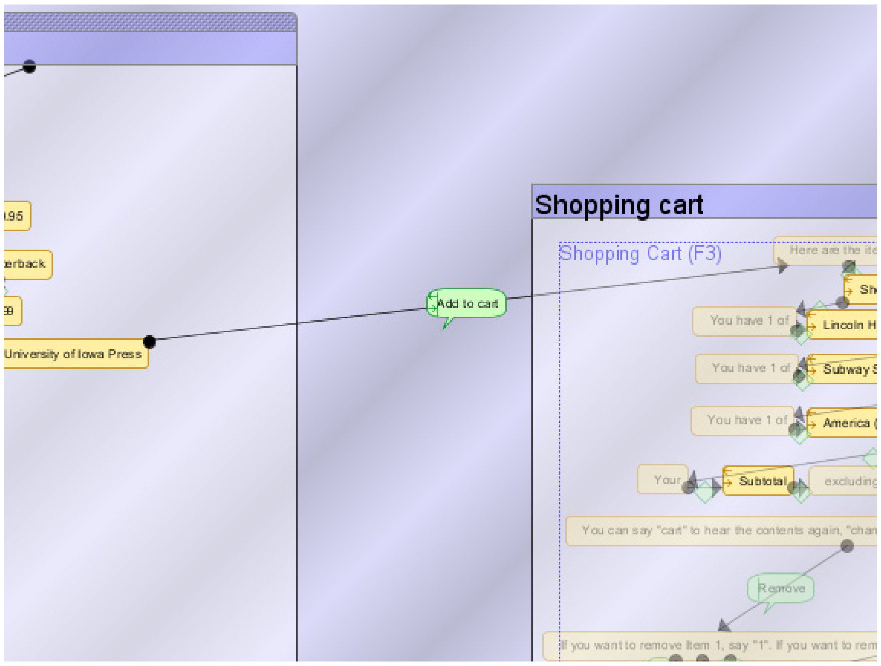


**FIGURE 1-11** An Add to Cart response linked to the Shopping Cart form that was automatically created by Damask.

To fix this, the designer adds a prompt in the This Device layer that says, “What would you like to do now?” A better prompt would include what choices the user could make at this stage, but since the designer has not yet decided what the choices will be, he decides this is good enough for now, and will refine it later. Then the designer re-anchors the response so that it originates from the new prompt (see Figure 1-12).

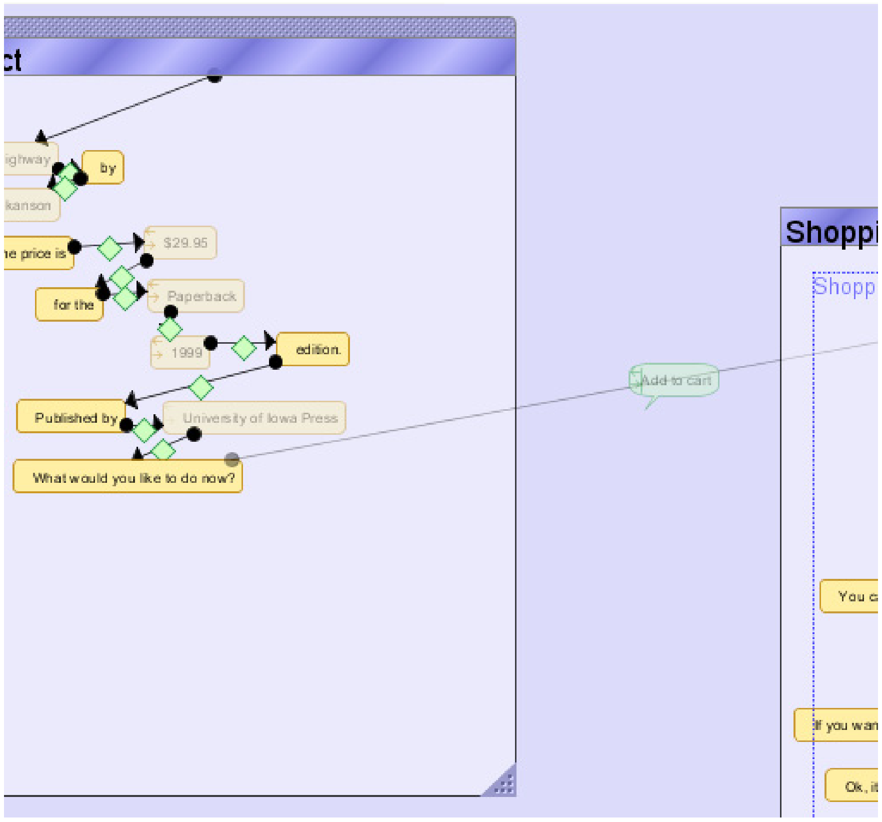


**FIGURE 1-12** The Add to Cart response after it was fixed up by the designer.

## 1.2 Dissertation Contributions

This dissertation makes the following research contributions:

- An understanding of current work practices for cross-device user interface design, achieved through field interviews and surveys
- The novel application of the following techniques to cross-device user interface design:
  1. *Design patterns* allow designers to describe their designs at a high level of abstraction and, by capturing interaction semantics, make it easier for a design tool to create interfaces appropriate for different devices.

2. *Layers* can provide a clean conceptual model for designers to keep track of which UI elements are consistent across devices and which are device-specific.

- A data model to represent cross-device UIs that incorporates design patterns and layers and links corresponding elements across devices
- A user interface design tool called Damask that improves the design of cross-device UIs by implementing the above concepts
- An understanding of how cross-device user interface design is influenced by design patterns and layers, achieved through laboratory studies

Damask uses concepts from the areas of design patterns and model-based user interfaces, which we describe in more detail below.

## 1.3 Design Patterns

*Patterns* were first introduced by Christopher Alexander and his colleagues in the field of architecture. He states, "Each pattern describes a problem which occurs over and over again in our environment, and then describes the core of the solution to that problem, in such a way that you can use this solution a million times over, without ever doing it the same way twice." [4] This basic definition has become popular in the software engineering (e.g., [56]) and human-computer interaction (HCI) fields [27, 182, 188, 189].

We believe that there are patterns in user interfaces for multiple devices, and that the structure of these pattern solutions can be dramatically different, depending on the devices' characteristics. For example, the desktop version of an ADD TO SHOPPING CART pattern could show the new contents of the shopping cart or show other products that the shopper is interested in. On the other hand, the voice version of the pattern could simply say, "OK, it has been added," to avoid overwhelming the shopper with information.

Since patterns describe interactions at a higher level than widgets, a tool that supports patterns can generate device-specific interfaces that are better optimized than a simple widget-by-widget transformation that many research systems do today. Also, simply documenting these patterns may help designers think more clearly about UIs on multiple devices, since they could see how the interfaces relate to each other using patterns as a vocabulary.

## 1.4 Model-Based User Interfaces

Damask's underlying representation of UI designs and patterns is based on the concept of *model-based user interfaces* [178]. Model-based UI research has been going on for about two decades, and its basic premise is the idea of designing user interfaces based not just on visual appearance but also on an abstract model of the interface. The model describes the interface at a higher level of abstraction than the actual widgets. For example, instead of describing a dialog box as having three radio buttons and two check boxes, an abstract model would describe it has having one part where the user can select one of three items, and two other on-off selections. This level of abstraction allows the possibility of rendering the user interface in other ways, such as using a drop-down list or presenting a voice menu instead of radio buttons. Using patterns for describing interfaces further increases the level of abstraction and allows even bigger differences in interfaces across devices.

While model-based user interfaces have the promise of creating flexible interfaces that can adapt to their environment, they have not been widely adopted in the commercial software development world, which has instead gravitated towards visual interface builders. We believe one reason for the lack of acceptance is the fact that many model-based UI tools do not match or augment the work practices of designers. They often force designers to think at a high level of abstraction too early in the design process. Designers are accustomed to thinking about concrete interfaces at the beginning of this process. In addition, specifying models often requires the designer to deal with preconditions, postconditions, and conditionals, which starts to look like programming. Most designers are not skilled at programming, so specifying models impedes their main task of designing UIs.

Damask's approach allows UI designers to specify their designs at a more abstract level, i.e., create an abstract model for the interface, but with a vocabulary that designers understand, via sketches and design patterns. Design patterns also enable Damask to generate interfaces that are more appropriate for the targeted device.

## 1.5 Outline

The rest of the dissertation is organized as follows. We describe our interviews with cross-device UI designers in Chapter 2 and our experience with an early prototype of a cross-device UI tool in Chapter 3. In Chapter 4, we describe the user interface of Damask, including how a designer will use it to design cross-device interfaces, and we describe how we implemented Damask in Chapter 5. The evaluation of Damask and its results are in Chapter 6. In Chapter 7, we first discuss two of the main concepts that Damask embodies, design patterns and model-based user interfaces, and related work in more detail. We discuss future work in Chapter 8, and conclude in Chapter 9.

# 2 Study of Current Practices in Cross-Device UI Design

To get a better understanding of how designers currently design cross-device UIs, we interviewed nine UI designers across eight companies who worked on cross-device UI projects. The seven men and two women were recruited through electronic mailing lists for UI designers. We interviewed six of the designers in their offices, two over the phone, and one by e-mail. All of these projects targeted desktop PCs and mobile phones, and all but one also targeted PDAs. None of them targeted voice. Table 2.1 lists the type of companies for which our participants worked and the devices for which they designed.

**TABLE 2.1** The participants for our study, their companies, and the platforms they designed for.

| Designer | Type of company | Desktop | Palm | Pocket PC | WAP phone | Hi-end phone |
|---|---|---|---|---|---|---|
| A1 | Web portal | ✓ | ✓ | ✓ | ✓ | |
| A2 | Enterprise software | ✓ | ✓ | ✓ | ✓ | |
| A3 | Mobile access to corporate data | ✓ | ✓ | ✓ | ✓ | |
| A4 | Corporate portal | ✓ | | ✓ | ✓ | |
| A5 | UI design firm | ✓ | ✓ | ✓ | ✓ | |
| A6 | | ✓ | ✓ | ✓ | ✓ | |
| A7 | Startup incubator | ✓ | | ✓ | | ✓ |
| A8 | Mobile phone carrier | | | | | |
| A9 | Mobile phone carrier | ✓ | ✓ | | ✓ | |

We focused our questions on how the designers addressed the issue of handling multiple devices (see Appendix A). We wanted to know whether their companies grouped designers by device (e.g., PDA vs. phone designers) or by application (e.g., e-mail vs. calendar). We asked how they maintained consistency across designs, whether the desktop and mobile versions were developed at the same time, whether the same team worked on both versions, and if not, whether the two teams discussed their designs with each other. We also asked them whether they observed recurring interaction design patterns [188] in their designs and whether they documented them.

Finally, we discussed our ideas for a cross-device UI design tool and asked them for their reactions and to speculate on how useful such a tool would be. While it is hard for someone to predict how they would use a tool, the questions were designed to learn more about the designers' concerns about such a tool, not to learn about specific features.

We will discuss our findings along the following themes: responsibilities of the designers; scope of cross-device projects; organization of project teams; managing UI consistency; tools and documentation, and patterns; and the need for real-time change across devices.

## 2.1 Roles and Responsibilities of the Designers

All of the designers were responsible for overall information and interaction design, and some also handled graphic design [140]. None of them were developers. Seven of the designers did detailed UI design work. The others, Designers A2 and A9, guided the people doing the detailed design work and made sure they followed good usability principles and adhered to the companies' mobile UI style guides. Table 2.2 lists the participants along with their job roles, educational background, and design experience.

**TABLE 2.2** The job role, educational background, and design experience of our participants.

| Designer | Job role | Educational background | | Years of UI design experience |
|---|---|---|---|---|
| A1 | UI designer | BFA | Industrial design | 6 |
| A2 | Usability engineer | PhD | Human factors | 3 |
| A3 | UI designer | BS | Kinesiology | 4 |
| A4 | Info. architect/ UI designer | BS | Cognitive science | 4 |
| A5 | Senior info. designer | BS | Graphic design | >5 |
| A6 | Human sciences director/senior interaction designer | BS | Computer science | >5 |
| A7 | VP R&D | PhD | HCI | >5 |
| A8 | HCI team member | BS | Physics | 1½ |
| A9 | User experience design manager | BA | English and fine arts | 6 |

## 2.2 Scope of Cross-Device Projects

For most of the cross-device projects, the mobile UI offered a subset of the desktop UI's functionality. For Designers A5 and A6, mobile access and desktop access were thought of as two aspects of their projects as a whole; neither was considered a subset or superset of the other. Designer A9's projects were focused on phone interaction; the desktop was used mostly for managing aspects of the mobile experience, like storing pictures that the user took with the phone's digital camera.

## 2.3 Organization of Project Teams

At all but one of the companies, there were at most three UI designers in charge of the UI design for a project. At the UI design firm, a project typically had two to six designers.

For six of the designers, the cross-device projects were targeted at multiple devices from the beginning. The designers worked on both the desktop and mobile versions at the same time.

Designers A1, A2, and A4 only worked on the mobile UIs. These applications were originally written for the desktop and were later ported to mobile devices. These designers did not consult the desktop UI designers or their design documents; they simply looked at the desktop UI directly.

When asked how the tool should support multiple designers, the designers did not suggest any elaborate features. Designer A9 said that he never saw other designers actually use collaboration features in other tools and stressed that the overall learning curve of a new tool has to be low for a designer to consider using it.

We were particularly interested in finding out how a team of designers typically split up responsibility for designing a cross-device UI project. There are several possibilities:

1 *Device added later:* A UI for a device is designed long after the UI for another device is done
2 *Group by feature:* One designer designs a large part of the UI for all devices, at the same time other designers work on other parts of the UI
3 *Group by device:* One designer designs the UI for only one device, at the same time other designers work on other devices

The process makes a big impact on the design of a cross-device design tool. For example, Process 3 is not a good fit for a tool that takes a UI designed by a designer, and presents a generated UI for another device to that same designer.

We found that Processes 1 and 2 were the most common. Only Designer A2 said that they followed Process 3, which was a mistake, because he and his colleagues had trouble maintaining consistency among the various device-specific UIs. For example, one application would say "e-mail" and another would say "message." Consequently, they switched to Process 2 for their next revision.

## 2.4 Managing UI Consistency Across Devices

All of the designers said that maintaining consistency across devices was a major issue. While the interaction obviously cannot be the same across all devices, the designers said that parts of the UI should be, such as menu order, terminology, colors and graphics. Designer A2 said that it was easier to keep device-specific UIs consistent if designers grouped themselves by application rather than device, as mentioned above.

The most common way that the designers achieved consistency was simply to check their designs manually to make sure they were being consistent, which was tedious. They did not have any specialized tools for this purpose.

Designers A5 and A6 typically created an information architecture diagram first, and then designed the user interface off of that. Making sure their UI designs were consistent with the information architecture typically kept the designs consistent with each other.

## 2.5 Tools, Documentation, and Patterns

The tools that the designers used were similar to those used by other web and interaction designers. The most commonly mentioned tool was Microsoft Visio [122], which was used not only for conceptual diagramming, but also for laying out mobile phone UIs. Other tools included paper, whiteboards, Adobe Illustrator [2], and Microsoft FrontPage [121]. None of the designers used computer tools specialized for handling multiple devices.

Three companies (A1, A2, and A9) developed style guides for mobile UIs. Since Designer A2's company also makes software development tools, the company's long-term goal is to incorporate the style guide standards directly into a development tool for mobile UIs.

Designer A2 and his co-workers also tried to tackle the cross-device UI design problem by developing their own cross-device application flow language. However, they found it hard to design a language that could encompass both high-level application flow and device-specific interaction. They eventually abandoned the project due to lack of time and manpower.

All of the designers said they observed recurring interaction design patterns in their work. Designers A1, A2 and A9 documented their patterns, incorporating them into their companies' mobile UI style guides. The others did not document their patterns because they did not have enough time or did not think they were useful enough to document.

When we told the designers about our idea of making design patterns a cornerstone of a cross-device design tool, all but one of the designers were enthusiastic; Designer A8 was not sure whether designers would be able to recognize patterns in their work often enough to be useful. The designers also thought that enabling designers to create their own patterns and add them to the tool's pattern library was very important, and many thought it was crucial.

## 2.6 Need for Synchronized Changes Across Devices

The designers' reactions varied on whether it was important to see the mobile phone UI change while they edited the desktop UI, and vice versa. Four of the designers did not think it was important; they were concerned that the transformation process simply would not be good enough to warrant real-time change. Two designers would like to have the option. The others did not know.

## 2.7 Implications for Cross-Device UI Design Tools

From the above findings, we came up with the following implications for cross-device UI design tools.

*Presenting retargeting results for one designer is useful.* All of the designers designed the user interface for several features across multiple devices (as opposed to working on a particular set of features for only one device). Therefore, a tool that takes a designer's UI for one device and presents that designer with UIs for other devices fits within current design practices.

*Support for synchronous collaboration is not a high priority.* According to our interviewees, explicit support for multiple designers in a tool is not a high priority, since detailed design work for a particular feature is usually done by one designer.

*Designers need help maintaining consistency of content across devices.* Consistency was identified as a major burden of cross-device designers. The challenge is to keep the appropriate content consistent across devices, while letting the layout and navigation flow between screens change to fit the target device.

*Support for design patterns.* Using design patterns as the foundation of a cross-device UI design tool is a sound idea, but allowing designers to create their own patterns is essential for the long-term usefulness of this feature.

# 3 Prototype of Cross-Device Design Tool

While our interviews allowed us to discover general aspects of the design process that we needed to support, we wanted to get more detailed feedback about how an early-stage cross-device design tool should behave and what features it should have. It is hard for people to speculate about what such a tool should be like without interacting with one. Since there are no early-stage cross-design tools, we quickly designed and evaluated a prototype of one. The prototype, called *HopiSketch*, was built using DENIM [105] for the user interface and Hopi [17] for the retargeting process. Due to time constraints, this was done before the interviews in the previous section were completed; consequently, we were not able to incorporate all of the findings of those interviews into HopiSketch.

## 3.1 User Interface

We decided to use a sketch-based interface for the user interface of HopiSketch because designers usually sketch on paper during the early stages of design [140]. The user interface is based on DENIM, an existing sketch-based tool for early-stage web

The work described in this chapter was done in conjunction with Lawrence D. Bergman, Guruduth Banavar, Danny Soroker, and Richard J. Cardone at IBM Watson Research Center.

design.

DENIM has one window (see Figure 3-1) with three main areas. The center area is a canvas where the designer creates web pages, sketches the contents of those pages, and draws arrows between pages to demonstrate the behavior of hyperlinks (see Figure 3-2). On the left is a slider that is used to set the current zoom level. The bottom area is a toolbox that holds tools for drawing, panning, erasing, and creating and inserting reusable components.

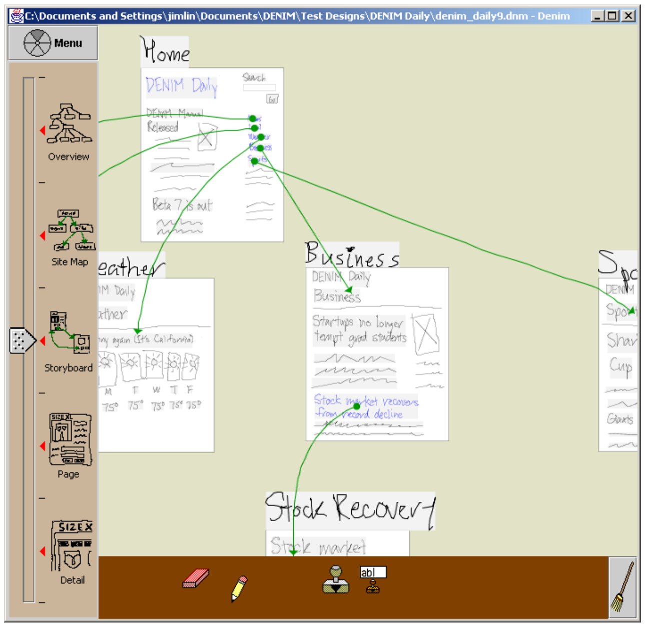

C:\Documents and Settings\jimlin\Documents\DENIM\Test Designs\DENIM Daily\denim_daily9.dnm - Denim
Menu
Overview
Site Map
Storyboard
Page
Detail
Home
DENIM Daily
Business
Stock Recovery
Stock market

**FIGURE 3-1** DENIM showing a typical design.

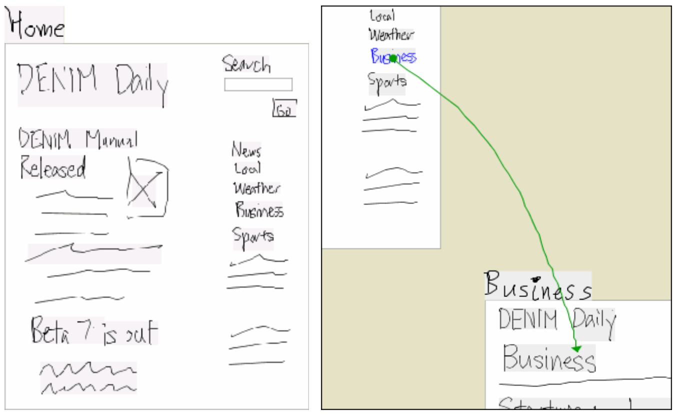


**FIGURE 3-2** a) A page with the label "Home" b) An arrow, whose source is a blue hyperlink, "Business."

Designers test the interaction of their designs in Run mode. Opening a pie menu over a page and selecting File→Run launches a separate browser window with the page loaded. The designer can navigate through the site design exactly like in a web browser, clicking on hyperlinks and using the Back and Forward buttons.

To create HopiSketch, we augmented DENIM to allow designers to insert radio

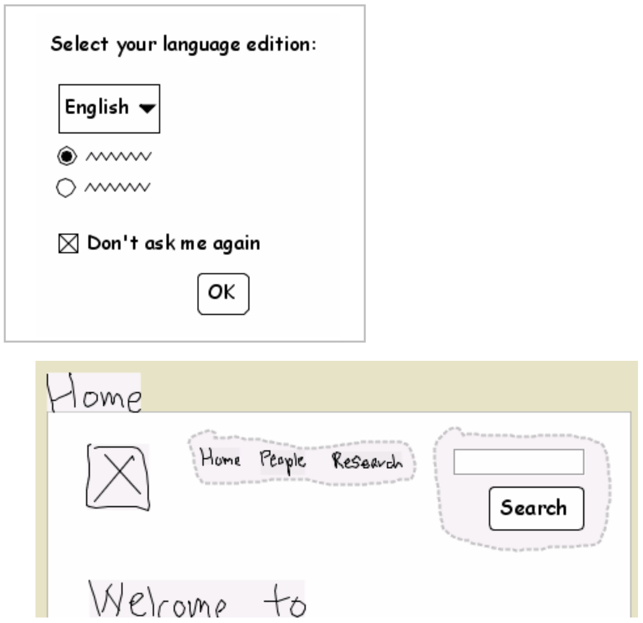


**FIGURE 3-3** a) *Top:* Web form widgets within a page.
b) *Bottom:* Two groups within a page: one containing *Home*, *People*, and *Research*; and the other containing a text box and a *Search* button.

buttons, check boxes, buttons, and drop-down boxes, which are commonly used in web applications, directly into their designs (see Figure 3-3a).

In addition, we added the ability for designers to group elements together to indicate that the elements are related. For example, a designer can group a text box and a Search button together to show that they should be treated as one unit (see Figure 3-3b). Groups also affect the behavior of radio buttons: within a group, only one radio button may be selected at a time.

HopiSketch focused on design for PCs and for Palm handheld devices. To retarget a PC design to the Palm, the designer presses a Retarget button. The tool takes the design, resizes the pages to fit the Palm's screen, and if needed, splits pages to minimize scrolling on the Palm. Elements within a retargeted page, such as handwriting and sketched images, are not resized or otherwise altered. Figure 3-4 shows a design for the PC and the results of retargeting the design to a Palm handheld.

## 3.2 Architecture

Figure 3-5 shows the overall architecture of HopiSketch. When designers press the

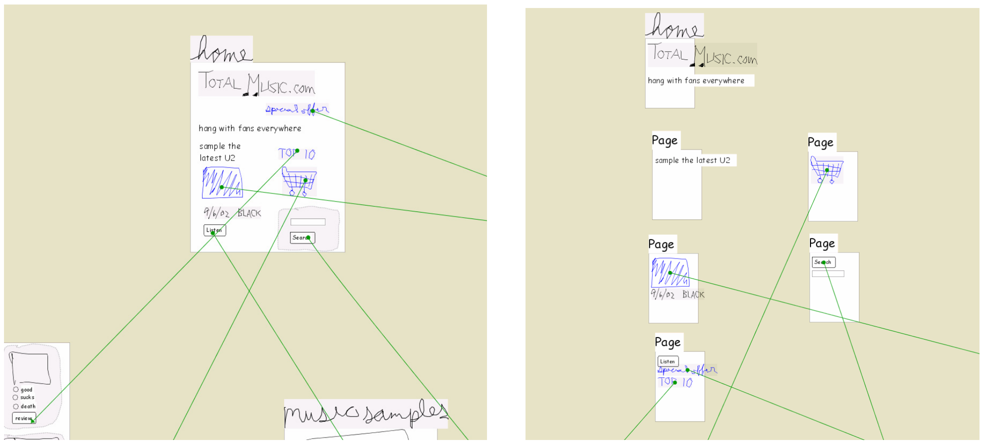


**FIGURE 3-4** *Left:* A UI design for the PC. *Right:* The design retargeted for the Palm handheld.

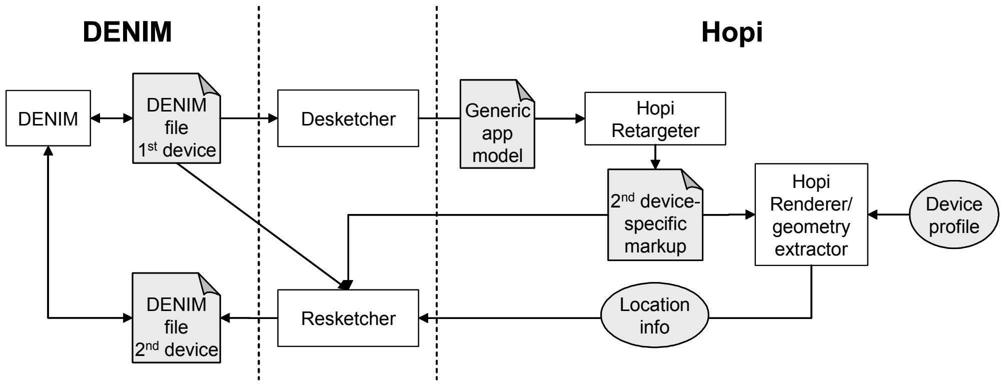


**FIGURE 3-5** The architecture of HopiSketch.

Retarget button, the system takes the design file and feeds it to a *de-sketcher*, which translates (or *de-sketches*) it into a generic model [49]. The model is based on XHTML [192] for general content elements and XForms [194] for form elements such as radio buttons and check boxes. One XHTML+XForms page in the model represents one page in the original DENIM file.

The model is then fed through Hopi, a system for designing cross-device web applications based on a generic model. The model first goes to Hopi's *retargeter*, which transforms it into a markup language for a target device. This process can result in one XHTML+XForms page being split up into several pages, depending on the characteristics of the target device. The retargeter creates pages that fit within a target device's screen, or are a little longer, allowing a bit of scrolling. The retargeter tries to keep elements that have been grouped together on the same page, although this is not always possible.

The resulting markup pages are then fed into Hopi's *renderer/geometry extractor*, which renders the markup using the predefined characteristics of the target device and determines the positions of elements in the markup.

Finally, a *re-sketcher* takes the markup, the extracted geometry, and handwritten elements from the original DENIM file, and creates a sketch-based version of the markup to be presented to the designer.

## 3.3 Evaluation of HopiSketch

To evaluate HopiSketch, we performed an informal task-based, usability test. The participants were introduced to our tool and then asked to create elements of a simple e-commerce site.

### *3.3.1 Participants*

Six designers participated in the usability study, four men and two women. All six designers were employed at user interface design or information architecture firms, had experience designing for the desktop web, and had at least some experience designing for mobile devices. Four of the designers have worked on cross-device user interfaces, although such interfaces are not the focus of their current work. Table 3.1 summarizes the characteristics of the participants.

### *3.3.2 Methodology*

The usability tests were performed on an IBM ThinkPad laptop with a Wacom Graphire tablet. First, we gave the designers a warm-up task to get used to the tablet. Next, we demonstrated HopiSketch and had the designers do some basic tasks, such as creating pages, adding elements to pages, and running the designs. Then, we asked the designers to create an online music store application for a desktop browser. We retargeted these desktop applications to Palm devices; the designers were then able to modify the generated results. About 60 minutes were available for the complete design task, including creating the desktop application and editing the Palm version.

**TABLE 3.1** Summary of participants of our prototype evaluation.

| Participant | Characteristics |
|---|---|
| B1 | UI designer<br>Graphic design background<br>Uses Photoshop and Illustrator<br>Has worked on > 20 cross-device projects |
| B2 | Interaction designer<br>Liberal arts background<br>Uses Photoshop, Fireworks, and Dream-weaver<br>Has worked on < 5 cross-device projects |
| B3 | Information architect<br>Programming and business background<br>Uses Photoshop, Visio, and Flash<br>Has not worked on any cross-device projects |
| B4 | Information architect<br>Media (TV, photography) background<br>Uses Visio and Photoshop<br>Has worked on < 5 cross-device projects |
| B5 | UI designer and usability engineer<br>Computer science background<br>Uses Illustrator and Dreamweaver<br>Has not worked on any cross-device projects |
| B6 | UI designer<br>Graphic design background<br>Uses Fireworks and Visio<br>Has worked on < 5 cross-device projects |

Finally, we debriefed the designers and had them fill out a questionnaire (Appendix B). We were looking for comments addressing two general themes:

- Were the tool and the generated user interfaces useful? Would the answer change depending on the number of devices being targeted?
- How can the tool be enhanced to better support the design of cross-device applications?

### 3.3.3 Results

We found that HopiSketch had implementation flaws that made it difficult for designers to perform some tasks. In particular, the de-sketching process was not sufficiently robust and mature to handle all the designers' sketches, which led to pages being split and elements within the pages being laid out in unexpected ways.

We also found that because the designers only had about 30–40 minutes to design for the desktop, their desktop designs were not very large. Consequently, some designers said that it would have been easier to simply re-sketch their small designs from scratch instead of starting from our generated user interfaces. Some of them were also slowed down by their lack of familiarity with the Wacom tablet.

Given the maturity of our prototype and the time constraints of the evaluation, most designers concluded that using HopiSketch was no faster than using paper and pencil for retargeting the designs that they had created. On the other hand, five of the six designers saw potential benefits of the tool within a broader context:

- Two of the designers, Designers B4 and B5, thought that for large designs, a design tool that can retarget could potentially save them a lot of time.
- Three of the designers also found value in the generated sketches, even though they were not ideal. Two of the designers, Designers B1 and B2, thought that the generated sketches still provided a useful starting point to design for the second device. Designer B2 said that by starting from the generated sketches, he would not forget to implement features in the PC version for the Palm version. Thus, if a feature was not present in the Palm version, it was because he explicitly deleted it from the generated design, not because he forgot to copy it from the pc version.
- Designer B1 said that the generated sketches were useful to show to clients, to demonstrate to them how unwieldy a Palm web site would be if it had all of the functionality of the PC web site.

- Another designer, Designer B6, said that he could imagine that a more robust version of the tool would generate sketches that would help him "see potential pitfalls (or opportunities)" in the design for the target device.

When we asked the designers the minimum number of target devices that would be required for a retargeting tool such as HopiSketch to be useful, all but one of the designers said two devices. One of them said that the tool would probably be most useful if the two devices were the same general type, such as from one mobile phone to another, as opposed to from desktop PC to mobile phone.

However, when we asked the designers how likely they were to use a commercial-strength retargeting tool for early-stage design, the reaction was more mixed. Three designers were likely to use one, one designer was neutral, and two said they were unlikely. One of the designers who was likely to use a retargeting tool said he would do so only if it were not sketch-based. This is because he would only use sketch-based tools for conceptual design, not for designing layouts for specific devices.

Finally, the designers gave us several suggestions that would make a retargeting tool more useful to them, which we describe in the next section.

## 3.4 Implications for Cross-Device UI Design Tools

The designers described a number of ways in which they believe a tool for retargeting designs could be more useful. Most of the suggestions are related to the theme of letting designers better understand, guide, and control the retargeting process. Each of the following suggestions was made by at least one designer. While these suggestions are not necessarily representative of the design community as a whole, we believe each suggestion has merit. We also discuss how Damask addresses these concerns.

*Control over retargeting.* Four of the designers mentioned that they would like to guide the retargeting process directly. They would like to be able to explicitly tag which sections of a page should be carried over to the target-device design, and which sections should be omitted, before the retargeting process takes place. One designer said he would like to make the tags conditional on what the target device is. Damask addresses this concern through the concept of *layers:* the layer in which an object is determines the devices in which it exists.

Another designer said that, when targeting the Palm, the tool should not split pages automatically, since the Palm handheld has scroll buttons. Instead, the tool should create pages that would scroll and then allow designers to split the pages themselves. This shows that information about the devices' characteristics must be taken into account throughout the retargeting tool for the tool to be effective. In Damask, the designer splits and merges pages manually.

*Iterative design.* Many designers wanted to better understand the retargeting process. For example, some said they would prefer a more iterative approach than the study permitted. Due to time and tool constraints, all of the designers went through the retargeting process only once. These designers would rather design a little bit for one device, retarget, look at the results, design a bit more for the first device, and so on. One designer specifically mentioned that he would like to see the design for the target device modified in real time while he worked on the design for the initial device. In Damask, retargeting occurs in real time.

There should also be a tighter relationship between designs of the same user interface on different devices. With HopiSketch, a retargeted design has no relationship to the original design once it has been generated. Ideally, the tool should be able to propagate changes made in a generated device-specific design back to the original. However, not all changes should be propagated. A designer may want to remove an element in a mobile phone version because it is unnecessary, but keep it in the desktop version because it aids navigation. Damask addresses this concern through the concept of *layers*.

*Templates and content replication.* Another theme was the ability to intelligently replicate content. For example, several designers mentioned that if they wanted a search box in the upper right-hand corner of every page, they would like to create a template that contains the search box, and apply that template to all of the pages in the site.

They also mentioned that if a page is split during retargeting, some elements in the original page, such as search or navigation aids, should be replicated on each of the resulting pages. Designers would need a way to specify which elements should be replicated, since it would be difficult to make such decisions automatically. Damask provides *templates* to support these concepts.

*Support for alternative design processes.* A cross-device tool should be flexible enough to support a variety of design practices, especially since cross-device design is a new discipline and design practices are still evolving. For example, our tool was designed to take a user interface for a large display, like a desktop PC, and retarget it to a device with a smaller display, like a Palm handheld. One designer said it was easier for him to add to a design than subtract from a design, so he would prefer to do the opposite of our tool: take a Palm user interface and merge its pages to form a desktop PC version. Damask lets designers start with any device they want.

*Improved page splitting.* All of the designers said that the algorithms for rearranging and splitting up content could be improved. One designer said that any handwriting and images should be shrunk to fit the dimensions of the handheld.

Damask does exactly this. Similarly, one designer mentioned that since Palm handhelds can scroll, groups should never be split between two or more pages. Instead, the tool should create a scrolling page that would keep all of the items of a group together. Damask lets designers manually split pages instead of automatically doing so.

*Sketch-based interface.* Some designers found the sketch-based interface appealing. Designer B1 said it took "napkin sketching to a new experiential level without making it beautiful," and that it allows him to focus on whether his ideas are valid. Designer B2 simply said that "it's a good way to work."

Others did not find it as compelling. Designer B4 wanted additional shape and alignment capabilities, such as provided by Visio or other diagramming tools. Designer B2 liked sketching, but said he uses sketching only for conceptual design. For layout design, he would prefer to use a more structured interface. Since opinion was split and we wanted to push the limits of user interface design, we decided to stick with a sketch-based interface for Damask.

Designer B1 suggested that the contents of the pages could contain a coarse grid similar to graph paper. This would help, but not force, designers to draw neater sketches, and would indirectly help the retargeting algorithms, since they work better when elements are aligned. Damask employs a grid for this purpose.

*Familiar interaction.* Some designers expressed reluctance to learn a new tool interface, and would have liked HopiSketch's user interface to have been more similar to the tools they already use. The most commonly mentioned tools were Adobe Photoshop and Microsoft Visio. Damask uses more familiar UI elements, such as standard toolbars and keyboard shortcuts.

*Handling different classes of devices.* There was some skepticism that our tool would be really useful for designing user interfaces to be run on different classes of devices, such as PCs and mobile phones. Designers B1, B2, and B3 said that the interaction flow is very different among different classes of devices, and that there is insufficient support in our tool to handle those differences.

A cross-device design tool should be able to support the design of applications whose user interfaces have very different interaction flows depending on the device. HopiSketch does not handle such design activities because it only transforms at the page and widget level. Higher levels of abstraction within the design are needed to support disparate interaction flows. Design patterns may be one such abstraction [104], and we will discuss how Damask uses them for this purpose in the next chapter.

# 4 Damask's User Interface

Damask is a design tool for the early-stage design and prototyping of cross-device UIs. It includes a catalog of design patterns for use in designing desktop-based web sites. Some of these patterns are also useful for designing mobile phone and voice UIs. These patterns include several generalized *solutions* capturing the essence of how to solve the problem stated in the pattern, one solution for web-style interaction on a PC, one for mobile phone displays, and one for prompt-and-response voice UIs.

With Damask, designers create their UI designs by sketching and by adding design pattern solutions to their design for one device. As they are creating their designs, Damask generates corresponding UI design sketches for the other two devices, which the designers can modify if desired. Finally, designers can use Damask to run their designs in a Run mode, or in the case of voice UIs, also export them to VoiceXML, so that they can test and interact with their design sketches.

First, we will describe Damask's user interface. Then we will walk through an example of how designers would create and run their UI designs, including how they would use design patterns and layers within their designs. In the next chapter, we will describe Damask's architecture and implementation.

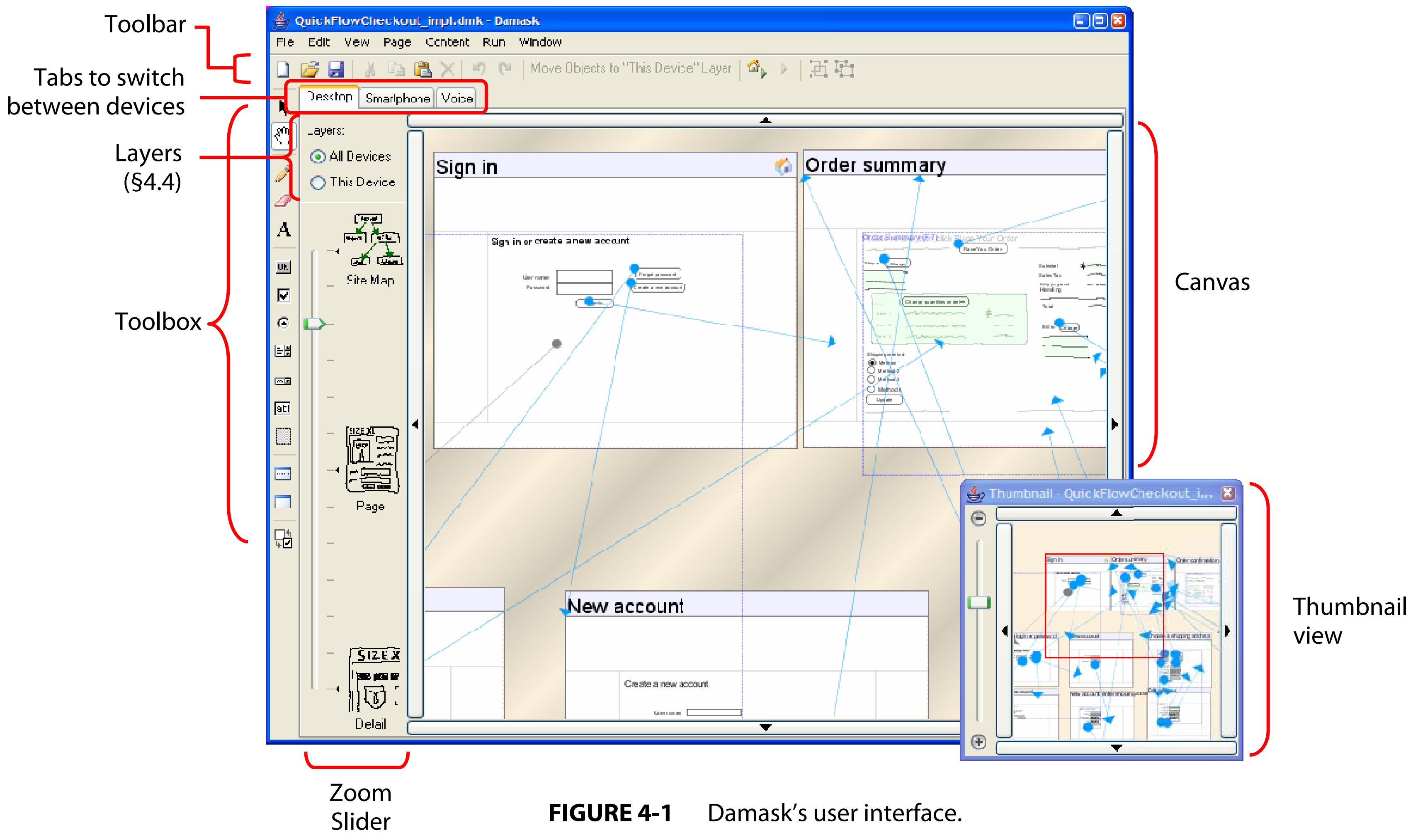

Toolbar
Tabs to switch between devices
Layers (§4.4)
Toolbox
Zoom Slider
Canvas
Thumbnail view
QuickFlowCheckout_impl.dmk - Damask
File Edit View Page Content Run Window
Move Objects to "This Device" Layer
Desktop Smartphone Voice
Layers:
All Devices
This Device
Site Map
Page
Detail
Sign in
Order summary
New account
Sign in or create a new account
Create a new account
Thumbnail - QuickFlowCheckout_i...

**FIGURE 4-1** Damask's user interface.

## 4.1 Damask's User Interface

Damask's user interface is similar to other design tools that our research group has developed, such as DENIM [105, 141] and SUEDE [91, 168] (see Figure 4-1).

The canvas contains the actual user interface design. The design includes which patterns it is using, as denoted by a blue outline and the name of the pattern. For example, in Figure 4-1, there is an instance of the ORDER SUMMARY pattern in the "Order summary" page. There are tabs above the canvas where designers can choose which target device they are viewing: desktop, smartphone, or voice. To view the different device-specific UIs at the same time, the designer can view the design in multiple windows.

To pan around the canvas, the designer can use the scroll buttons along the edges of the canvas, or use the hand tool in the toolbox and drag it over the canvas. The designer uses the zoom slider, just to the left of the canvas, to zoom in and out.

Damask also includes a Thumbnail view, which is a miniature view of the canvas that is similar to the Navigator palette in Adobe Photoshop [3]. The red rectangle within the thumbnail shows the current view of the canvas in the main window. The designer can move or resize the rectangle, which pans and zooms the main canvas appropriately.

Damask also has a Pattern Browser window (see Figure 4-2), where designers can browse for patterns to be instantiated in their designs, find the details about a particular pattern, and instantiate a pattern.

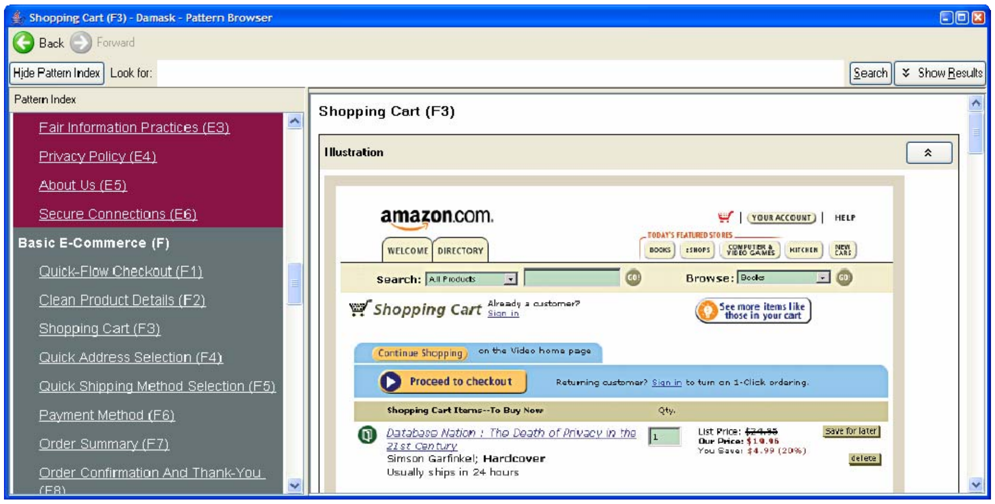


**FIGURE 4-2** Damask's pattern browser.

## 4.2 Designing Desktop and Mobile Phone UIs

A graphical desktop or mobile phone UI design consists of pages of content, linked together with transition arrows that define behavior, in the spirit of DENIM.

### 4.2.1 *Pages*

To create a page, the designer uses the pencil tool and drags out a rectangle on the canvas indicating the size of the new page (see Figure 4-3). By default, Damask will create a page of a default size if the designer draws a rectangle smaller than that size. The default size depends on the device (1024×768 for desktop, 180×220 for smartphones).

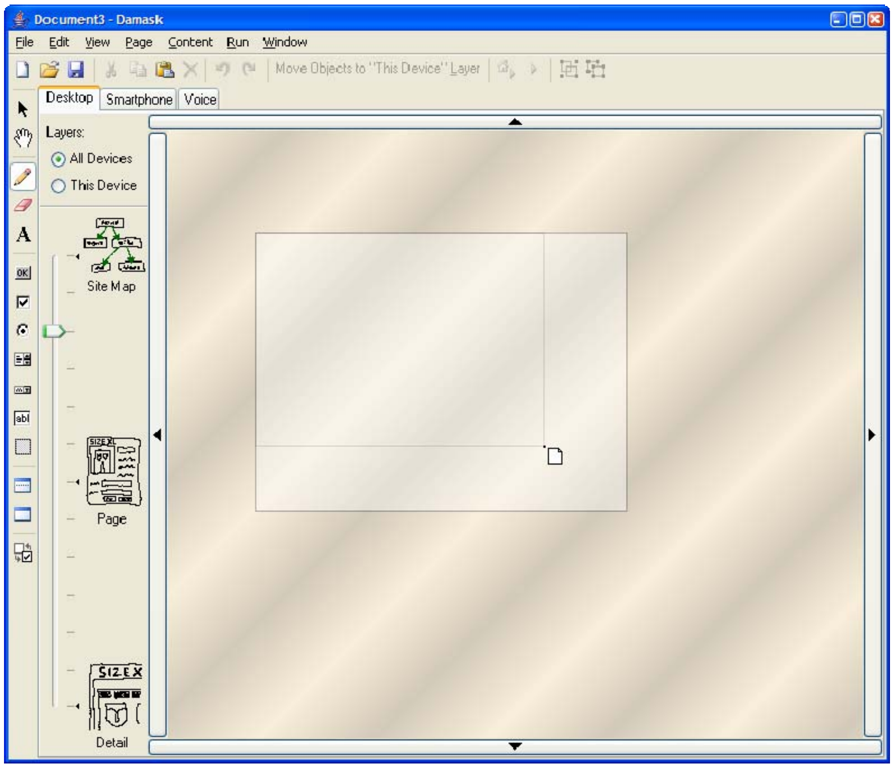


**FIGURE 4-3** Creating a page.

The designer can set the focus to a page by tapping in the page. Damask then adds a drag bar to the top of the page so that it can be moved, and a grip in the lower right-hand corner so that it can be resized. If the designer makes the page larger than the default size, Damask will draw a red dotted "fold" line to indicate that an end-user would have to scroll to see any content below or to the right of the fold (see Figure 4-4).

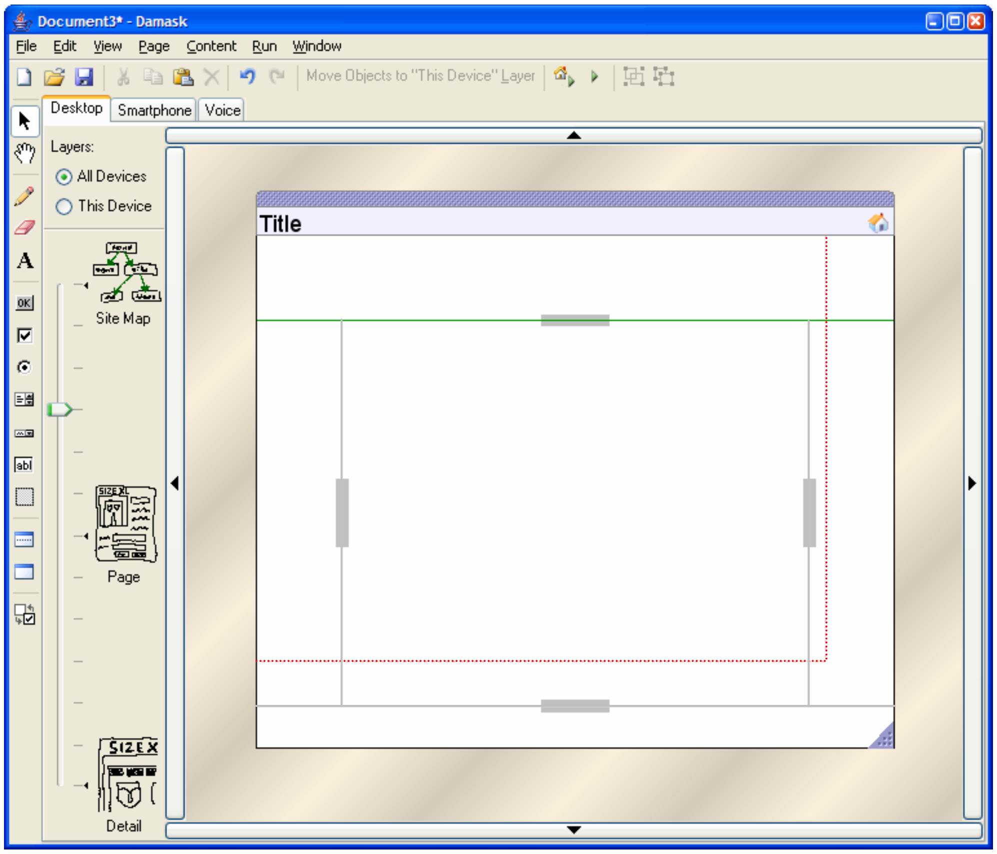


**FIGURE 4-4** A page, with gray bars for dragging and resizing regions and a red dotted "fold" line.

Each page is made up of several page regions: north, south, east, west, and center. These regions reflect the sections into which a page is typically split up, for example, north for the navigation bar, east for related items, and center for the main content. In desktop pages, all five page regions are visible. In smartphone pages, only the north, south, and center regions are visible, since a smartphone's screen typically isn't wide enough to have content along the sides. A designer can resize a region by dragging the gray bars in the center of a region boundary (see Figure 4-4).

When Damask's zoom level is near the Page level or lower (as indicated by the zoom slider to the left of the canvas), a light blue grid is drawn in the background of the page, as a subtle guide to help designers lay out elements within the page.

Damask has no alignment tools, because we want to maintain a lo-fi feel for the user interface [99], and not tempt designers into spending too much time fiddling with details, during the early phases of the design process [200].

A designer can cut, copy, and paste pages through the Page menu on the main menu bar, or by right-clicking on a page. The designer can also set the home page of the UI through that menu. By default, the page that is first created is the home page. The home page is denoted by a home icon in the upper right-hand corner of the page (see Figure 4-4).

Damask includes tools to split and merge pages (see Figure 4-5). To split a page, designers select the split tool from the toolbox and then click at the location in the page where they want it to be split horizontally (see Figure 4-6). They can merge two pages they had previously split by clicking on the merge tool and then clicking on one of the two pages.

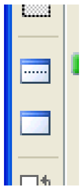


**FIGURE 4-5** The tools for splitting a page (above) and merging pages (below).

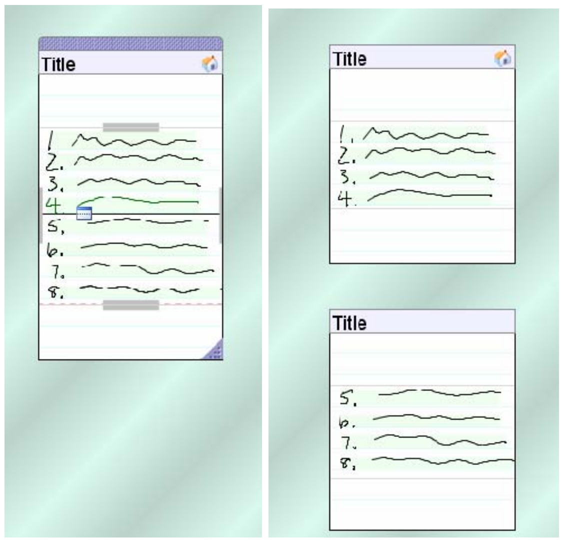
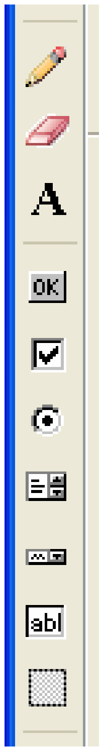


**FIGURE 4-6** *Left* Using the split tool to split a page. *Right* The result of splitting a page.

### 4.2.2 *Adding user interface controls to pages*



To add content and interaction to a page, a designer adds *controls* to the page. There are eight types of controls: labels, buttons, check boxes, radio buttons, list boxes, drop-down boxes, text boxes, and panels (see Figure 4-7).

A label has three modes: ink, typed text, and an image. To add ink, the designer selects the pencil tool ✏ and draws inside the page. Damask adds ink strokes that are spatially close to the most recently created label, which has a green border, to that label (Figure 4-8). Each label has a light-green background to indicate to the designer how Damask has grouped together the strokes.

**FIGURE 4-7** The toolbox buttons used for creating and erasing controls.

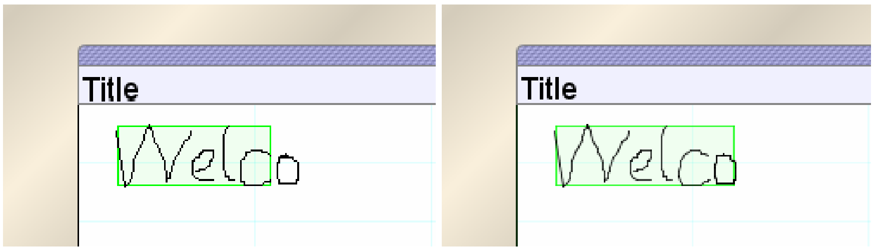


**FIGURE 4-8** Adding an ink stroke to a page.

If designers want to group ink strokes in a different way, they can select the labels which they want to be merged into one label, and then click the group tool in the top toolbar. Damask ungroups the selected labels into individual ink strokes, and then regroups them into one label (see Figure 4-9). Clicking the ungroup tool simply ungroups the selected labels into separate strokes without regrouping them.

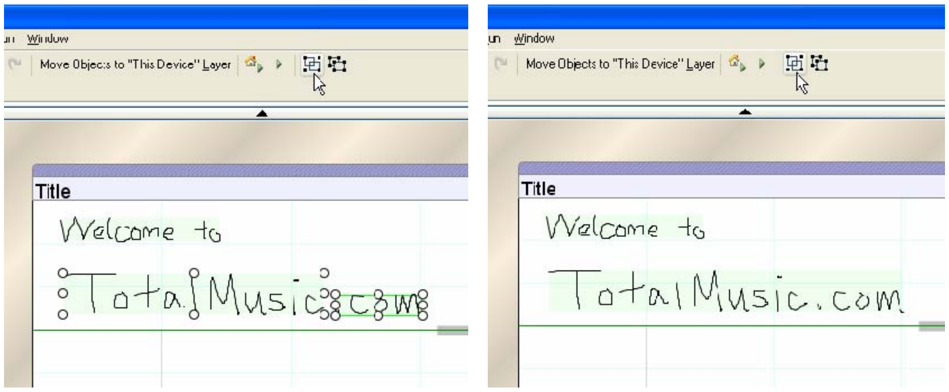

Window
Move Objects to "This Device" Layer
Title
Window
Move Objects to "This Device" Layer
Title

**FIGURE 4-9** Grouping labels together into one label, by clicking the Group button in the toolbar.

To add typed text, the designer selects the text tool **A**, clicks on the canvas, and types in a label. To edit the text, the designer selects the text tool, clicks on the label, and types.

A designer can also change the display mode of a label, e.g., from ink to text, by right-clicking on the piece of content, and choosing the new type from the menu (see Figure 4-10). This allows the maintenance of different modes of the same label in parallel. This is useful for when designers want to show a more finished version of a design to a client, but then want to go back to a more informal representation so they are not unnecessarily distracted by details while they change their design.

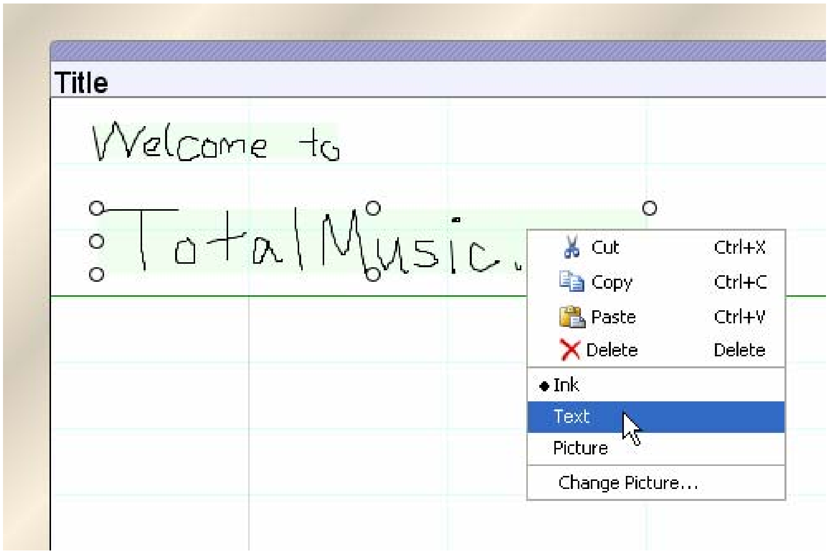


**FIGURE 4-10** Changing the display mode of a label.

This is also the mechanism to add an image; the designer first creates a label by either sketching or typing, then right-clicks on the label, selects Change Picture… from the resulting pop-up menu, and then selects an image from the file chooser.

To add another type of control, the designer selects the corresponding tool in the toolbox ( ), and then clicks within the page. To modify the textual content of a control, the designer selects the *text* tool **A** and then clicks on the control (similar to how Microsoft PowerPoint behaves). The caption's text is then edited in place, or in the case of a list box or drop-down box, a dialog box pops up allowing the designer to edit the control's list of text.

To change the default state of a control, like making a particular check box checked by default, the designer selects the change control state tool at the bottom of the toolbox. The designer can then interact with the design as if the design were live, for example, clicking on a radio button to change which one is selected (see Figure 4-11).

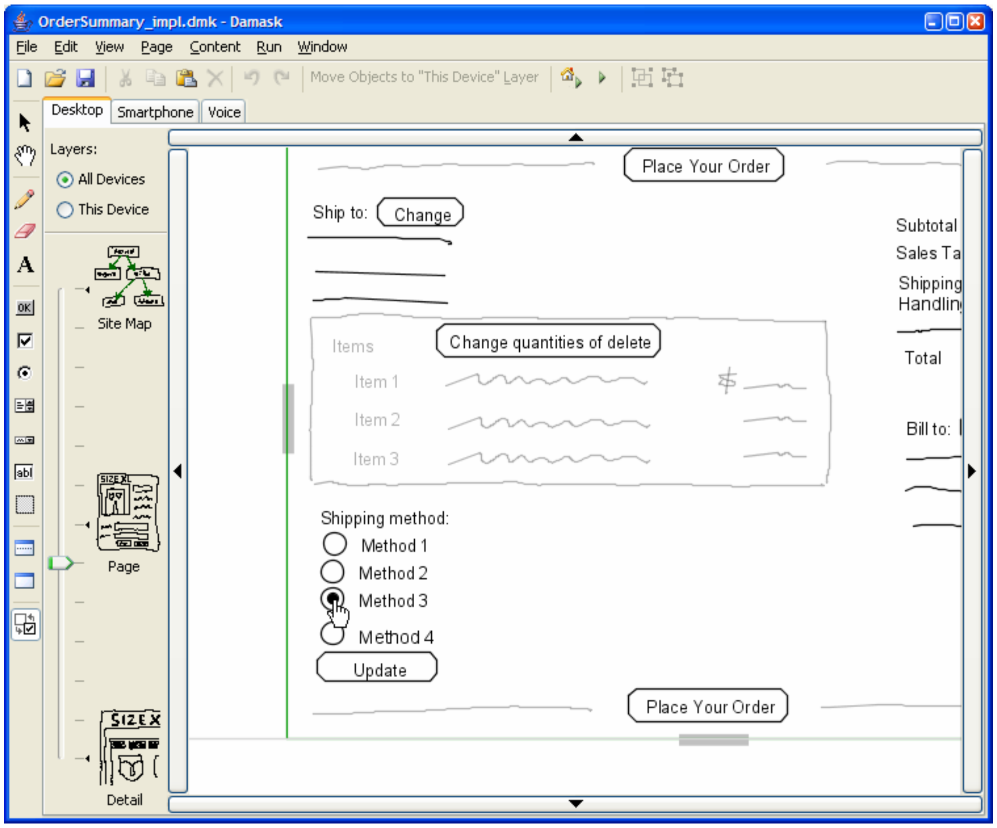


**FIGURE 4-11** Changing the state of a radio button.

Panels are used to logically group related sets of controls manually. They are not meant to be visible in the final user interface. They are also used to separate groups of radio buttons; for any set of radio buttons within a panel, only one can be active at any time. Also, for all of the radio buttons outside of every panel, only one radio button can be active. To use a panel, the designer selects the panel tool from the toolbox ▨ and drags out a rectangle, and then adds or pastes the desired controls within the panel (see Figure 4-12).

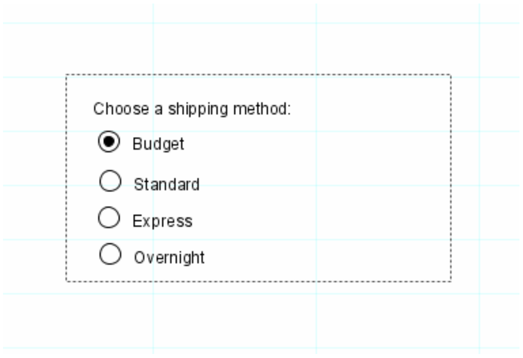


**FIGURE 4-12** A panel containing a label and radio buttons in the canvas. The dotted border is invisible at run time.

### *4.2.3 Specifying behavior with arrows*

An arrow between two pages represents a relationship between those pages. Like DENIM, Damask provides organizational and navigational arrows (see Figure 4-13).

*Organizational* arrows are used to represent a relationship between two pages. For example, the designer may want to show that the concepts between two pages are related, or that the end-user tends to go from one particular page to another, but the designer does not want to fill in the details on how at this time. These arrows are generally created early in the design process, during the information architecture phase [140, 141].

A *navigational* arrow specifies a transition from one page to another in the interface. Such an arrow from an item on one page, such as a word, image, or button, to another page means that, in Run mode, the user can click on the item to transition to the other page. More specifically, if a navigational arrow starts from a word, the word behaves like a hyperlink on a web page in Run mode.

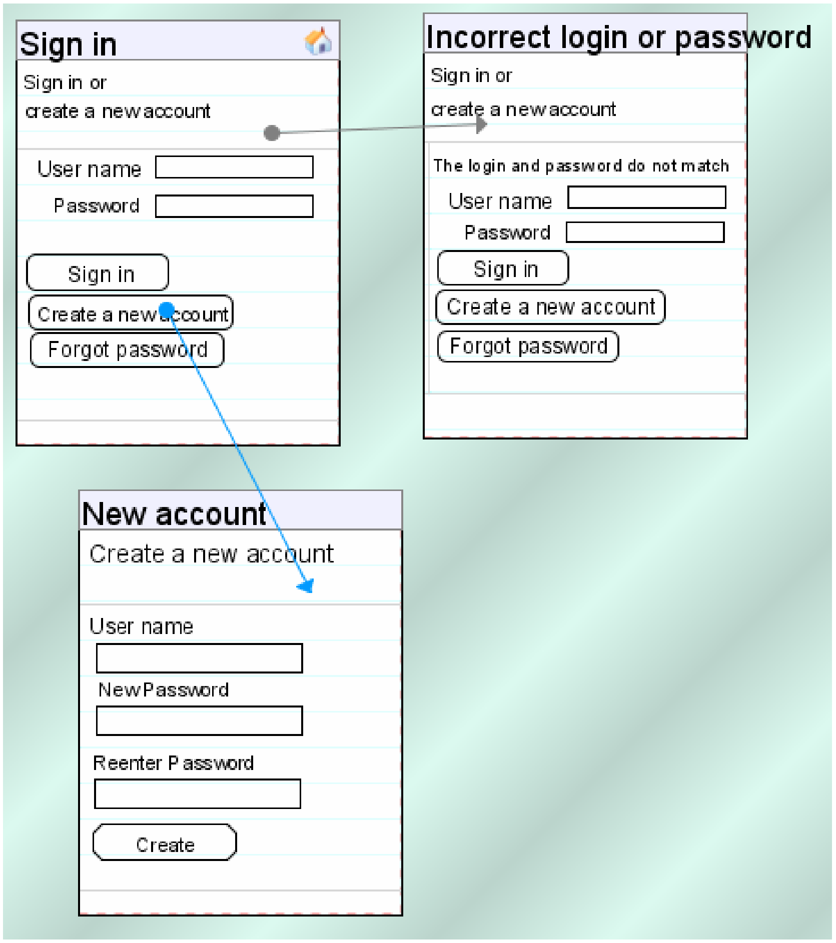


**FIGURE 4-13** An organizational arrow (gray) between the "Sign in" and "Incorrect login or password" pages, and a navigational arrow (blue) between the "Create a new account" button and the "New account" page.

To create an arrow, the designer draws a stroke using the pencil tool between two pages. The system checks if the stroke is an arrow. Organizational arrows start on one page and end in another. This creates a gray arrow from the source to the destination. Navigational arrows start on a specific object in one page and end in another page. This creates a green arrow from the source to the destination. When creating a navigational arrow, any organizational arrows from the source page to the destination page are removed, since the navigational arrow now has at least the same information as the organizational arrow did. As additional feedback, if the arrow originates from a text or ink object, that object becomes blue, like a hyperlink in a web page.

### 4.2.4 Manipulating objects within a page

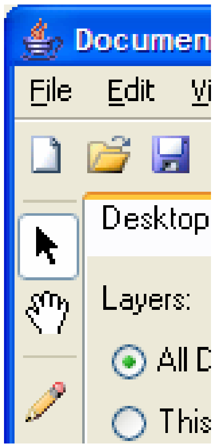


**FIGURE 4-14** The pointer tool.

A designer can select an object by choosing the pointer tool in the toolbox (see Figure 4-14), and then either clicking on an object, or shift-clicking on or drawing a selection rectangle around multiple objects. The designer can then move the object (or objects) by dragging, resize by dragging one of the resize handles, or cut, copy, and paste through a right-click context menu, the Edit menu, or the standard keyboard shortcuts.

### 4.2.5 Templates

Web sites often have common elements across many pages. Damask includes a templates feature so that designers do not have to create the same elements on every page. To see the templates associated with the design, the designer chooses View→Templates. The Templates pane slides in from the right (see Figure 4-15).

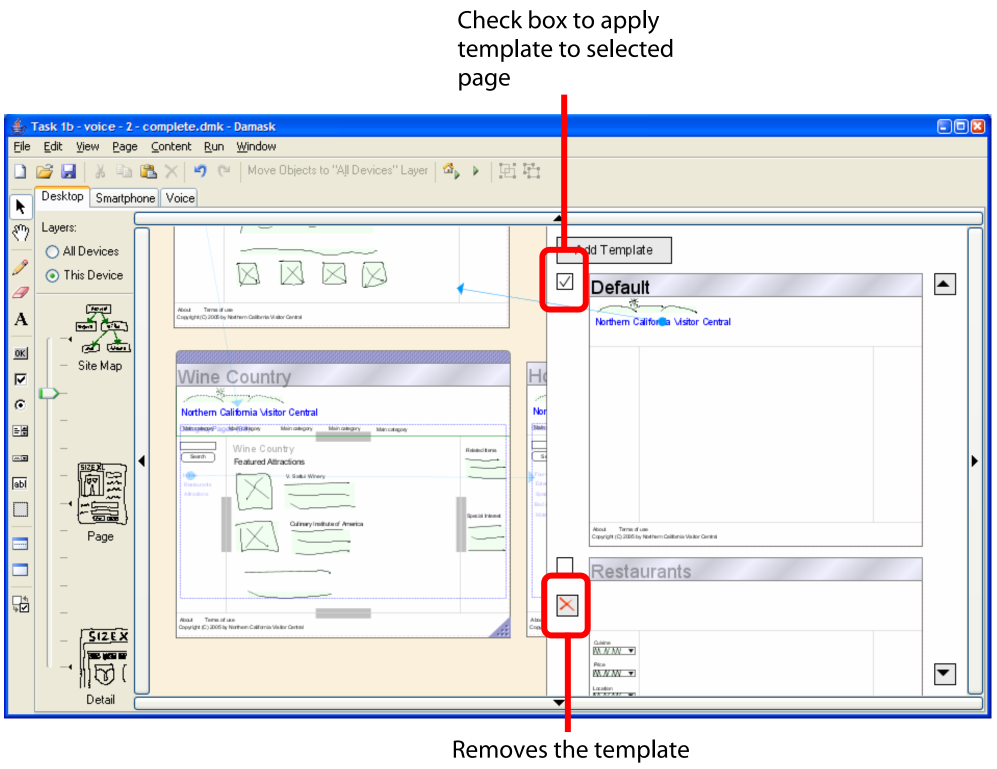

Check box to apply template to selected page
Removes the template
Task 1b - voice - 2 - complete.dmk - Damask
File Edit View Page Content Run Window
Move Objects to "All Devices" Layer
Desktop Smartphone Voice
Layers:
All Devices
This Device
Site Map
Page
Detail
Add Template
Default
Restaurants
Wine Country
Northern California Visitor Central

**FIGURE 4-15** The template pane. Clearing the check box next to the default template would remove it from the currently selected page, in this case, the Wine Country page.

When the designer adds elements to a page in the Template pane, all of those elements are added to every page that uses that template. To apply a template to a particular page, the designer first clicks in that page to give it focus if necessary, then clicks the check box to the left of the desired template page. By default, the template pane includes a page titled "Default" that is applied to every page in the design, although it can be removed on a page-by-page basis. Designers can add more templates by clicking on the Add Template button at the top of the pane, and remove a template by clicking the button with a red X next to that template. Templates added by the designer are *not* applied to every page by default. More than one template can be applied to a page; the elements are simply composed together.

### 4.2.6 Run window

Damask allows designers to interact with their design sketches in a Run window, where designers can test out the interaction of their designs. Selecting Run→Run from Selected Page displays the page with the focus in the Run window; Run→Run from Home Page displays the home page. There are also two toolbar buttons corresponding to the two menu items (see Figure 4-16).

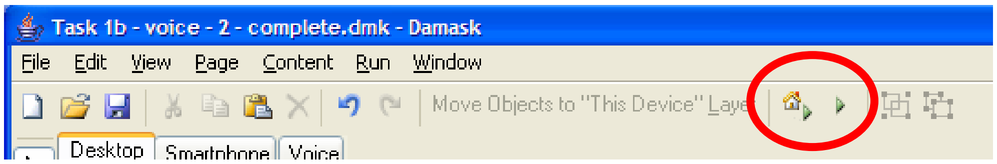

Task 1b - voice - 2 - complete.dmk - Damask
File Edit View Page Content Run Window
Move Objects to "This Device" Laye
Desktop Smartphone Voice

**FIGURE 4-16** The toolbar buttons for *Run from Home Page* and *Run from Selected Page*.

Inside the Run window, the designer can navigate through the design as if it were running in a web browser. If the Run window is displaying a desktop page, back and forward buttons are provided to simulate a web environment. Any changes made to the page in the main Damask window are automatically reflected in the Run window. If it is displaying a smartphone UI, then a telephone keypad is put beneath the displayed UI, although currently only the Back button is active (see Figure 4-17).

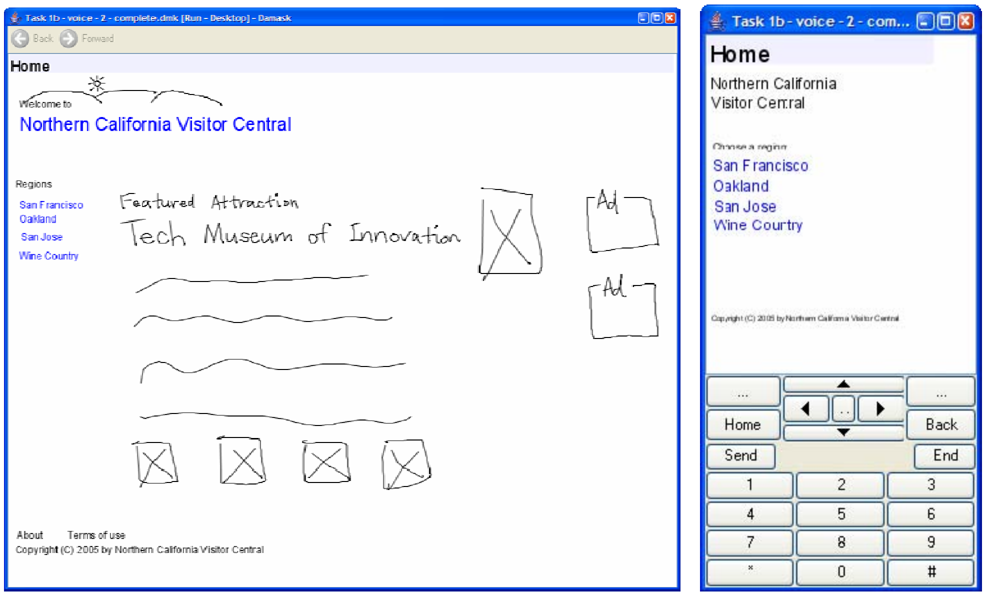

Task 1b - voice - 2 - complete.dmk [Run - Desktop] - Damask
Back
Forward
Home
Welcome to
Northern California Visitor Central
Regions
San Francisco
Oakland
San Jose
Wine Country
Featured Attraction
Tech Museum of Innovation
Ad
Ad
About
Terms of use
Copyright (C) 2005 by Northern California Visitor Central
Task 1b - voice - 2 - com...
Home
Northern California
Visitor Central
Choose a region
San Francisco
Oakland
San Jose
Wine Country
Copyright (C) 2005 by Northern California Visitor Central
Home
Back
Send
End
1
2
3
4
5
6
7
8
9
*
0
#

**FIGURE 4-17** Run windows for desktop (left) and smartphone (right) designs.

## 4.3 Designing Voice UIs

Designing a voice user interface in Damask is quite different than designing a web or smartphone UI. Damask's voice mode is similar to SUEDE's design graph area [91]. A voice user interface is represented by computer *prompts*, connected by possible human *responses*. Related prompts and responses are grouped together into *forms*. There are only four tools available in voice mode: selecting and moving objects, panning the canvas, creating objects, and removing objects (see Figure 4-18).

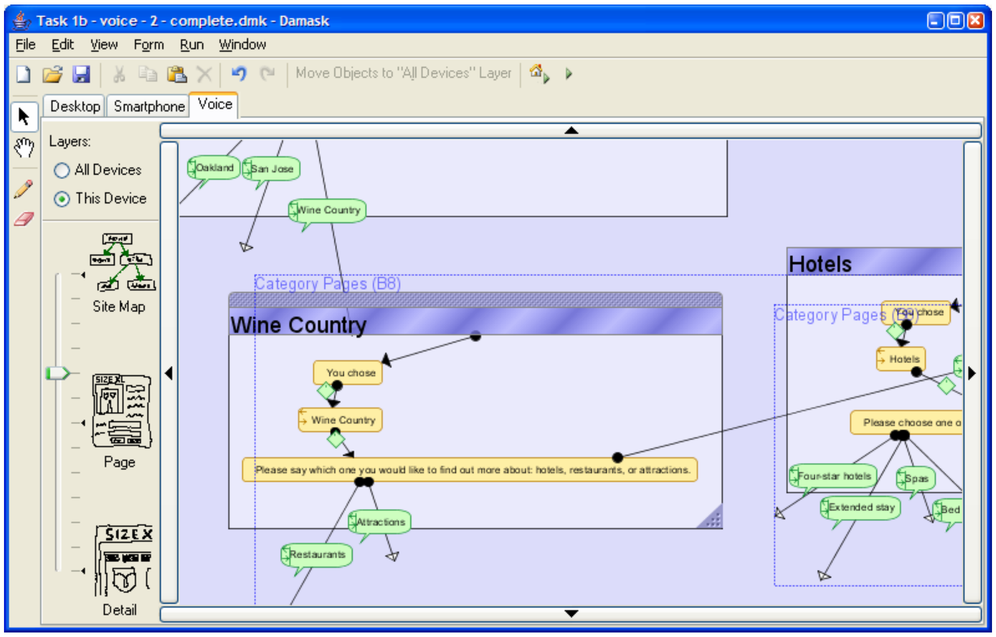


**FIGURE 4-18** A voice UI design, including a form titled "Wine Country," orange computer prompts and green user responses.

### 4.3.1 *Forms*

Forms are similar in concept to forms in VoiceXML interfaces and analogous to pages in web sites. They group together related prompts and responses into a single entity. Creating and manipulating forms is the same as interacting with pages (Section 4.2.1).

### 4.3.2 *Prompts*

Prompts represent phrases that the computer speaks. To create a prompt, the designer uses the pencil tool , taps within a form, and types the prompt (see Figure 4-19). To edit the text in a prompt, the designer uses the pencil tool and taps within the prompt. To move the prompt, the designer uses the selection tool .

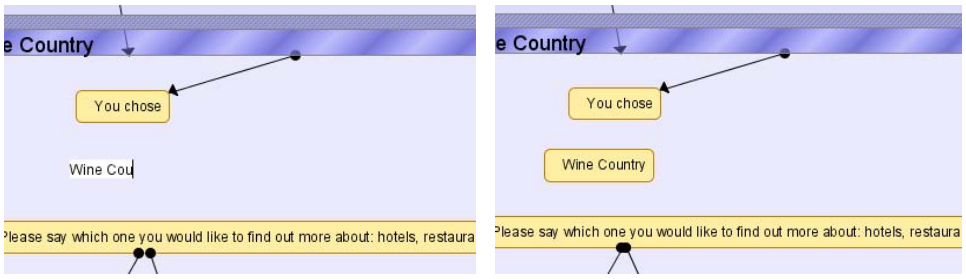

e Country
You chose
Wine Cou
Please say which one you would like to find out more about: hotels, restaura
e Country
You chose
Wine Country
Please say which one you would like to find out more about: hotels, restaura

**FIGURE 4-19** Inserting a prompt.

### *4.3.3 Responses*

Responses represent the phrases that people say in response to prompts. To create a response, the designer uses the pencil tool , and drags a line from the prompt to which the response is responding, to another prompt which the computer will say in reply to the response. The line becomes an arrow; a voice balloon with the response text appears along the arrow (see Figure 4-20). It is possible to create a response which does *not* end at a prompt; it is simply a dead end until it is hooked up to another prompt.

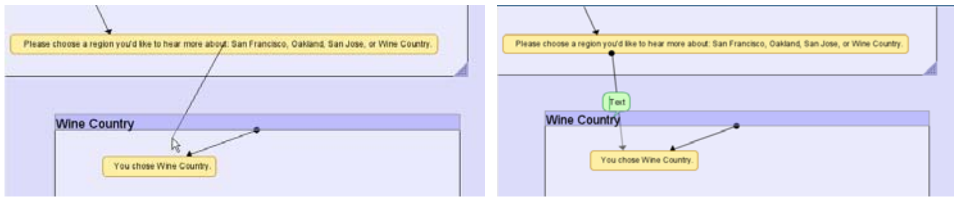

Please choose a region you'd like to hear more about: San Francisco, Oakland, San Jose, or Wine Country.
Wine Country
You chose Wine Country.
Please choose a region you'd like to hear more about: San Francisco, Oakland, San Jose, or Wine Country.
Text
Wine Country
You chose Wine Country.

**FIGURE 4-20** Creating a response.

The designer can drag the endpoints of the arrow to other prompts. To edit the response text, the designer taps on the text with the pencil tool . If designers want more than one phrase to go between the same prompts, they can separate the possible response phrases with carriage returns (see Figure 4-21).

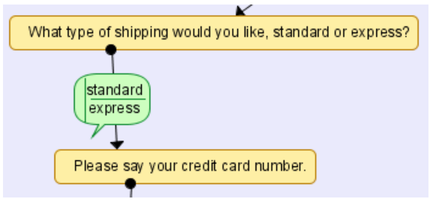

What type of shipping would you like, standard or express?
standard
express
Please say your credit card number.

**FIGURE 4-21** A response with more than one possible phrase.

There are two special cases for response text:

- An asterisk (*) will match any response that is not matched by any other response. If there are no other responses, then * matches anything.
- No text in a response from Prompt A to Prompt B means that the computer says Prompt B immediately after Prompt A. In this case, the voice balloon becomes a diamond shape, which the designer can click on to add text back into the response.

A designer can create a response that goes from a prompt directly to another form, as opposed to another prompt. In this case, the destination prompt of the response is assumed to be the *initial prompt* of the form, indicated by a special empty response which is anchored at the top of the form and points to the initial prompt. In Figure 4-22, “What is your favorite vegetable?” is the initial prompt. Designers can change the initial prompt by moving the endpoint of the special response.

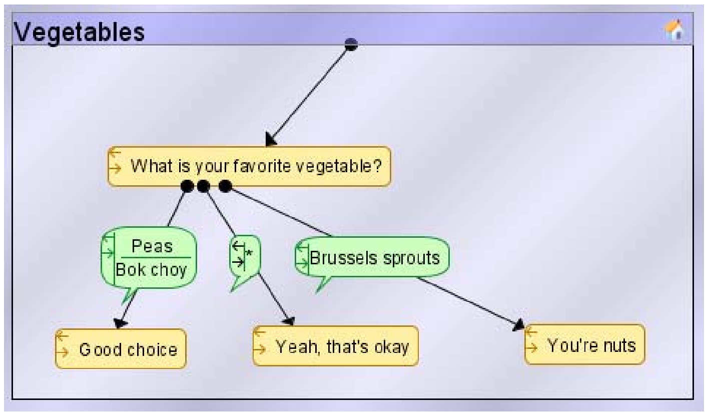


**FIGURE 4-22** The prompt *What is your favorite vegetable?* is the initial prompt in this form.

### 4.3.4 Universals

*Universals* are commands that a user can say at any time to perform a global action. Designers create universals in the Templates pane, which contains a default form with a blank prompt that is used solely for anchoring the start of universal responses. To create a universal, the designer creates a response from that prompt to the prompt that should be played when an end-user says the universal (see Figure 4-23).

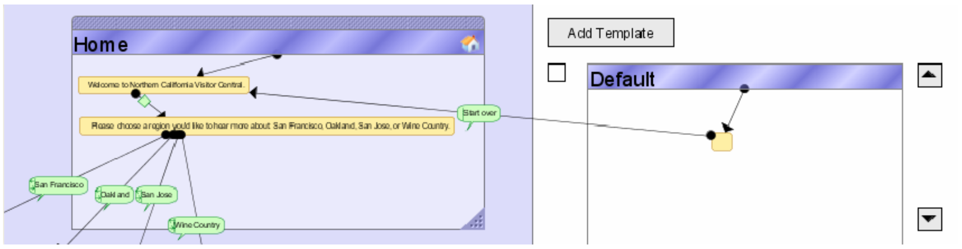
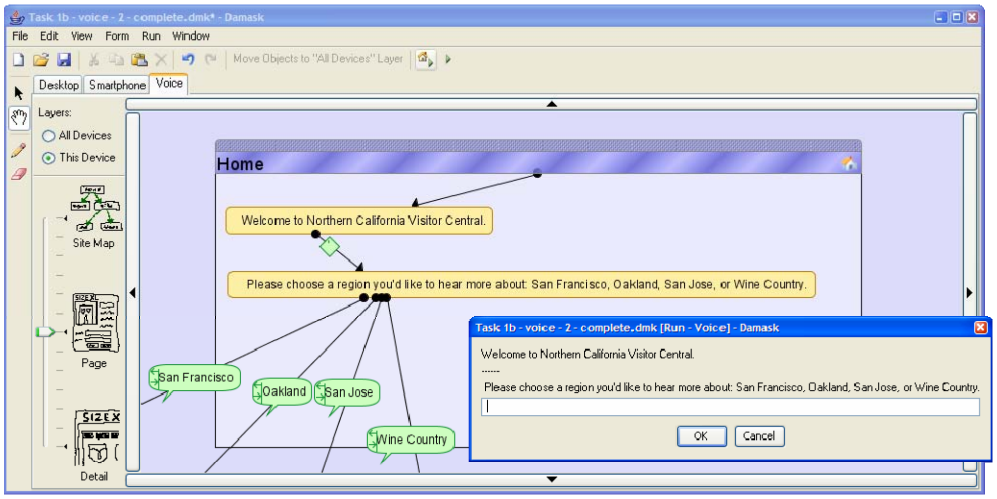

Home
Welcome to Northern California Visitor Central.
Please choose a region you'd like to hear more about: San Francisco, Oakland, San Jose, or Wine Country.
Start over
San Francisco
Oakland
San Jose
Wine Country
Add Template
Default

**FIGURE 4-23** A universal which goes to the Home form whenever a user says "Start Over."

### 4.3.5 Run window

When the designer chooses Run from Home Form or Run from Selected Form from the Run menu, a dialog box pops up with the first prompt in the form as the message of the dialog box. If there is an empty response (i.e., a response with no text) that starts from this prompt, then the dialog box also shows the prompt at the other end of the empty response, and repeats this process until it reaches a prompt that has no empty responses.

A user types in his or her response in the text box below the computer's prompt and clicks OK (see Figure 4-24). A text box is used in the window so that the designer can test whether the display prompt contains enough information so that a user knows what to say in response. This technique has been used in other voice development tools such as the VoiceXML terminal in Tellme Studio [181]. The user can click Cancel to end early. If a prompt has no responses, then only an OK button is displayed, without a text box.



Welcome to Northern California Visitor Central.

Please choose a region you'd like to hear more about: San Francisco, Oakland, San Jose, or Wine Country.

Task 1b - voice - 2 - complete.dmk [Run - Voice] - Damask

Welcome to Northern California Visitor Central.

------

Please choose a region you'd like to hear more about: San Francisco, Oakland, San Jose, or Wine Country.

**FIGURE 4-24** Run mode for voice user interfaces, with the same form in design mode in the background.

If the response matches one of the designer's responses, then the appropriate prompt or prompts are displayed in the dialog box. If the user types in nothing, then the dialog box says, "Sorry, I didn't hear you," and redisplays the prompt. If the user's response does not match one of the responses that the designer defined, then the dialog box says, "Sorry, I didn't catch that," and redisplays the prompt.

The Run window is intended for the designer to do a quick sanity check of the voice UI design. Nonetheless, reading a prompt is not the same as listening to a prompt. For this, there is an option to export the design to VoiceXML, which is described next.

### *4.3.6 Exporting to VoiceXML*

To verbally interact with the voice UI, Damask allows designers to export their voice UI to a VoiceXML [193] file, which they can then upload to a web site such as Tellme Studio or BeVocal Café [24], or Voxeo Evolution [191], and interact with their

VoiceXML file over the phone. The generated file is not meant to be used for final deployment of a voice UI; it is solely for prototyping purposes. To generate the file, the designer chooses Export to VoiceXML from the File menu.

The generated file behaves in the following way. The computer first says the first prompt in the home form. If there is an empty response that starts from this prompt, then the computer also says the prompt at the other end of the empty response, and repeats this process until it reaches a prompt that has no empty responses.

A user says something in her response. If the response matches one of the designer's responses, then the computer says the appropriate prompt or prompts. If the user says nothing after a period of time, then the computer says, "Sorry, I didn't hear you," and repeats the prompt. If the user's response does not match one of the responses that the designer defined, then the computer says, "Sorry, I didn't catch that," and repeats the prompt.

You may notice that the VoiceXML file is the audio equivalent of the Run window discussed in the previous section. In fact, the Run window was first created to test the VoiceXML generation feature.

### 4.3.7 Limitations of the VoiceXML export feature and the Run window

While the Run window and the VoiceXML export feature are useful for quick tests of voice UIs, they are not ideal for doing extensive usability tests with potential users. The biggest drawback is that if the user does not say one of the choices that the designer had chosen beforehand, then the Run window or the VoiceXML file does not recognize the user's response and gets stuck. Also, text-to-speech (TTS) systems still sometimes have problems saying natural-sounding speech, and Damask does not have a facility for recording human speech for either prompts or responses. For this, a Wizard-of-Oz interface, like the one in SUEDE [91], would be more suitable.

## 4.4 Layers

Layers are used as the basic concept for determining whether a page/form or a control appears on every device or just one device. For each device, there are two layers: All Devices and This Device. An object in the All Devices layer exists in every device, although the location of that object can vary across devices, and moving an object in one device does not affect the corresponding object in the other devices. An object in the This Device layer exists only for the device that is currently being viewed.

The designer switches between the two layers by choosing the appropriate radio button next to the top left-hand corner of the canvas (see Figure 4-25). If designers select an object that is not in the current layer, Damask pops up a dialog box asking if they would like to switch to the other layer so that they can select the object.

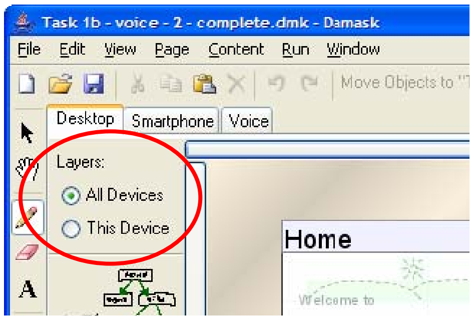
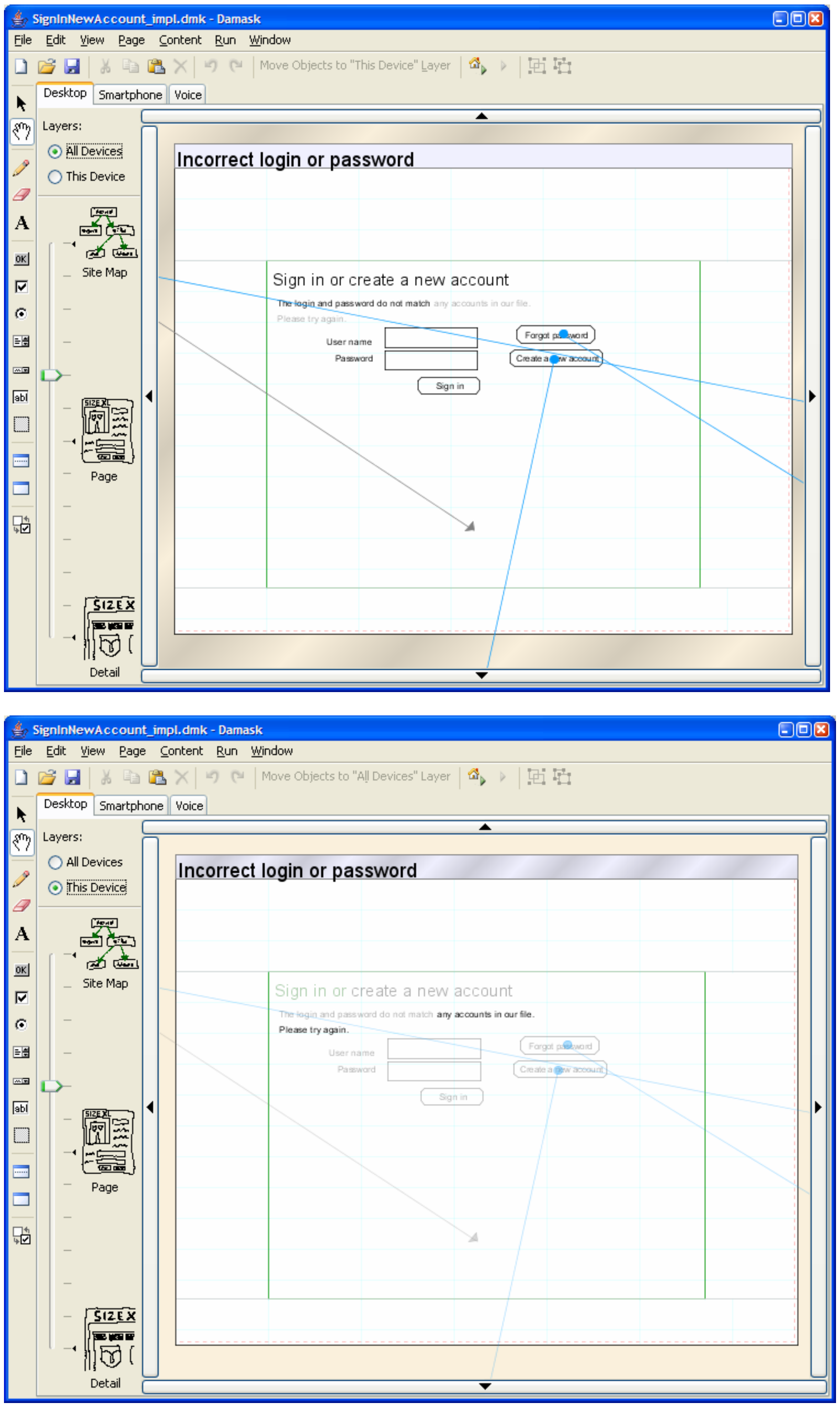


**FIGURE 4-25** The radio buttons used for changing the active layer.

Controls in the active layer are shown normally, while those in the inactive layer are grayed out. A page or form in the All Devices layer has a striped title bar when the This Device layer is active (see Figure 4-26). However, a page or form in the This Device layer is grayed out when the All Devices layer is active, for reasons explained later.



Incorrect login or password

Sign in or create a new account

The login and password do not match any accounts in our file. Please try again.

**FIGURE 4-26** The same page with *All Devices* as the current layer (above) and *This Device* as the current layer (below). The phrase "any accounts in our file. Please try again" is in the *This Device* layer, and all of the other objects are in the *All Devices* layer.

The background of the canvas is striped in the All Devices layer but plain in the This Device layer. The purpose is to be a more obvious visual indicator to designers of which layer they are currently viewing (see Figure 4-26).

There are some restrictions on placing a control in a page, depending on what the layer the control and the page are in:

- If a control is being created when the All Devices layer is active, it cannot be placed in a page on the This Device layer, since there is no corresponding page in the other devices in which the new control could be placed. This is why such a page is grayed out in this situation.
- If an arrow is being created in the All Devices layer, it cannot start from or end at a page on the This Device layer, for similar reasons.

Currently, Damask does not give feedback if the designer tries to do either of these actions, although future versions should.

Damask includes a button on the toolbar that allows the designer to change the layer of an object. When the designer selects objects in the All Devices layer, and presses the toolbar button labeled Move Object to This Device Layer, the selected objects are removed from the All Devices layer and placed in the This Device layer for the currently selected device. This also removes the objects from the other devices (see Figure 4-27). Conversely, when the designer selects objects in the This Device layer, and presses the toolbar button labeled Move Object to All Devices Layer, the selected objects are removed from the This Device layer and placed in the All Devices layer.

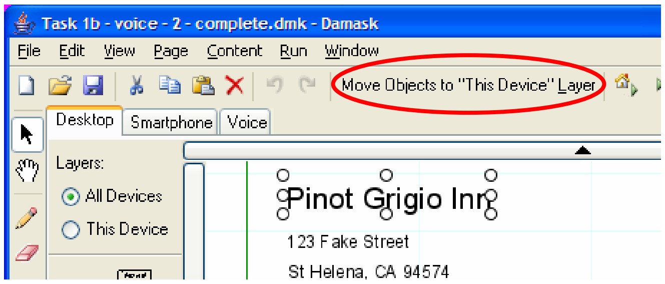

Task 1b - voice - 2 - complete.dmk - Damask
File Edit View Page Content Run Window
Move Objects to "This Device" Layer
Desktop
Smartphone
Voice
Layers:
All Devices
This Device
Pinot Grigio Inn
123 Fake Street
St Helena, CA 94574

**FIGURE 4-27** The "Move Object to This Device" button in the toolbar.

These buttons interact with template objects in the same way as normal objects. For example, if the designer selects an object in the template pane in the All Devices layer and presses Move Object to This Device Layer, the selected objects are removed from the All Devices layer and placed in the This Device layer for the currently selected device in the template. This has the effect of removing the template object from the other devices.

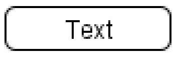
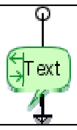
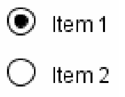
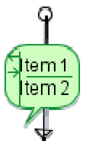
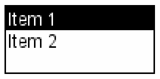
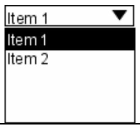
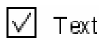
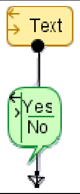
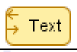
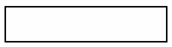
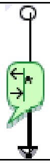
### 4.4.1 *Relationship between desktop/smartphone and voice UI elements*

When an element is placed in the All Devices layer, Damask needs to handle how to map desktop/smartphone elements to voice elements. In the case of pages and forms, it is straightforward. Creating a page in the desktop or smartphone view also creates a form in voice view, and vice versa. However, if a designer splits a page in, say, the smartphone view, *both* pages still correspond to the one desktop page and one voice form. This is because splitting a page is done typically for display purposes only; the logical grouping of the elements within the page or pages still remain the same.

It gets more complicated with controls. Adding a control in one device adds the control below all of the other controls in the other devices. In other words, the controls are simply laid out vertically in the order the designer added them. While this is a simple and predictable policy, it is not particularly smart. A more complete version of Damask would incorporate an inferencing algorithm to figure out which controls are grouped together, using techniques such as those in ScanScribe [161]. This would be coupled with a more robust layout algorithm, such as that in SUPPLE [55] or GADGET [48].

Also, Damask has a particular mapping between desktop/smartphone controls and voice controls. Table 4.1 lists the control that is created in the voice view when a control in the desktop or smartphone view is added.

**TABLE 4.1** Visual controls and their corresponding voice controls.

| Visual controls | Corresponding voice controls |
|---|---|
| Button  | Response  |
| Radio buttons  | Response  |
| List box  | |
| Drop-down box  | |
| Check box  | Prompt + Response  |
| Label Text | Prompt  |
| Text box  | Response  |

In addition, if there is a button immediately after a set of radio buttons, a list box, a drop-down box, a check box, or a text box, and there is an arrow starting from that button, then the corresponding voice response points to the same control as the arrow. Otherwise, it points to nothing (see Figure 4-28).

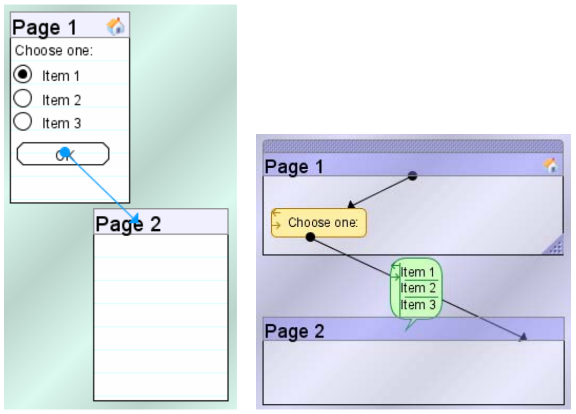
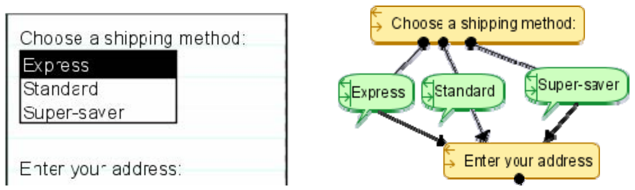


**FIGURE 4-28** A set of radio buttons and a button that links to another page, and the equivalent voice design. Note that the text of the radio buttons is the same as the text of the response, while the destination of the arrow from the OK button is the same as the destination of the response.

One might think that a check box, list box, drop-down box, or set of radio buttons would more naturally map to several responses, one response per choice. We did not choose this for the following reason. Suppose you create a desktop page with a list box followed by a label, and that inserting the list box results in several separate responses inserted in the voice view. That would result in the following design:



Now, suppose we take the "Express" response and point it to a different prompt. This results in the voice UI saying "Enter your address" if the user says "Standard" or "Super-saver," but the UI saying something else if the user says "Express." But this causes a dilemma for the desktop page: to remain consistent, the "Enter your address" label should be visible if "Standard" or "Super-saver" is selected but invisible when

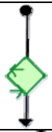
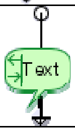
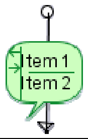
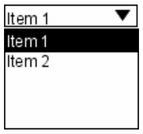
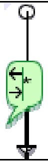
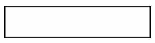
"Express" is selected. However, we do not want to complicate Damask's visual language to accommodate such a special condition, and it is doubtful that the designer would want such behavior.

Is there another way for Damask to keep the desktop page equivalent to the changed voice UI? Damask could split the page between the list box and the "Enter your address" label, add a "Next" button after the list box, and then create an arrow from the button, whose endpoint depends on the item selected in the list box. However, designers would not see all of these changes until they switched from the Voice view to the Desktop view. The drastic nature of these changes would almost certainly result in total confusion for the designers, who would be wondering what they did to cause that page to be split.

All of this is a consequence of the fact that we have one common model backing up desktop, smartphone, and voice designs.

Table 4.2 lists the control that is created in the desktop and smartphone views when a control in the voice view is added.

**TABLE 4.2** Voice controls and their corresponding visual controls.

| Voice control | | Corresponding visual control | |
|---|---|---|---|
| Response with no text (to connect two prompts without user input) |  | Nothing | |
| Response with one line of text |  | Button | |
| Response with two or more lines of text |  | Drop-down box |  |
| Response with * |  | Text box |  |
| Prompt | | Label | Text |

Furthermore, if the response points to another form or a control in another form, then an OK button will be added right after the corresponding desktop/smartphone control (unless that control is already a button or hyperlink). This is because voice responses encapsulate both receiving user input and acting on that input, whereas in a desktop/smartphone design, one control receives user input, while a button allows that input to be acted upon.

### 4.4.2 *Synchronizing text between desktop/smartphone UIs and voice UIs*

Often, the text between a desktop or smartphone UI and a voice UI is different, even if the general meaning is the same. For example, a desktop UI might have a check box labeled "Send Me a Copy," whereas a voice UI would ask the user, "Would you also like me to send a copy to you?" expecting a yes or no response. A designer could move the voice control to the This Device layer, but this would remove the check box from the other views. The designer then would have to create new check boxes in the This Device layers in the desktop and smartphone views, and these new check boxes would not be synchronized.

Instead, we have another mechanism for controlling the synchronization of text between voice and desktop/smartphone UI designs (see Figure 4-29). On the left side of each prompt and response, there is a two-arrow icon. The designer clicks on the icon to toggle between keeping the text for that prompt synchronized and unsynchronized. When the text is not synchronized, the icon is dimmed.

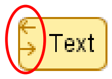
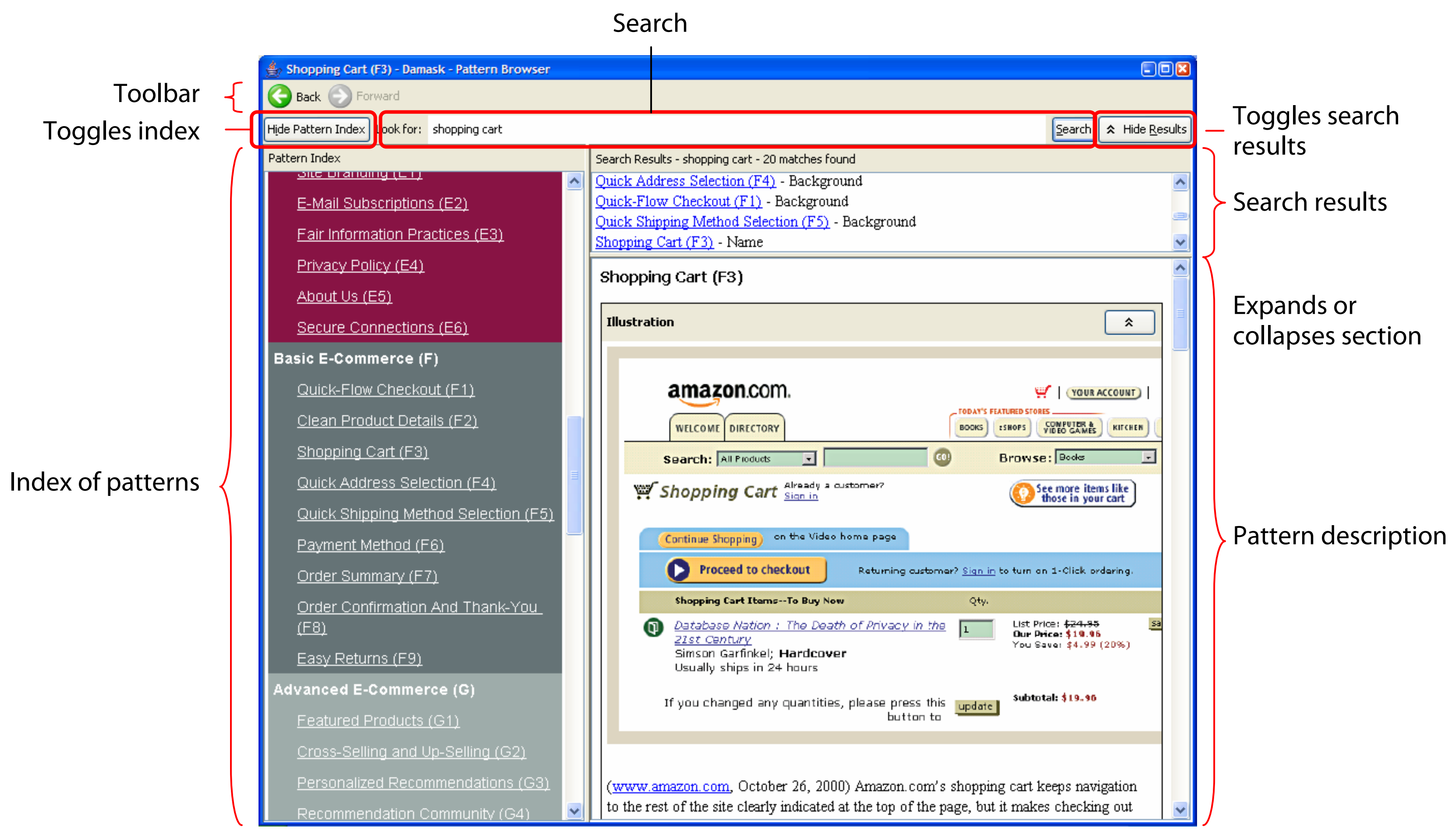


**FIGURE 4-29** The "synchronize text" indicator, circled on this prompt, is dimmed to indicate the prompt's text is not to be synchronized with other devices.

## 4.5 Patterns

Damask comes with a catalog of 90 patterns, from the book *The Design of Sites* [188]. To view the patterns, the designer goes to the View menu and chooses Pattern Browser. The Pattern Browser window consists of three areas (see Figure 4-30). An index of all patterns runs along the left-hand side. The top part includes a toolbar, a search text box, and a pane that contains search results. The main part of the window is the description of the pattern that the designer has selected from the index, as well as links to other patterns that are related.



Search Results - shopping cart - 20 matches found

Quick Address Selection (F4) - Background
Quick-Flow Checkout (F1) - Background
Quick Shipping Method Selection (F5) - Background
Shopping Cart (F3) - Name

E-Mail Subscriptions (E2)
Fair Information Practices (E3)
Privacy Policy (E4)
About Us (E5)
Secure Connections (E6)

**Basic E-Commerce (F)**
Quick-Flow Checkout (F1)
Clean Product Details (F2)
Shopping Cart (F3)
Quick Address Selection (F4)
Quick Shipping Method Selection (F5)
Payment Method (F6)
Order Summary (F7)
Order Confirmation And Thank-You (F8)
Easy Returns (F9)

**Advanced E-Commerce (G)**
Featured Products (G1)
Cross-Selling and Up-Selling (G2)
Personalized Recommendations (G3)
Recommendation Community (G4)

Shopping Cart (F3)

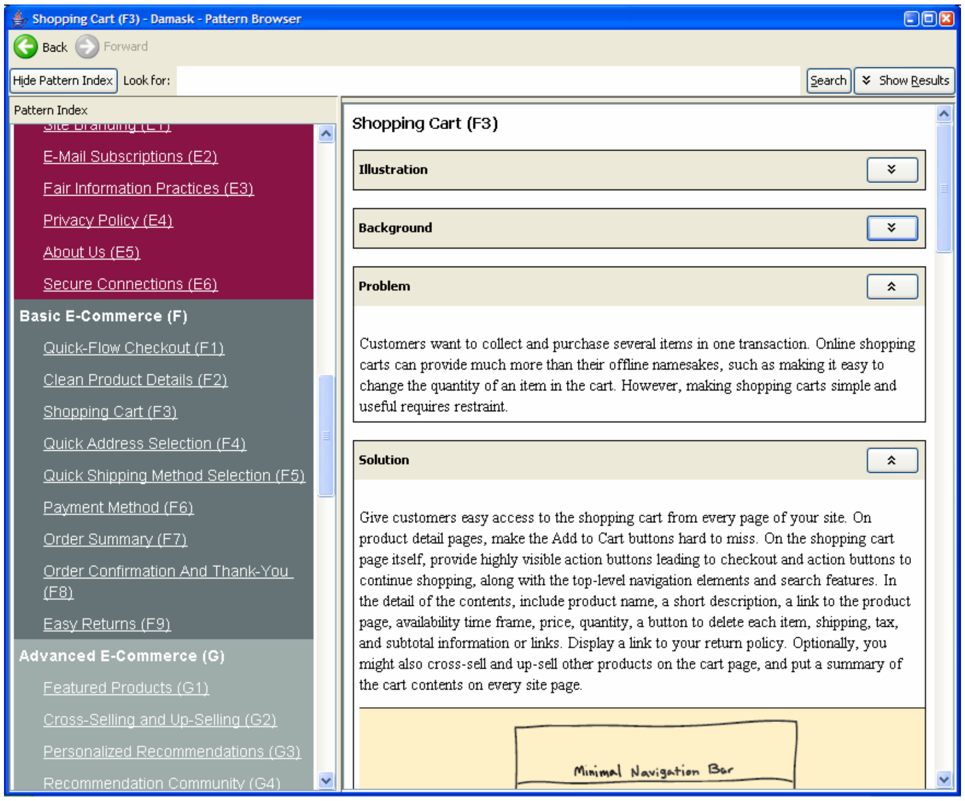
(www.amazon.com, October 26, 2000) Amazon.com's shopping cart keeps navigation to the rest of the site clearly indicated at the top of the page, but it makes checking out

**FIGURE 4-30** The pattern browser.

Each pattern has five parts, which is similar to the structure of patterns found in several publications such as *A Pattern Language* [4] and *The Design of Sites* [188]:

- name
- background and sensitizing image
- problem
- solution
- related patterns

Each section can be expanded or collapsed by clicking on the associated button with a chevron (see Figure 4-31).



Customers want to collect and purchase several items in one transaction. Online shopping carts can provide much more than their offline namesakes, such as making it easy to change the quantity of an item in the cart. However, making shopping carts simple and useful requires restraint.

Give customers easy access to the shopping cart from every page of your site. On product detail pages, make the Add to Cart buttons hard to miss. On the shopping cart page itself, provide highly visible action buttons leading to checkout and action buttons to continue shopping, along with the top-level navigation elements and search features. In the detail of the contents, include product name, a short description, a link to the product page, availability time frame, price, quantity, a button to delete each item, shipping, tax, and subtotal information or links. Display a link to your return policy. Optionally, you might also cross-sell and up-sell other products on the cart page, and put a summary of the cart contents on every site page.

**FIGURE 4-31** The pattern browser with the Illustration and Background sections collapsed.

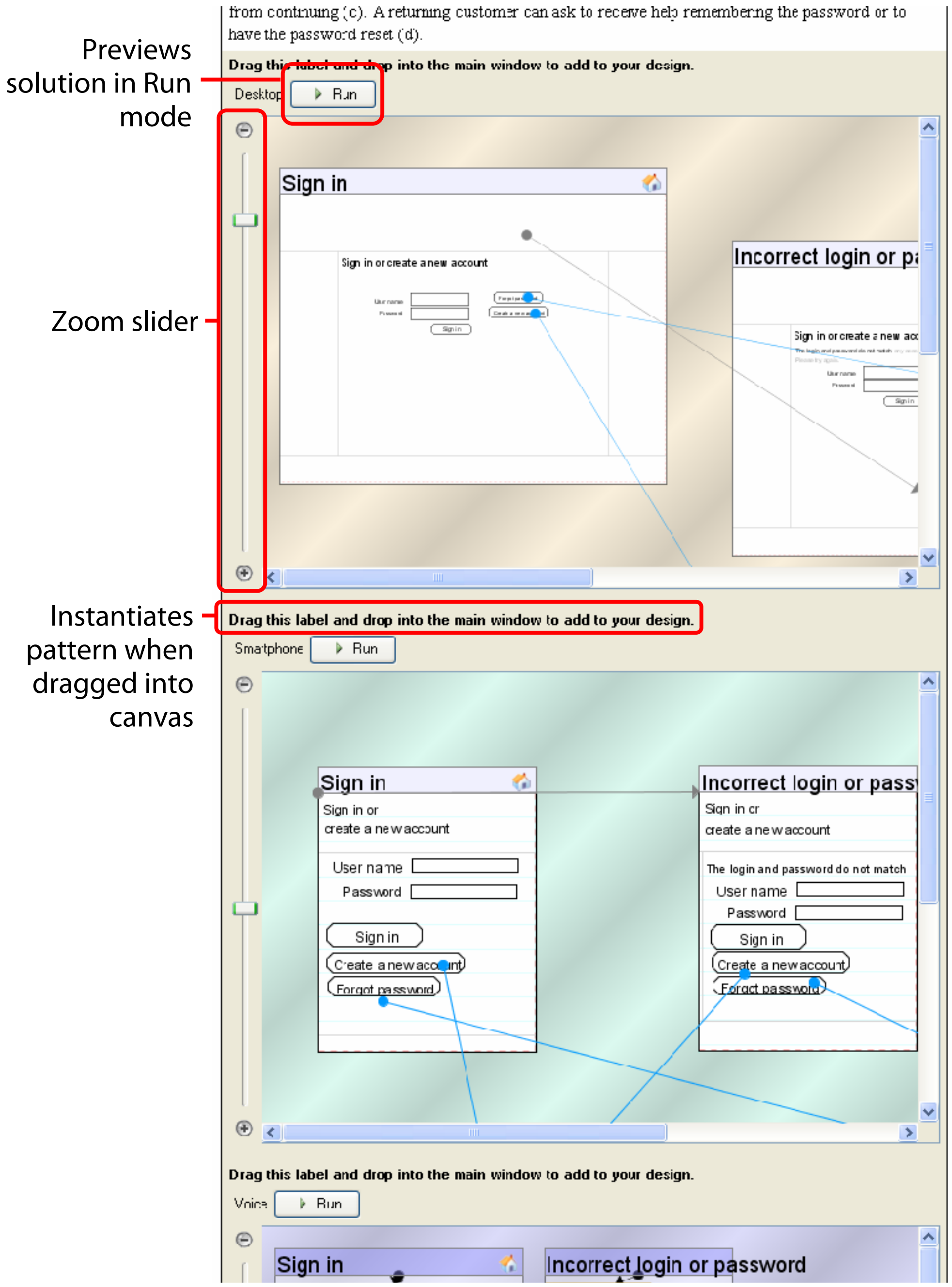
For eleven of the patterns, the Solution section contains generalized solutions for that pattern, with one solution for each device supported by Damask (see Table 4.3).

The patterns that were implemented were low level enough to have a meaningful set of pages and controls (as opposed to a pattern such as SITE ACCESSIBILITY, which is more conceptual). Each solution is simply a Damask design, with the same structure as a UI design that a designer created in Damask (see Figure 4-32). Since the pattern already takes care of issues such as putting pages and controls in the appropriate layer, designers using it could potentially save a lot of time, if it fit their needs.

We picked the patterns listed in Table 4.3 to implement, primarily because they formed a cohesive group around which we could create an experiment for evaluation, which is described in Chapter 0. Damask-based solutions would be suitable for more than the eleven implemented, but they were not done due to time constraints.

**TABLE 4.3** The patterns for which Damask solutions were implemented.

| | |
|---|---|
| B8 | Category Pages |
| C1 | Homepage Portal |
| F1 | Quick Flow Checkout |
| F2 | Clean Product Details |
| F3 | Shopping Cart |
| F4 | Quick Address Selection |
| F5 | Quick Shipping Method Selection |
| F6 | Payment Method |
| F7 | Order Summary |
| F8 | Order Confirmation and Thank You |
| H2 | Sign In/New Account |


from continuing (c). A returning customer can ask to receive help remembering the password or to have the password reset (d).

Drag this label and drop into the main window to add to your design.

Drag this label and drop into the main window to add to your design.

Drag this label and drop into the main window to add to your design.

Previews solution in Run mode
Zoom slider
Instantiates pattern when dragged into canvas
Desktop
Run
Sign in
Incorrect login or pa
Sign in or create a new account
Smartphone
Run
Sign in
Sign in or create a new account
User name
Password
Sign in
Create a new account
Forgot password
Incorrect login or pass
Sign in or create a new account
The login and password do not match
User name
Password
Sign in
Create a new account
Forgot password
Voice
Run
Sign in
Incorrect login or password

**FIGURE 4-32** The pattern browser showing solutions of the SIGN-IN/NEW ACCOUNT pattern for the desktop and smartphone, which can be added to a Damask design.

For each device-specific solution, designers can zoom in and out using the zoom slider, and preview the solution in Run mode by clicking the Run button. To add a pattern instance to a Damask design, the designer drags the label that starts with

Drag this label and drop into… directly onto the canvas in the main window. Damask instantiates the pattern and adds the instance to all devices, regardless of the current layer in the main window. This means that it does not matter which device solution the designer drags and drops.

A designer can also merge a new pattern instance with an existing page, by dragging the aforementioned label on top of an existing page. This merges that existing page with the "home page" of the pattern, denoted with a home icon in the Pattern Browser. Only one existing page can be merged with a pattern page.

The pattern instance is surrounded by a blue dotted box plus the name of the pattern in transparent blue text (see Figure 4-33). It automatically resizes itself if the elements that make up the pattern instance are moved or deleted.

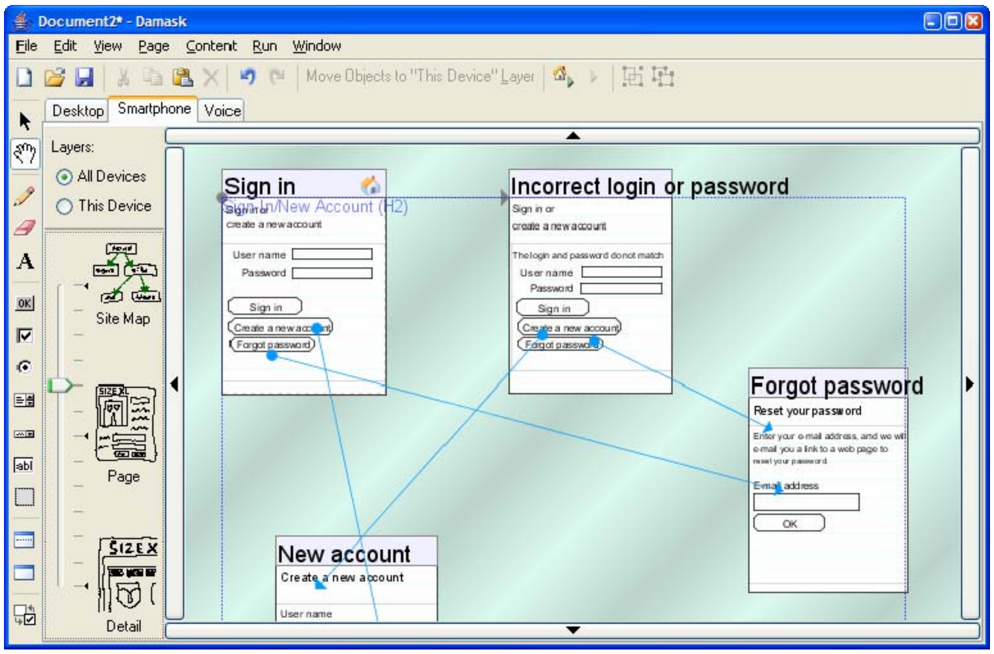


**FIGURE 4-33** An instance of the SIGN-IN/NEW ACCOUNT pattern in a UI design.

Designers can now change the pattern instance in any way they want, including adding, deleting, and moving around controls, arrows, or pages. The original pattern is unaffected. The relationship between a pattern solution and a pattern instance is similar to a prototype-instance relationship in prototype-based object-oriented languages such as Self [172] and JavaScript [44], and user interface management systems (UIMSs) like Garnet and Amulet [136].

## 4.6 Summary

Damask's design reflects the concerns that cross-device user interface designers had expressed in our interviews and an evaluation of our initial cross-device UI design tool. Its use of model-based UI principles and layers allow designers quickly see how changes they make to one device UI manifest themselves in the other device UIs. Layers let designers control which parts of their design get retargeted to other devices and which do not. Designers can create common elements in their UI designs using Damask's templates feature. Finally, design patterns include prebuilt UI design fragments that can have very different interaction flows depending on the device, which facilitates the creation of cross-device user interface designs that are optimized for each target device.

# 5 Architecture and Implementation of Damask

Damask is written in Java 2 Standard Edition version 1.4. Besides the standard Java libraries, it uses two other libraries. SATIN [71] is a library for pen-based applications. Damask uses it specifically for recognizing whether ink strokes should be grouped. Piccolo [21] is a 2D graphics library for Java, which Damask uses for the canvas area.

The architecture of Damask is based on the model-view-controller (MVC) pattern [93]. MVC decouples the data model from the user interface, allowing multiple views and types of user interaction to manipulate the same data model. In Damask's case, the view is the UI of Damask that was described in Chapter 4, while the model is the abstract model of the user interface being designed by the designer. When the user interface receives an event from the designer, such as a key press or a mouse move, the UI calls a method in the controller, which modifies the model. When the model is changed, it fires events. The view objects listens for events and adjusts the user interface appropriately.

We will first discuss the model and the basic architecture, and then talk in more detail about how the model relates to the Desktop and Smartphone views and the Voice view.

## 5.1 Implementation of Model Objects

In this section we will talk about how the model backing the user interface is implemented.

### 5.1.1 *Interaction graph and interaction elements*

`InteractionElement` is the base class for every element within a model, including dialogs, components within dialogs, and connections (see Figure 5-1). (Actually, for implementation purposes, `InteractionElement` is an interface which only one class implements, `AbstractInteractionElement`. But to simplify this discussion, we will consider `InteractionElement` and `AbstractInteractionElement` to be equivalent.) Instances of `InteractionElement` are organized within a scenegraph. The root of the scenegraph is an instance of the `InteractionGraph` class, which contains three lists, `Dialogs`, `Connections`, and `PatternInstances`, all of which are subclasses of `InteractionElement` (see Figure 5-2).

Devices are represented by instances of `DeviceType`. A `DeviceType` contains information like a name and screen dimensions (for visual devices). Damask has three types: `DeviceType.DESKTOP`, `DeviceType.SMARTPHONE`, and `DeviceType.VOICE`. A fourth type, `DeviceType.ALL`, represents any device.

`InteractionElement` contains methods that return the bounding box and the affine transform that controls its position. These methods are parameterized by a device, represented by an instance of `DeviceType`. For example, let `l` be a label and `e` be its corresponding `InteractionElement` instance. Suppose `l` is in the All Devices layer, and has a size of 100×20 pixels and a location of (40, 30) in every device. Then, `e.getBounds(device)` returns a rectangle of (0, 0, 100, 20) and `e.getTransform(device)` will return a translation transform of (40, 30), for *all* devices.

In contrast, now suppose `l` is in the This Device layer of the Desktop view, and has a size of 80×20 pixels and a location of (70, 50). Therefore, `e.getBounds(DeviceType.DESKTOP)` returns a rectangle of (0, 0, 80, 20) and `e.getTransform(DeviceType.DESKTOP)` returns a translation transform of (70, 50). However, because it is in the This Device layer, `l` does not exist in the other two devices. Therefore, both `e.getBounds(device)` and `e.getTransform (device)` return null, if `device` equals `DeviceType.SMARTPHONE` or `DeviceType.VOICE`.

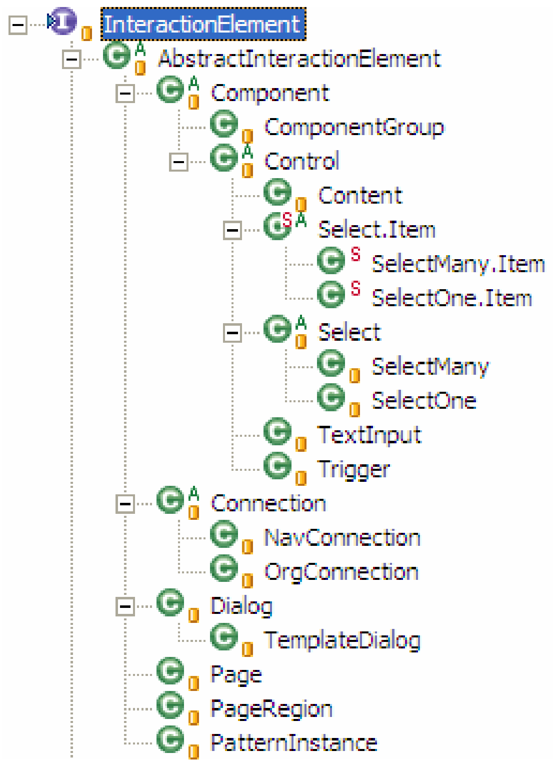
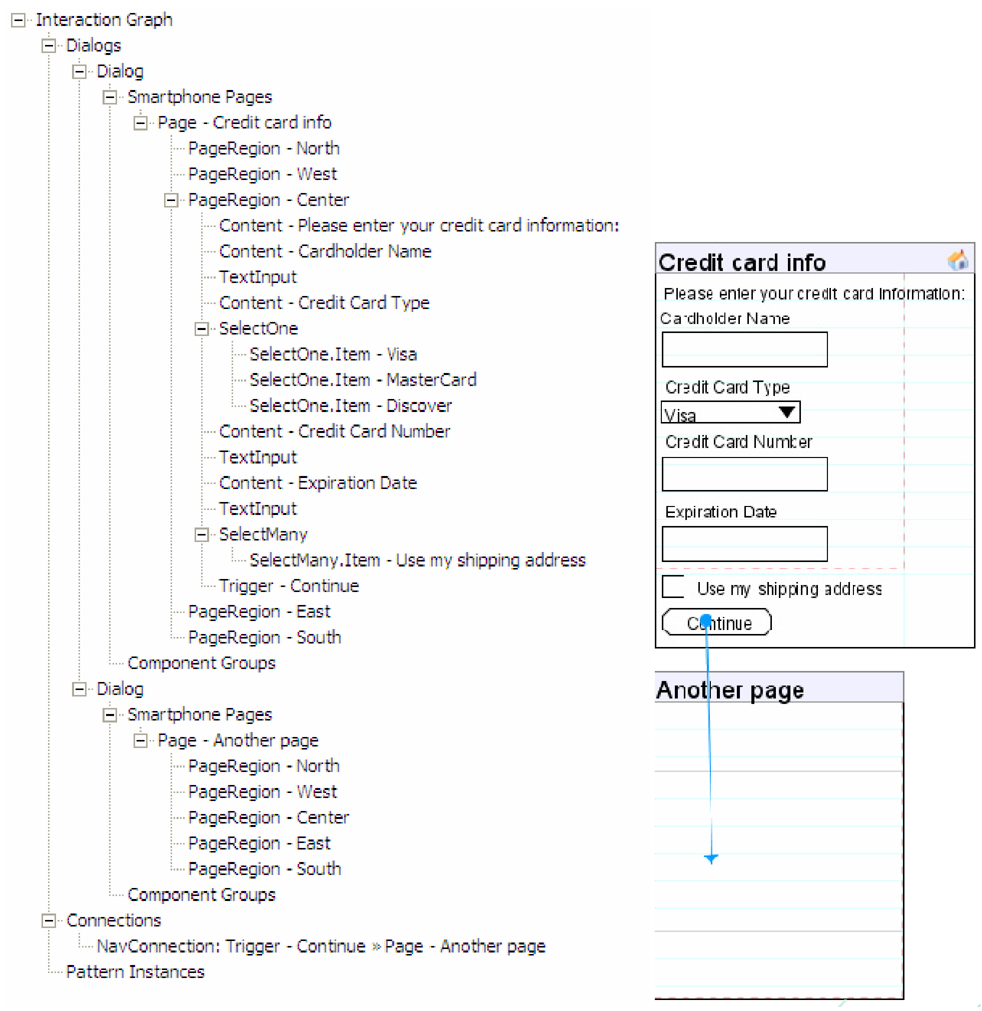


**FIGURE 5-1** The class hierarchy for `InteractionElement`. For implementation purposes, `InteractionElement` is actually an interface, but it has only one implementation, `AbstractInteractionElement`.



```
⊟ Interaction Graph
  ⊟ Dialogs
    ⊟ Dialog
      ⊟ Smartphone Pages
        ⊟ Page - Credit card info
            PageRegion - North
            PageRegion - West
          ⊟ PageRegion - Center
              Content - Please enter your credit card information:
              Content - Cardholder Name
              TextInput
              Content - Credit Card Type
            ⊟ SelectOne
                SelectOne.Item - Visa
                SelectOne.Item - MasterCard
                SelectOne.Item - Discover
              Content - Credit Card Number
              TextInput
              Content - Expiration Date
              TextInput
            ⊟ SelectMany
                SelectMany.Item - Use my shipping address
              Trigger - Continue
            PageRegion - East
            PageRegion - South
        Component Groups
    ⊟ Dialog
      ⊟ Smartphone Pages
        ⊟ Page - Another page
            PageRegion - North
            PageRegion - West
            PageRegion - Center
            PageRegion - East
            PageRegion - South
        Component Groups
  ⊟ Connections
      NavConnection: Trigger - Continue » Page - Another page
    Pattern Instances
```

**FIGURE 5-2** The model (left) for a typical smartphone user interface design (right).

### 5.1.2 *Layers*

The user interface concept of layers is implemented indirectly. Each `InteractionElement` object has a `deviceType` property that is set when the object is constructed. If the `deviceType` is a single device, such as `DeviceType.DESKTOP`, then the element's view object is in the This Device layer of the corresponding device's view (in this case, the Desktop view). If the `deviceType` is `DeviceType.ALL`, then the element's view object is in the All Devices layer in every device-specific view.

### 5.1.3 *Dialogs and templates*

When a designer creates a page in the Desktop or Smartphone view, an instance of `Dialog` is created in the model. A `Dialog` contains a list of instances of `Page`s for each device type (see Figure 5-3). For example, when a designer creates a page on the All Devices layer, the new page's `Dialog` has three lists, each containing one `Page`. In contrast, when a designer creates a page on the This Device layer on the Desktop view, only the list of desktop-specific `Page`s in the `Dialog` contains an instance of `Page`, while the other two lists are empty.

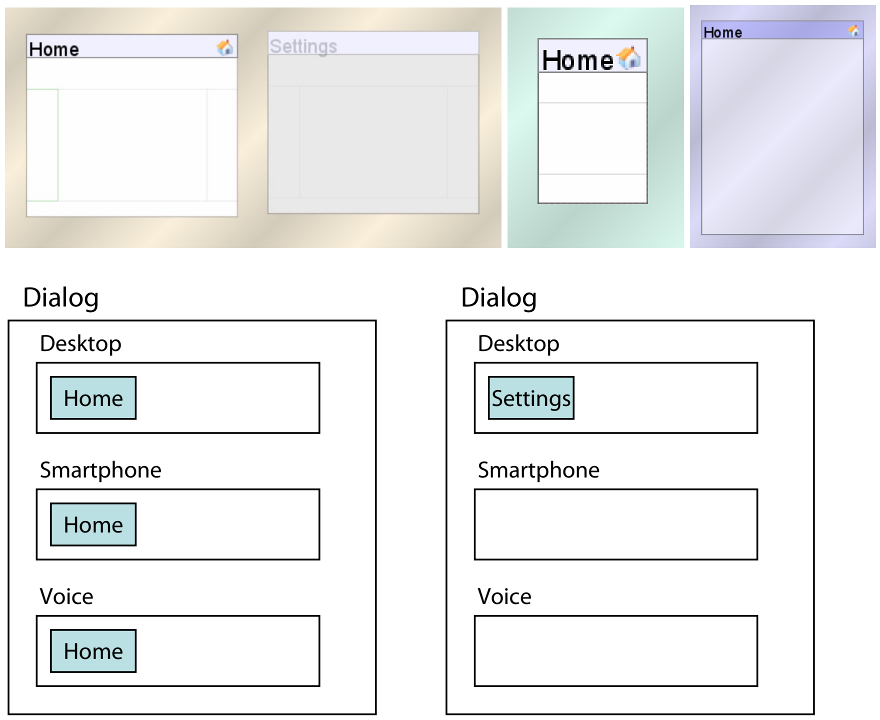


**FIGURE 5-3** A page that is on the All Devices layer ("Home") and a page that is on the This Device layer in Desktop view ("Settings"), along with the associated model.

Normally, for a desktop or smartphone design, there is one `Page` for each supported device within a `Dialog`. However, splitting a page results in the model adding a new `Page` to the appropriate list in the `Dialog`, with the appropriate contents moved from the original `Page` to the new `Page`. So a desktop or smartphone `Page` in the model represents a desktop or smartphone page in the user interface, and a `Dialog` groups together pages that have been split. `Dialog`s themselves are invisible in the user interface.

Each `Page` contains five `PageRegions`, which correspond to the page regions seen in Figure 4-4. Each `PageRegion` contains a list of zero or more user interface `Control`s (see Figure 5-2).

A template in Damask is implemented by a `TemplateDialog`, which is a subclass of `Dialog`. A `Dialog` object contains a list of `TemplateDialog` objects that are being "applied" to the dialog.

### 5.1.4 Components

`Control` is the abstract base class of all user interface controls in Damask, which are loosely based on XForms [194] controls. Damask supports the following `Control`s:

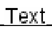
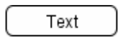
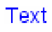
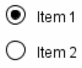
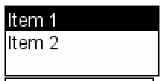
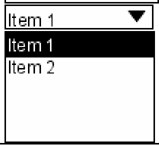
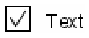
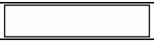
- `Content`: A piece of content, either ink, text, or an image
- `Trigger`: Lets the user transition from one page to another
- `SelectOne`: Allows the user to select one item from a collection
- `SelectMany`: Allows the user to select more than one item from a collection
- `TextInput`: Allows the user to input any text

`SelectOne` and `SelectMany` controls are actually composite `Control`s, each consisting of a list of `SelectOne.Item`s and `SelectMany.Item`s, respectively. Each `Item` is also a `Control`. `Trigger`, `SelectOne.Item`, and `SelectMany.Item` each contain an instance of `Content` that defines the text, image, or sketch that they display.

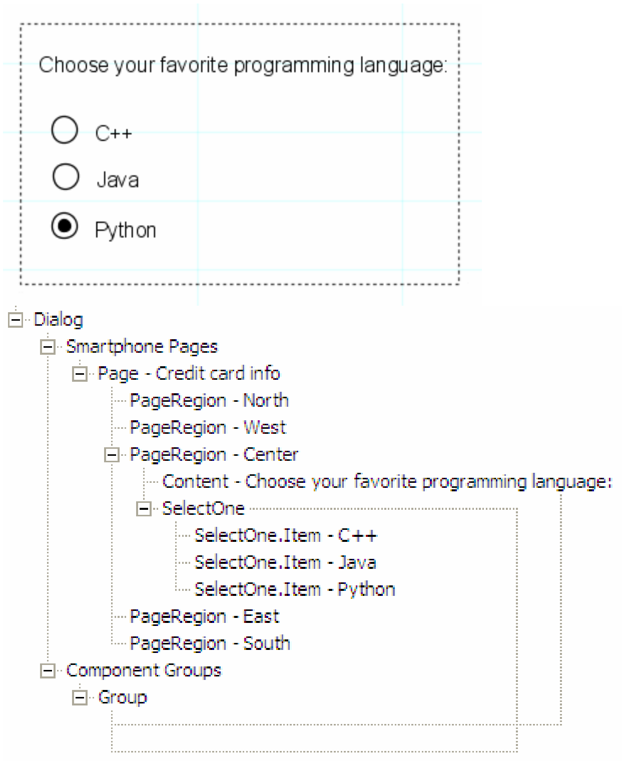
Table 5.1 shows the correspondence between controls in the desktop and smartphone views and `Control`s in the model. Both `Trigger` and `SelectOne` have a

Style property that determines how it is rendered in the Desktop and Smartphone views.

**TABLE 5.1** Views of visual controls and their corresponding model objects.

| View in desktop/smartphone | | Control in model |
|---|---|---|
| Label |  | Content |
| Button |  | Trigger |
| Hyperlink |  | |
| Radio buttons |  | SelectOne<br>SelectOne.Item<br>SelectOne.Item |
| List box |  | |
| Drop-down box |  | |
| Check box |  | SelectMany<br>SelectMany.Item |
| Text box |  | TextInput |

Each Dialog also contains a list of ComponentGroups. A ComponentGroup contains Controls or other ComponentGroups. A ComponentGroup belongs to a Dialog, instead of a Page, so that if a Page is split, Controls that are in more than one Page can still be in the same ComponentGroup. The idea behind ComponentGroup was to allow groups of controls to be named and manipulated together, although this idea has not been fully explored in the current version of Damask. Currently, ComponentGroups are realized as panels in the user interface (see Figure 5-4).


Choose your favorite programming language:

- ○ C++
- ○ Java
- ● Python

```
Dialog
  Smartphone Pages
    Page - Credit card info
      PageRegion - North
      PageRegion - West
      PageRegion - Center
        Content - Choose your favorite programming language:
        SelectOne
          SelectOne.Item - C++
          SelectOne.Item - Java
          SelectOne.Item - Python
      PageRegion - East
      PageRegion - South
  Component Groups
    Group
```

**FIGURE 5-4** A panel with a label and three radio buttons, and the backing model. Note how the group only points to controls; it does not contain them.

### *5.1.5 Connections*

The `Connection` class is the abstract base class of all connections in Damask. There are two types of connections. An organizational connection (`OrgConnection`) connects `Pages` together. A navigational connection (`NavConnection`) links a `Control` in one `Page` to another `Page`.

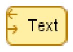
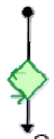
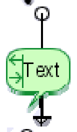
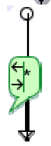
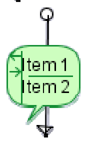
In desktop and smartphone views, `Connections` are rendered as arrows. `OrgConnections` are shown as gray arrows, while `NavConnections` are shown as blue arrows.

A `NavConnection` also has an event type which needs to be triggered by the end-user for the connection to be activated. There are five event types defined: `Select`, `Invoke`, `SecondaryInvoke`, `HoverOver`, and `HoverOff`. Currently, designers can only create connections with `Invoke` events attached, which correspond to clicking on a hyperlink or button. `Select` is meant for selecting an item without triggering it, such as from a list. `SecondaryInvoke` is for right-clicking on a GUI or tap-and-hold on a Pocket PC. `HoverOver` and `HoverOff` are for moving the mouse cursor over or off of an object.

## 5.1.6 Correspondence between model and voice view

The correspondence between view objects in voice UI designs and their corresponding controls in the model is more complicated than in the case of desktop and smartphone UI designs. In addition to the basic properties inherited from `InteractionElement`, `Controls` also have optional voice-specific properties, such as the bounds and text of the voice prompt, or the coordinates of the response arrow. Table 5.2 shows the correspondence.

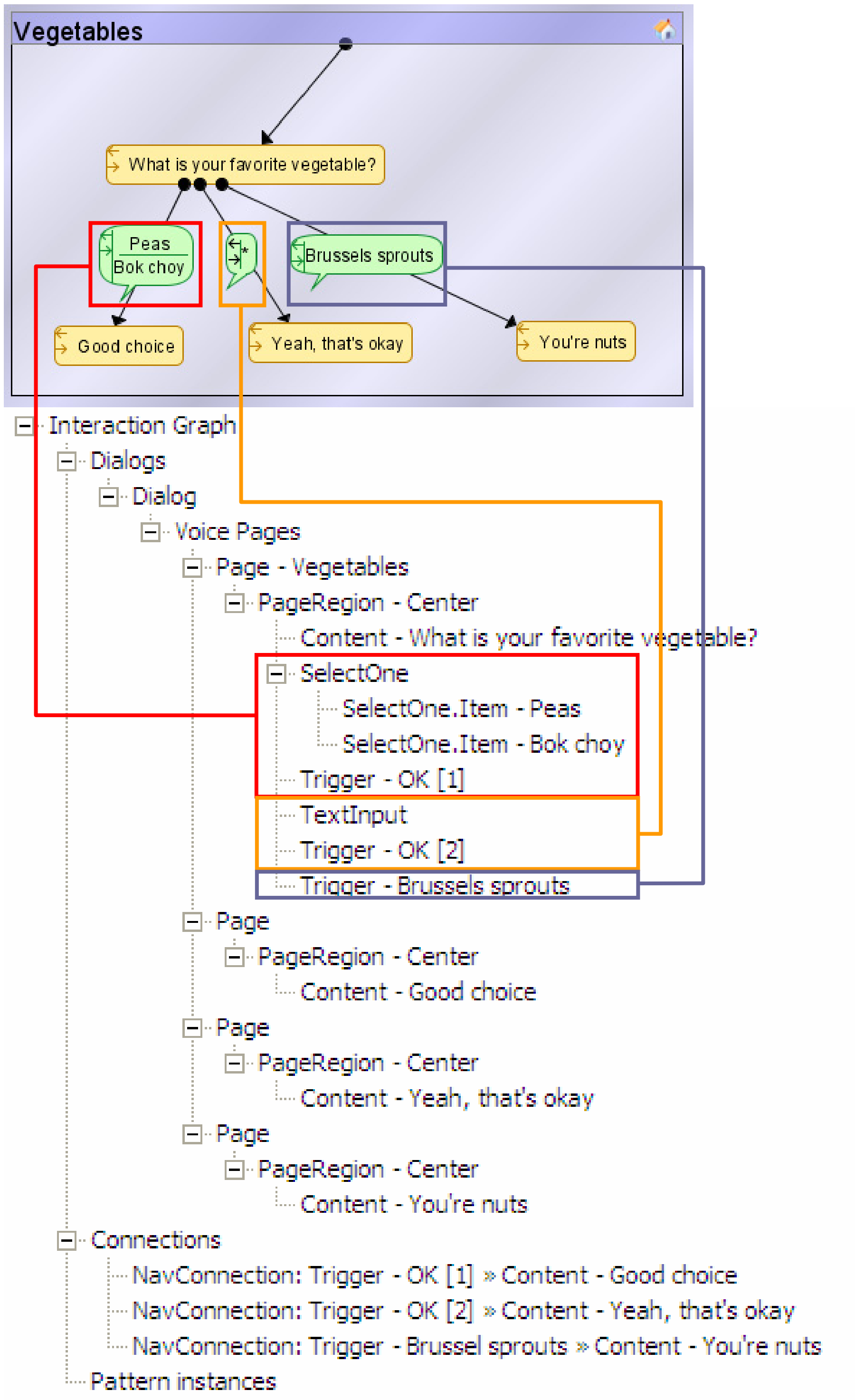
**TABLE 5.2** Views of voice controls and their corresponding model objects.

| View in voice | | Control in model |
|---|---|---|
| Prompt |  | Content with “Text” as its text |
| Response with no text |  | Trigger with no content (invisible in desktop/smartphone) |
| Response ”Text” |  | Trigger with “Text” as its content |
| Response with * |  | TextInput, followed by Trigger with content “OK” |
| Response with two or more lines of text |  | SelectOne with 2 choices (Item 1, Item 2), followed by Trigger with content “OK” |

Note that *every* response created by the designer has a Trigger associated with it. This is because voice responses encapsulate both receiving user input (represented by a control like a `TextInput` or `SelectOne`) and acting on that input (represented by a `Trigger`).

Now we will explain how these controls are linked together to form a voice user interface. In a desktop or smartphone interface, the controls within a page do not have any dependencies with each other, so a list of Controls within a Page is sufficient. However, in a voice UI, a flat list is not enough structure for a network of prompts and responses within a form.

Instead, we group together the `Control`s that represent a prompt and all responses that originate from that prompt, and place them into one `Page`. We then use `NavConnection`s to link the `Trigger`s in one `Page` to another `Page`. This allows us to form a network of prompts and responses. Figure 5-5 shows an example.



```
Interaction Graph
  Dialogs
    Dialog
      Voice Pages
        Page - Vegetables
          PageRegion - Center
            Content - What is your favorite vegetable?
            SelectOne
              SelectOne.Item - Peas
              SelectOne.Item - Bok choy
            Trigger - OK [1]
            TextInput
            Trigger - OK [2]
            Trigger - Brussels sprouts
        Page
          PageRegion - Center
            Content - Good choice
        Page
          PageRegion - Center
            Content - Yeah, that's okay
        Page
          PageRegion - Center
            Content - You're nuts
  Connections
    NavConnection: Trigger - OK [1] » Content - Good choice
    NavConnection: Trigger - OK [2] » Content - Yeah, that's okay
    NavConnection: Trigger - Brussel sprouts » Content - You're nuts
  Pattern instances
```

**FIGURE 5-5** A voice UI in Damask and its corresponding model. Empty page regions are omitted for brevity.

### 5.1.7 Patterns and pattern instances

The basic description of each pattern is stored in a file in a format called the Pattern Language Markup Language (PLML) [46], which is based on XML. The background image and Damask-based solution are stored in separate files, referenced by the PLML file. When Damask is started, a stub entry for each pattern is placed into Damask's pattern library. The stub is replaced with the actual pattern when the user views the pattern in the Pattern Explorer. This is to save memory.

Damask has several classes to manage patterns. `Pattern` represents a pattern; it has properties for subsections such as `Name`, `Background`, and `Solution`. `PatternSolution` represents the Damask solution that some of the patterns include; it is a subclass of `InteractionGraph`. `PatternInstance` represents an instance of a pattern within a Damask UI design. A `PatternInstance` object does not contain the elements that make up the pattern instance, since members of an instance can cut across pages and groups. Instead, it points to the elements within the main `InteractionGraph` that represents the design.

To add an instance of a pattern within a design:

1 The pattern's solution, an instance of `PatternSolution`, is cloned. This of course also clones the `Dialog`s, `Connection`s, and `PatternInstance`s within the `PatternSolution`.

2 The `Dialog`s, `Connection`s, and `PatternInstance`s in the cloned `PatternSolution` are then removed from the `PatternSolution` and merged directly into the `InteractionGraph` object representing the original design. The cloned `PatternSolution` is disposed.

3 Finally, a new `PatternInstance` object, that references the `Dialogs`, `Connections`, and `PatternInstances` that were moved in Step 2, is created and added to the `InteractionGraph`.

See Figure 5-6.

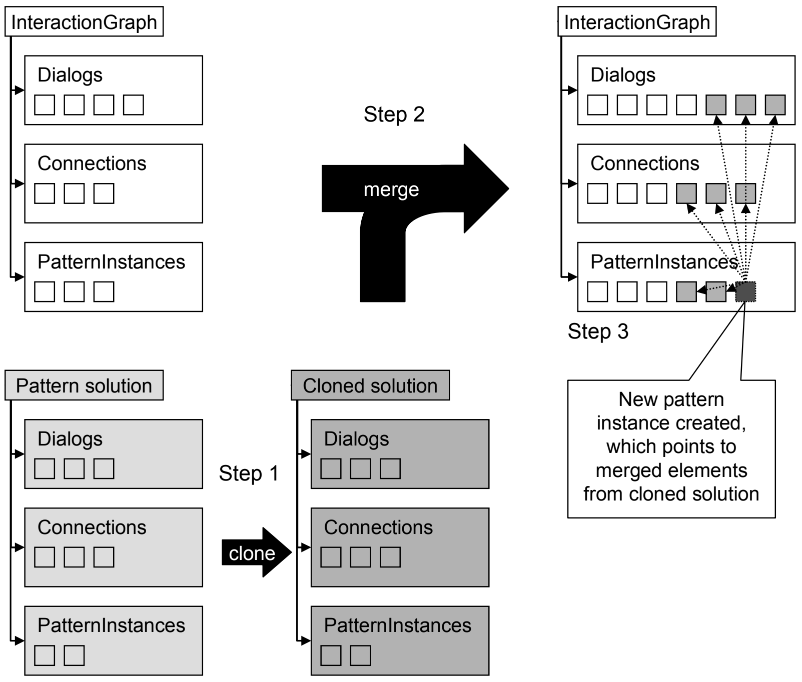


**FIGURE 5-6** Instantiating a pattern.

## 5.2 Implementation of View Objects

Piccolo [21] is a 2D graphics library for Java. It uses a scenegraph to structure the objects drawn in a canvas, and adds high-level support for zooming, animation, event handling, picking objects, and so on. Damask uses Piccolo heavily, subclassing many of its classes for Damask's user interface. All of Piccolo's classes begin with a capital P.

### 5.2.1 *Canvas*

Each device-specific canvas in Damask is an instance of `DamaskCanvas`, which is a subclass of `PCanvas`. `PCanvas` is the bridge that allows Piccolo graphics to be embedded in a Swing interface. The `DamaskCanvas` object views an instance of a subclass of `PLayer`. (This should not be confused with the concept of layers described in Section 4.4.) The actual graphical objects that make up a Damask design reside within the `PLayer`. Desktop and smartphone designs reside on a `VisualLayer`, while voice designs are on a `VoiceLayer`. The zoom slider and the panning buttons in the UI (see Figure 4-1) zoom and pan the canvas by changing the scale and translation transform attached to the `DamaskCanvas`'s camera.

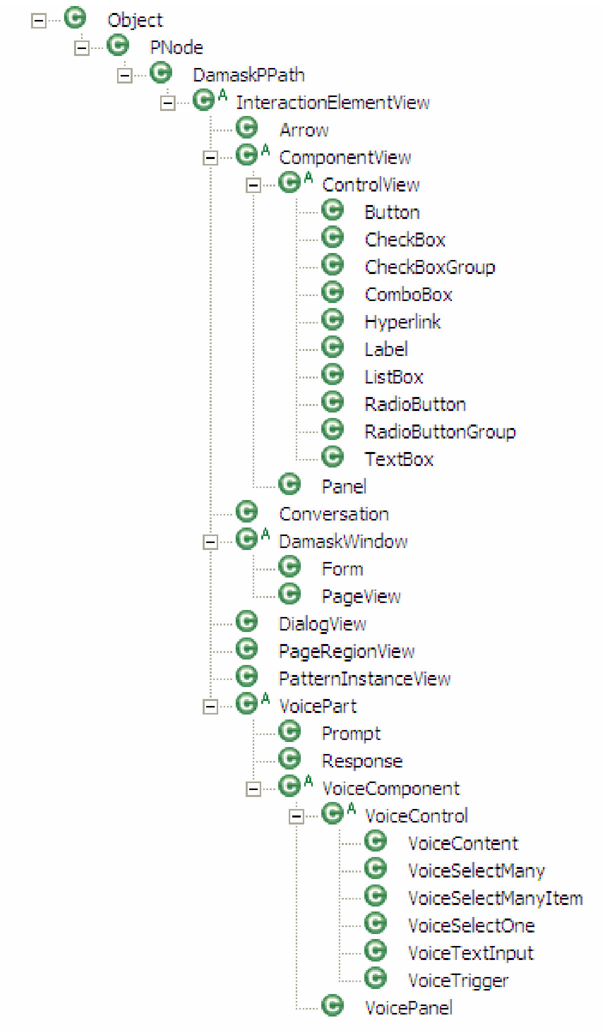
### 5.2.2 Views of interaction elements

Just like `InteractionElement` is the base class of every model class, `InteractionElementView` is the base class of every view class. `InteractionElementView` is a subclass of `PNode`, which is the base class in Piccolo for all graphical objects. A `PNode` object keeps track of its bounding box, has an affine transform attached to it that affects its scale and position, and can contain other `PNodes`. One can subclass `PNode` to override its drawing behavior. For example, `PPath` draws an arbitrary graphical path that is passed into its constructor.

`PNode`s can also handle UI events, such as keyboard and mouse events, by attaching an event listener. When a UI event occurs, Piccolo dispatches the event to the node in the scenegraph which has the focus, typically a leaf node. If that node has an event listener that can handle the event, it will be called. Otherwise, it will percolate up the scenegraph until it reaches a node that can handle it, or no node handles the event and it is dropped.

Subclasses of `InteractionElementView` handle views for dialogs, pages, controls, connections, and pattern instances. Figure 5-7 shows the class hierarchy for

`InteractionElementView`. We will explain these classes in more detail in the following sections.

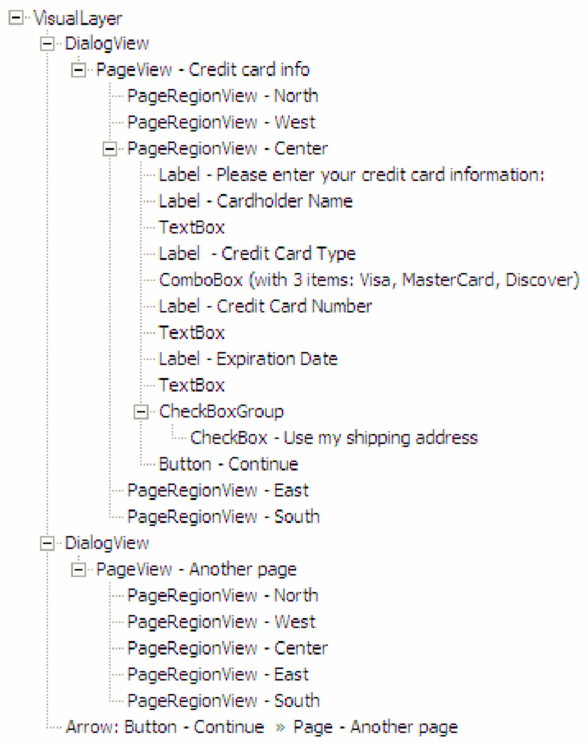


```
Object
  PNode
    DamaskPPath
      InteractionElementView A
        Arrow
        ComponentView A
          ControlView A
            Button
            CheckBox
            CheckBoxGroup
            ComboBox
            Hyperlink
            Label
            ListBox
            RadioButton
            RadioButtonGroup
            TextBox
          Panel
        Conversation
        DamaskWindow A
          Form
          PageView
        DialogView
        PageRegionView
        PatternInstanceView
        VoicePart A
          Prompt
          Response
          VoiceComponent A
            VoiceControl A
              VoiceContent
              VoiceSelectMany
              VoiceSelectManyItem
              VoiceSelectOne
              VoiceTextInput
              VoiceTrigger
            VoicePanel
```

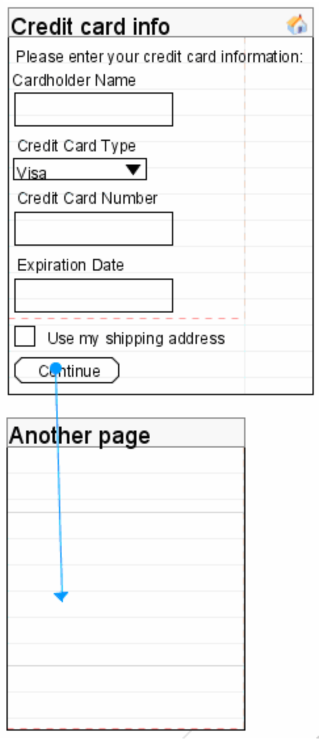
**FIGURE 5-7** The class hierarchy for `InteractionElementView`.

### 5.2.3 *Pages and controls*

For almost every model object that represents pages and controls, there is an equivalent view object for desktop/smartphone UIs and voice UIs. Figure 5-8 shows a typical runtime structure, for the same UI that is shown in Figure 5-2.



```
VisualLayer
  DialogView
    PageView - Credit card info
      PageRegionView - North
      PageRegionView - West
      PageRegionView - Center
        Label - Please enter your credit card information:
        Label - Cardholder Name
        TextBox
        Label  - Credit Card Type
        ComboBox (with 3 items: Visa, MasterCard, Discover)
        Label - Credit Card Number
        TextBox
        Label - Expiration Date
        TextBox
        CheckBoxGroup
          CheckBox - Use my shipping address
        Button - Continue
      PageRegionView - East
      PageRegionView - South
  DialogView
    PageView - Another page
      PageRegionView - North
      PageRegionView - West
      PageRegionView - Center
      PageRegionView - East
      PageRegionView - South
  Arrow: Button - Continue  »  Page - Another page
```



Credit card info

Please enter your credit card information:

Cardholder Name

Credit Card Type

Visa

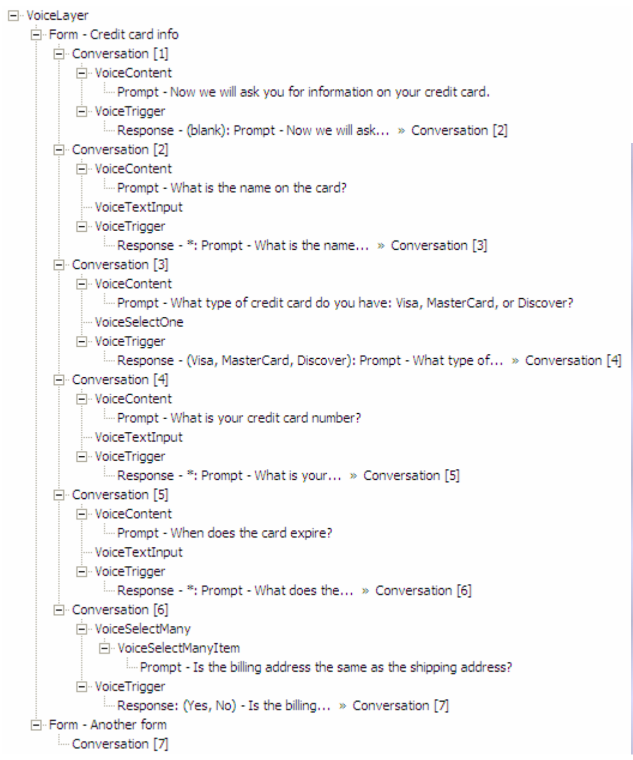
Credit Card Number

Expiration Date

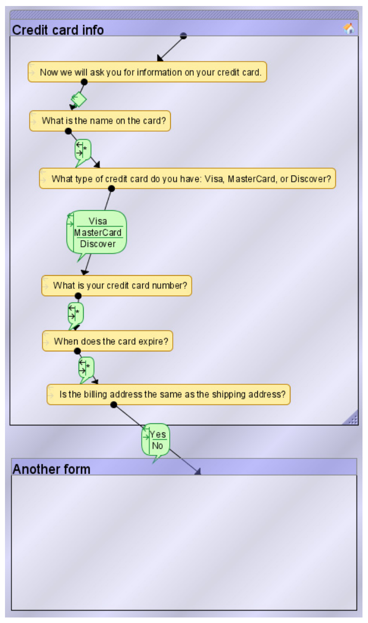
☐ Use my shipping address

Continue

Another page



```
VoiceLayer
  Form - Credit card info
    Conversation [1]
      VoiceContent
        Prompt - Now we will ask you for information on your credit card.
      VoiceTrigger
        Response - (blank): Prompt - Now we will ask...  »  Conversation [2]
    Conversation [2]
      VoiceContent
        Prompt - What is the name on the card?
      VoiceTextInput
      VoiceTrigger
        Response - *: Prompt - What is the name...  »  Conversation [3]
    Conversation [3]
      VoiceContent
        Prompt - What type of credit card do you have: Visa, MasterCard, or Discover?
      VoiceSelectOne
      VoiceTrigger
        Response - (Visa, MasterCard, Discover): Prompt - What type of...  »  Conversation [4]
    Conversation [4]
      VoiceContent
        Prompt - What is your credit card number?
      VoiceTextInput
      VoiceTrigger
        Response - *: Prompt - What is your...  »  Conversation [5]
    Conversation [5]
      VoiceContent
        Prompt - When does the card expire?
      VoiceTextInput
      VoiceTrigger
        Response - *: Prompt - What does the...  »  Conversation [6]
    Conversation [6]
      VoiceSelectMany
        VoiceSelectManyItem
          Prompt - Is the billing address the same as the shipping address?
      VoiceTrigger
        Response: (Yes, No) - Is the billing...  »  Conversation [7]
  Form - Another form
    Conversation [7]
```



Credit card info

Now we will ask you for information on your credit card.

What is the name on the card?

What type of credit card do you have: Visa, MasterCard, or Discover?

Visa
MasterCard
Discover

What is your credit card number?

When does the card expire?

Is the billing address the same as the shipping address?

Yes
No

Another form

**FIGURE 5-8** The scenegraph (top left and bottom left) for a smartphone UI (top right) and its equivalent voice UI (bottom right).

Table 5.3 lists the model objects and the corresponding view objects. All of the view objects are descendants of `PNode`—none of them are Swing widgets. It should look familiar, as it is the basis of Tables 4.1, 4.2, 5.1, and 5.2.

**TABLE 5.3** The model objects for pages and controls, along with their corresponding view objects.

| Model | Desktop/Smartphone View | Voice View | Components of VoiceView |
|---|---|---|---|
| Dialog | DialogView (invisible) | Form | |
| Page | PageView | Conversation | |
| PageRegion | PageRegionView | (none, merged with Conversation) | |
| OrgConnection | Arrow | (none) | |
| NavConnection | Arrow | determines destination of Response | |
| Content | Label | VoiceContent | Prompt |
| Trigger | Button<br>Hyperlink | VoiceTrigger | Response |
| TextInput | TextBox | VoiceTextInput | (determines text of Response of following Trigger) |
| SelectOne | RadioButtonGroup<br>ListBox<br>ComboBox | VoiceSelectOne | (determines text of Response of following Trigger) |
| SelectOne.Item | RadioButton | (none) | (none) |
| SelectMany | CheckBoxGroup | VoiceSelectMany | |
| SelectMany.Item | CheckBox | VoiceSelectManyItem | Prompt<br>(determines text of Response of following Trigger) |

Keeping with the model-view-controller paradigm, virtually all changes to the model go through the controller, which changes the model, which fires events that the views listen to and change themselves. The following explains what happens when the designer wants to add a button to a desktop page in the All Devices layer (i.e., the equivalent button and trigger also will be added to the smartphone and voice views, respectively). Other events, such as adding a page or deleting an object, occur in essentially the same way.

1 When the designer selects the button tool, the `VisualCanvas` object attaches the associated event listener, which is an instance of `InsertButtonHandler`, to itself.

2 When the designer clicks on the `PageRegionView`, the `InsertButtonHandler` handles the event. It creates a new instance of the model object associated with a button, `Trigger`, and fills in the appropriate desktop properties such as location. Then it creates a new command for adding the `Trigger` to the appropriate `PageRegion`. The command figures out what the appropriate smartphone and voice properties of the new `Trigger` should be. Finally, it asks the command queue associated with the current document to execute the command.

3 The command queue executes the command, which adds the new `Trigger` to the specified `PageRegion`. This fires an `elementAdded` event to anyone listening to changes to the `PageRegion`.

4 The `PageRegionViews` in Desktop and Smartphone view receives the `elementAdded` event. In response, it creates a Button with the Trigger's properties, and adds the Button to itself. Likewise, the associated Conversation in Voice view receives the `elementAdded` event, creates a `VoiceTrigger`, and adds the `VoiceTrigger` to itself.

All edits go through the same process of setting up a `Command` and having the command queue execute it. When the designer creates or deletes an object, or changes its caption if it has one, Damask constructs the `Command` so that the action affects all devices if the object is in the All Devices layer. Currently, moving an object results in a `Command` that affects only one device.

To undo or redo an action, Damask simply calls the command queue to undo or redo. All commands in Damask have undo and redo capabilities.

### *5.2.4 Layers*

Layers in Damask are implemented indirectly. Suppose the designer is in the Desktop view and changes the active layer from All Devices to This Device. First, the `VisualLayer` sets a `deviceTypeForNewElement` property from `DeviceType.ALL` to `DeviceType.DESKTOP`. This property determines the device type of all new objects created in the `VisualLayer`. Then every object in the `VisualLayer` compares its model's `deviceType` property with the `VisualLayer`'s `deviceTypeForNewElement` property. If they do not match, then the object is grayed out and is not selectable; otherwise, the object is drawn normally and is selectable.

### *5.2.5 Templates*

The implementation of templates takes advantage of Damask's MVC architecture. As mentioned earlier, a `Dialog` object maintains a list of `TemplateDialog` objects to represent the templates being applied to the dialog. In the corresponding desktop and smartphone views, the `PageRegionViews` that make up the corresponding `DialogView` register listeners for events from the `TemplateDialog`. That way, when `Controls` are added or removed from the `TemplateDialog`, all of the corresponding `DialogViews` can add or remove corresponding `ControlViews`. A similar operation occurs in the voice view.

### 5.2.6 *Patterns and pattern instances*

There are two main view classes for patterns and pattern instances. A `PatternInstanceView` object draws the text and the blue dotted box around a pattern instance. A `PatternBrowser` object represents the Pattern Browser window. It uses standard Swing widgets to render the contents of the window, including the Search box, the patterns index, and the pattern itself.

Damask uses Java's standard drag-and-drop mechanism to support dragging a pattern solution and dropping it on to the main window for pattern instantiation. When a designer drags a solution from the Pattern Browser window, the `PatternBrowser` object takes the `PatternSolution` object associated with the current `Pattern` instance and the wraps it in a `Transferable`, which is a standard Java class for handling data being dragged and dropped. When the designer drops it in the main window, the `DamaskCanvas` has a `DropTargetListener`, another standard Java interface, that listens for objects being dropped onto it. The `DropTargetListener` figures out whether the object being dropped is a `DamaskSolution`. If it is, then a `Command` is created that performs the steps for pattern instantiation described in Section 5.1.7.

## 5.3 Run mode

When the designer opens a page in Run mode, a new window called a `DamaskRunFrame` is created. It contains Back and Forward buttons and a `DamaskRunCanvas`, a descendant of `PCanvas`, where the page is displayed. The `DamaskRunCanvas` takes the page and creates a `PageView`, just like a normal `DamaskCanvas`. It also sets the `PageView`'s `InRunMode` property to true. This attaches an event listener for every `ControlView` within the `PageView`. The event listener listens for UI events from the designer in the Run window.

For example, when a `Button` is pressed in the Run window, the attached event listener inverts the `Button`'s color when the mouse is pressed. When the mouse is released, it gets the `Trigger` object that is the model for the `Button`, finds the `NavConnection` whose source is the `Trigger`, finds the destination `Page` of the `NavConnection`, and then tells the `DamaskRunCanvas` to display the destination `Page`.

## 5.4 VoiceXML Generation

Explaining VoiceXML generation is best done with respect to an example (see Figure 5-9).

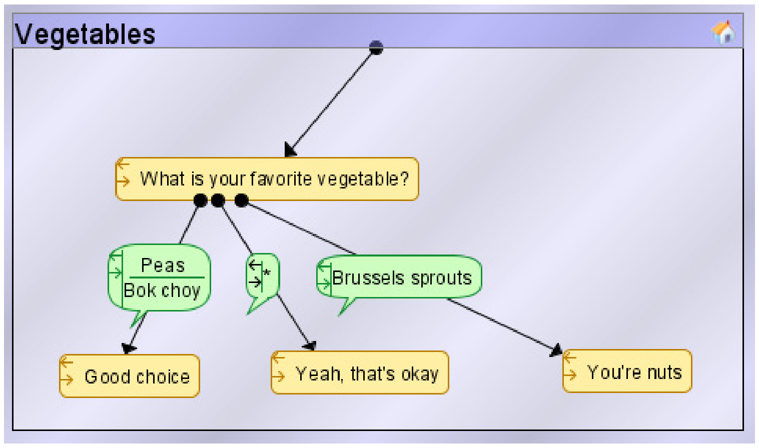


**FIGURE 5-9** An example voice UI for generating a VoiceXML file. This is the same as Figure 5-5.

In a VoiceXML interface, groups of related voice functionality are divided into `<form>`s. To generate a VoiceXML file from a voice user interface, first Damask creates a `<form>` for each `Page`, which in a voice ui is a prompt and all responses that originate from that prompt. In this case, there will be four `<form>`s, one for each prompt. Each `<form>` has an associated `<field>`, which is essentially a variable. The value of the variable will be set depending on what the user says. The field is not used in all cases, but it does not hurt to generate it.[1]

```
<?xml version="1.0" encoding="UTF-8"?>
<vxml version="2.0">
  <form id="form_home">
    <field name="field_vegetable">
    </field>
  </form>
  <form id="form_good">
    <field name="field_good">
    </field>
  </form>
  <form id="form_okay">
    <field name="field_okay">
    </field>
  </form>
  <form id="form_bad">
    <field name="field_bad">
    </field>
  </form>
</vxml>
```

For each prompt in the Damask design, a `<prompt>` tag is added to the appropriate `<form>`. (The rest of the VoiceXML examples focus on `<form_home>`, since that is where most of the complexity in this example is.)

```
  <form id="form_home">
    <field name="field_vegetable">
      <prompt>
        <audio>What is your favorite vegetable?</audio>
      </prompt>
    </field>
  </form>
```

[1] In the VoiceXML snippets, we are using human-readable names for the `id` and `name` attributes to make them easier to follow. However, Damask actually uses machine-generated GUIDs.

A `<grammar>` tag is set up to handle the user's response. Each entry in the grammar corresponds to one response in the Damask design. The grammar assigns a value to the field depending on what the user says. In this case, if the user says "Brussels sprouts," then the field `field_vegetable` is assigned the value "bad_vegetable." If the user says "peas" or "bok choy," then `field_vegetable` is assigned "good_vegetable."

```
<form id="form_home">
  <field name="field_vegetable">
    <prompt>
      <audio>What is your favorite vegetable?</audio>
    </prompt>
    <grammar type="application/x-gsl" mode="voice">
      <![CDATA[
      [(brussels sprouts)] {<field_vegetable "bad_vegetable">}
      [peas (bok choy)] {<field_vegetable "good_vegetable">}
      ]]>
    </grammar>
  </field>
</form>
```

The response with the asterisk (*), which matches all utterances that do not match any other response, is handled by the `<nomatch>` tag, which in this case makes the UI go to the `form_okay` form. If there is no response with an asterisk, then a default `<nomatch>` tag is generated, in which the computer says, "I'm sorry, I didn't get that," and then replays the `<prompt>`.

```
<form id="form_home">
  <field name="field_vegetable">
    <prompt>
      <audio>What is your favorite vegetable?</audio>
    </prompt>
    <nomatch>
      <goto next="#form_okay" />
    </nomatch>
    <grammar type="application/x-gsl" mode="voice">
      <![CDATA[
      [(brussels sprouts)] {<field_vegetable "bad_vegetable">}
      [peas (bok choy)] {<field_vegetable "good_vegetable">}
      ]]>
    </grammar>
  </field>
</form>
```

A `<filled>` tag handles what to do once the field has been assigned. In this case, the UI goes to different forms depending on the field's value.

```
<form id="form_home">
  <field name="field_vegetable">
    <prompt>
      <audio>What is your favorite vegetable?</audio>
    </prompt>
    <nomatch>
      <goto next="#form_okay" />
    </nomatch>
    <grammar type="application/x-gsl" mode="voice">
      <![CDATA[[
      [(brussels sprouts)] {<field_vegetable "bad_vegetable">}
      [peas (bok choy)] {<field_vegetable "good_vegetable">}
      ]]>
    </grammar>
    <filled>
      <if cond="field_vegetable == 'good_vegetable'">
        <goto next="#form_good" />
        <elseif cond="field_vegetable == 'bad_vegetable'" />
        <goto next="#form_bad" />
      </if>
    </filled>
  </field>
</form>
```

Finally, to handle the case where the user does not say anything within a default period of time, Damask always inserts a `<noinput>` tag, which says a message and then replays the `<prompt>`.

```
<form id="form_home">
  <field name="field_vegetable">
    <prompt>
      <audio>What is your favorite vegetable?</audio>
    </prompt>
    <nomatch>
      <goto next="#form_okay" />
    </nomatch>
    <grammar type="application/x-gsl" mode="voice">
      <![CDATA[[
      [(brussels sprouts)] {<field_vegetable "bad_vegetable">}
      [peas (bok choy)] {<field_vegetable "good_vegetable">}
      ]]]>
    </grammar>
  </field>
    <filled>
      <if cond="field_vegetable == 'good_vegetable'">
        <goto next="#form_good" />
        <elseif cond="field_vegetable == 'bad_vegetable'" />
        <goto next="#form_bad" />
      </if>
    </filled>
    <noinput>
      <audio>I'm sorry, I didn't hear you.</audio>
      <reprompt />
    </noinput>
</form>
```

Figure 5-10 is the entire generated VoiceXML file for the voice UI in Figure 5-9, with the original machine-generated IDs.

```
<?xml version="1.0" encoding="UTF-8"?>
<vxml version="2.0">
  <form id="formid_75_37cb61f8f3397d86_5f797ad0_104d8161f48__7fb4">
    <field name="id_75_37cb61f8f3397d86_5f797ad0_104d8161f48__7fb4">
      <prompt>
        <audio>What is your favorite vegetable?</audio>
      </prompt>
      <nomatch>
        <goto
next="#formid_91_37cb61f8f3397d86_5f797ad0_104d8161f48__7fa4" />
      </nomatch>
      <grammar type="application/x-gsl" mode="voice"><![CDATA[[
[(brussel sprouts)] {<id_75_37cb61f8f3397d86_5f797ad0_104d8161f48__7fb4
"id_109_37cb61f8f3397d86_5f797ad0_104d8161f48__7f92">}
[peas (bok choy)] {<id_75_37cb61f8f3397d86_5f797ad0_104d8161f48__7fb4
"id_119_37cb61f8f3397d86_5f797ad0_104d8161f48__7f88">}
]]]></grammar>
      <filled>
        <if cond="id_75_37cb61f8f3397d86_5f797ad0_104d8161f48__7fb4 ==
'id_119_37cb61f8f3397d86_5f797ad0_104d8161f48__7f88'">
          <goto
next="#formid_83_37cb61f8f3397d86_5f797ad0_104d8161f48__7fac" />
          <elseif cond="id_75_37cb61f8f3397d86_5f797ad0_104d8161f48__7fb4
== 'id_109_37cb61f8f3397d86_5f797ad0_104d8161f48__7f92'" />
```

```
          <goto
next="#formid_99_37cb61f8f3397d86_5f797ad0_104d8161f48__7f9c" />
        </if>
      </filled>
      <noinput>
        <audio>I'm sorry, I didn't hear you.</audio>
        <reprompt />
      </noinput>
    </field>
  </form>
  <form id="formid_83_37cb61f8f3397d86_5f797ad0_104d8161f48__7fac">
    <field name="id_83_37cb61f8f3397d86_5f797ad0_104d8161f48__7fac">
      <prompt>
        <audio>Good choice</audio>
      </prompt>
      <noinput>
        <audio>I'm sorry, I didn't hear you.</audio>
        <reprompt />
      </noinput>
      <nomatch>
        <audio>I'm sorry, I didn't get that.</audio>
        <reprompt />
      </nomatch>
    </field>
  </form>
  <form id="formid_91_37cb61f8f3397d86_5f797ad0_104d8161f48__7fa4">
    <field name="id_91_37cb61f8f3397d86_5f797ad0_104d8161f48__7fa4">
      <prompt>
        <audio>Yeah, that's okay</audio>
      </prompt>
      <noinput>
        <audio>I'm sorry, I didn't hear you.</audio>
        <reprompt />
      </noinput>
      <nomatch>
        <audio>I'm sorry, I didn't get that.</audio>
        <reprompt />
      </nomatch>
    </field>
  </form>
  <form id="formid_99_37cb61f8f3397d86_5f797ad0_104d8161f48__7f9c">
    <field name="id_99_37cb61f8f3397d86_5f797ad0_104d8161f48__7f9c">
      <prompt>
        <audio>You're nuts</audio>
      </prompt>
      <noinput>
        <audio>I'm sorry, I didn't hear you.</audio>
        <reprompt />
      </noinput>
      <nomatch>
        <audio>I'm sorry, I didn't get that.</audio>
        <reprompt />
      </nomatch>
    </field>
  </form>
</vxml>
```

**FIGURE 5-10** The generated VoiceXML file for the user interface in Figure 5-9.

## 5.5 Summary

Damask's architecture is driven by the desire for a single abstract model to represent a cross-device user interface design. The model-view-controller-based implementation decouples the device-specific views from the abstract model. Model-based UI techniques enable designers to quickly see how changes to one device UI manifest themselves in the other device UIs.

# 6 Damask Evaluation

In this chapter, we present the results of our evaluations of Damask. We wanted to find out whether designers could understand and use the patterns for a particular design task, use layers to control which parts of a user interface design gets retargeted, and create designs that had more functionality and quality with patterns than without patterns, within a fixed amount of time.

We asked eight professional web UI designers and four professional voice UI designers to design user interfaces using Damask under two conditions, one with layers and patterns and one without. For each condition, they designed a desktop UI and a UI for one other platform, either smartphone or voice. We found that compared to not using layers and patterns, designers using layers and patterns spent less time designing a UI for two devices. For desktop UIs, the designs with patterns were more similar to the two most usable commercial UIs in the same domain than the designs without patterns, and they had more functionality. Designers generally understood and liked the design pattern concept, and the patterns were used appropriately during the design process. We also found that designers generally understood how to use layers to control which parts of their user interface designs would appear across all device types, although the implementation needs to be refined to make it more usable.

## 6.1 Experimental Procedure

The evaluation consisted of two phases. The first phase addressed designing desktop and smartphone UIs, and the second addressed designing desktop and voice UIs. We did not address designing for all three types of devices at the same time, as that would have made the experimental session with each participant prohibitively long. The participants used a Fujitsu T Series Lifebook (1.4 GHz Pentium M with 1 GB RAM) running Windows XP Tablet PC Edition 2005 and Java 2 Standard Edition 5.0, at a screen resolution of 1024×768 (see Figure 6-1).

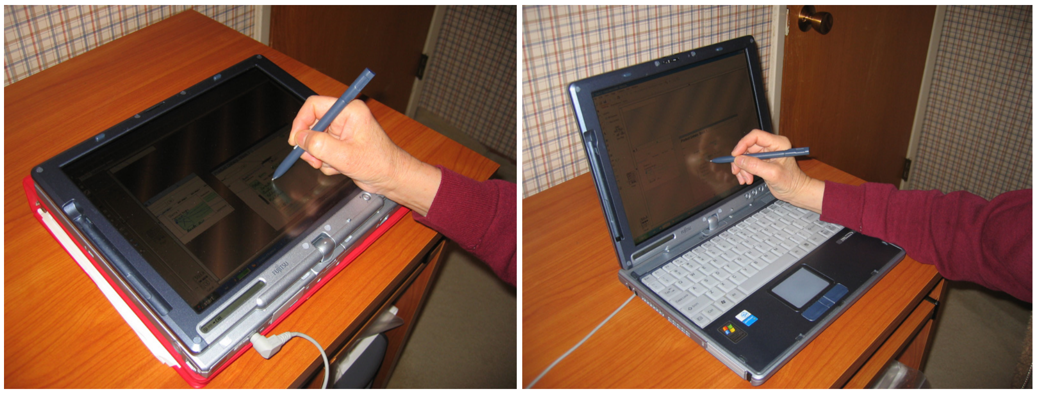


**FIGURE 6-1** The Fujitsu Tablet PC used in the experiments, in tablet mode (left) and laptop mode (right).

The experimental sessions were recorded using a video camera. Out of the 28 sessions, in the first four sessions the camera was aimed at the Tablet PC itself. To improve the video quality, the camera was aimed at an external LCD monitor hooked up to the Tablet PC for the rest of the sessions. I was present for all of the sessions, and for half of the sessions, an undergraduate student working with me was also present to set up the equipment and take additional notes.

### 6.1.1 Phase 1: Study of desktop and smartphone user interface design

Eight designers participated in Phase 1. Three additional designers were pilot testers. The participants were divided evenly by random selection into two groups. In Group A, participants used Damask without layers and patterns first, and in Group B, participants used Damask with layers and patterns first. In both groups, the evaluation was spread out over two sessions on two different days. The sessions were separated by one to 42 days (mean of 16 days, median of 7½ days, standard deviation of 17.76), depending on the participant's schedule. The following are the tasks that all of the participants did, followed by how they were ordered depending on the group:

- **Basic tutorial (§C.3):** We gave participants a tutorial on Damask, without the layers or patterns features enabled, which lasted about ten minutes.
- **Tablet PC warm-up task (§C.4):** The participants did a short warm-up task to get acquainted with the Tablet PC, where they had to edit a drawing and draw a bumblebee.
- **Damask warm-up task (§C.6.1 Task 1 for desktop/smartphone participants, §C.5.1 for desktop/voice participants):** The participants took an existing Damask design, added a new page with some content, and then linked an existing page to the new page.
- **Online music store (§C.6.1 Task 2 for desktop/smartphone participants, §C.6.2 for desktop/voice participants):** The participants were asked to design an online music store targeting both the desktop web and either smartphones or voice. The design should support browsing through a category to buy a CD, add the CD to a shopping cart, and then checkout the shopping cart. The desktop/smartphone participants were told to also support smartphone users

in buying ringtones. There was no time limit for the design task, but it was designed to take about 1½ to 2 hours.

- **Questionnaire about Damask without layers or patterns (§C.6.3):** Participants filled out a post-test questionnaire regarding their experiences with using Damask without layers or patterns.
- **Layers tutorial (§C.7.1):** We gave participants a tutorial on the layers feature in Damask, which lasted about five minutes.
- **Pattern introduction (§C.7.3):** We demonstrated how to use the Pattern Browser with Damask, which lasted about five minutes.
- **Pattern learning:** The participants browsed the pattern collection for about ten minutes and were told that there would be a quiz afterwards. This encouraged them to look through the collection carefully. The participants then took the quiz (§C.7.4), which was open-book.
- **Layers and patterns warm-up (§C.7.2, §C.7.5):** The participants modified an existing design, using layers and patterns.
- **Online bookstore (§C.7.6):** The participants were asked to design an online bookstore targeting both the desktop web and either smartphones or voice. The design should support browsing through a category to buy a book, add the book to a shopping cart, and then checkout the shopping cart. The desktop/smartphone participants were told to also support smartphone users in finding the nearest physical bookstore to the user's current location. There was no time limit for the design task, but it was designed to take about 1½ to 2 hours.

- **Questionnaire about Damask with layers and patterns (§C.7.7):** Participants filled out a post-test questionnaire regarding their experiences with using Damask with layers or patterns.

**TABLE 6.1** The details and task ordering of Phase 1 (desktop/smartphone)

| Group A | Group B |
|---|---|
| *Session 1* | |
| 1. Basic tutorial<br>2. Tablet PC warm-up task<br>3. Damask warm-up task<br><br><br><br><br>4. Online music store<br>5. Questionnaire about Damask without layers or patterns | 1. Basic tutorial<br>2. Tablet PC warm-up task<br>3. Damask warm-up task<br>4. Layers tutorial<br>5. Pattern introduction<br>6. Pattern learning<br>7. Layers and patterns warm-up<br>8. Online bookstore<br>9. Questionnaire about Damask with layers and patterns |
| *Session 2* | |
| 1. Layers tutorial<br>2. Pattern introduction<br>3. Pattern learning<br>4. Layers and patterns warm-up<br>5. Online bookstore<br>6. Questionnaire about Damask with layers and patterns | <br><br><br><br>1. Online music store<br>2. Questionnaire about Damask without layers or patterns |

### 6.1.2 *Phase 2: Study of desktop and voice user interface design*

Four designers participated in Phase 2. Two other designers started Phase 2, but were dropped after they notified us they could not continue after the first session. As in Phase 1, the participants for Phase 2 were divided into two groups, but this time, there were three sessions instead of two. This was to give them more time to learn the UIs for designing both desktop and voice designs, which are very different. The time between sessions 1 and 2 was 1–15 days (mean 7 days) and between sessions 2 and 3 was 1–15 days (mean 6 days). The first session was the same for both conditions, while the second and third sessions were basically swapped between the two groups. The tasks were essentially the same as in Phase 1, with the addition of another Damask warm-up task in Session 1 to design an online bank (see Section C.5.2). We also de-emphasized sketching for Phase 2, so that text phrases typed into the desktop design would easily be transferred into the voice design.

**TABLE 6.2** The details of and task ordering of Phase 2 (desktop/voice)

| Group A | Group B |
|---|---|
| *Session 1* | |
| 1. Basic tutorial<br>2. Tablet PC warm-up task<br>3. Damask warm-up task<br>4. Damask warm-up task 2 | 1. Basic tutorial<br>2. Tablet PC warm-up task<br>3. Damask warm-up task<br>4. Damask warm-up task 2 |
| *Session 2* | |
| 1. Online music store<br>2. Questionnaire about Damask without layers or patterns | 1. Layers tutorial<br>2. Pattern introduction<br>3. Pattern learning<br>4. Layers and patterns warm-up<br>5. Online bookstore<br>6. Questionnaire about Damask with layers and patterns |
| *Session 3* | |
| 1. Layers tutorial<br>2. Pattern introduction<br>3. Pattern learning<br>4. Layers and patterns warm-up<br>5. Online bookstore<br>6. Questionnaire about Damask with layers and patterns | 1. Online music store<br>2. Questionnaire about Damask without layers or patterns |

## 6.2 Participants

In Phase 1, we screened for user interface designers who had experience designing web sites and optional experience designing for mobile phones. Each participant was promised an Amazon.com gift certificate worth $250. We gave priority to designers who had some mobile phone design experience.

All eight participants in Phase 1 had at least five years of experience designing web sites, but their mobile phone UI experience was much less (see Table 6.3). Two had one to two years of experience, five had less than one year, and one had no experience. There were equal numbers of men and women, as well as equal numbers of those between 21 and 30 years of age and those over 30.

All of the participants use paper and whiteboards during the design process. The three computer programs used by more than half of the participants were Visio (7 out of 8), Microsoft Word (6), and Adobe Photoshop (5).

**TABLE 6.3** Demographics of Desktop/Smartphone phase (Phase 1) participants*

| Participant | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | % total |
|---|---|---|---|---|---|---|---|---|---|
| Age | 21-30 | 21-30 | 21-30 | 31-40 | 31-40 | 41-50 | 21-30 | 31-40 | |
| Sex | M | M | M | F | F | F | M | F | |
| *Design areas in which participants felt knowledgeable* | | | | | | | | | |
| GUI | ✓ | ✓ | ✓ | | ✓ | ✓ | ✓ | ✓ | 88 |
| Graphic design | ✓ | | | ✓ | | | ✓ | ✓ | 50 |
| Info architect | ✓ | ✓ | | | ✓ | ✓ | ✓ | ✓ | 75 |
| Mobile phone | | | | ✓ | | | | | 12 |
| Web | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | 100 |
| Other | | | | | [1] | [2] | [3] | | |
| Years of experience | ≥ 5 | ≥ 5 | ≥ 5 | ≥ 5 | ≥ 5 | ≥ 5 | ≥ 5 | ≥ 5 | |
| Web | ≥ 5 | ≥ 5 | ≥ 5 | ≥ 5 | ≥ 5 | ≥ 5 | ≥ 5 | ≥ 5 | |
| Mobile phone | < 1 | < 1 | 1-2 | 1-2 | < 1 | 0 | < 1 | < 1 | |
| *UI tools used* | | | | | | | | | |
| Dreamweaver | | | | | ✓ | ✓ | ✓ | ✓ | 50 |
| Excel | | ✓ | | | | | | ✓ | 25 |
| Flash | | | | | ✓ | | | | 12 |
| FrontPage | | | | | | | | ✓ | 12 |
| Illustrator | ✓ | ✓ | | ✓ | | | | ✓ | 50 |
| Paper | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | 100 |
| Photoshop | | ✓ | ✓ | | ✓ | | ✓ | ✓ | 62 |
| PowerPoint | ✓ | | ✓ | | | | | ✓ | 38 |
| Text editor | | | ✓ | | ✓ | | | ✓ | 38 |
| Visio | ✓ | ✓ | ✓ | | ✓ | ✓ | ✓ | ✓ | 88 |
| Whiteboard | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | 100 |
| Word | | ✓ | | ✓ | ✓ | ✓ | ✓ | ✓ | 75 |
| OmniGraffle | | | | | | ✓ | | | 12 |

[1] Accessibility design
[2] Product copy for mobile phones
[3] Software design

In Phase 2, we screened for user interface designers who had experience designing voice user interfaces. Each participant was promised an Amazon.com gift certificate worth $375, since there was one more session per person than in Phase 1.

* Participants 1, 2, and 3 were pilot subjects.

Three of the four participants had at least five years of experience designing voice user interfaces, and the other one had three to four years (see Table 6.4). None of them had more than a year of web experience. There were equal numbers of men and women, as well as equal numbers of those between 21 and 30 years of age and those over 30. All of the participants in this phase use Visio and Microsoft Word, as well as in-house proprietary tools, during the design process. Over half also used paper and whiteboards.

**TABLE 6.4** Demographics of Desktop/Voice phase (Phase 2) participants†

| Participant | 13 | 15 | 16 | 17 | % total |
|---|---|---|---|---|---|
| Age | 21-30 | 31-40 | 21-30 | 31-40 | |
| Sex | F | M | M | F | |
| *Design areas in which participants felt knowledgeable* | | | | | |
| GUI | ✓ | ✓ | | | 50 |
| Voice | ✓ | ✓ | ✓ | ✓ | 100 |
| Web | ✓ | | | | 25 |
| Mobile phone UI | | ✓ | | | 25 |
| Other | Interaction | | | | |
| Years of experience | ≥ 5 | ≥ 5 | 3-4 | ≥ 5 | |
| Web | < 1 | < 1 | < 1 | < 1 | |
| Voice | ≥ 5 | ≥ 5 | 3-4 | ≥ 5 | |
| *UI tools used* | | | | | |
| Paper | ✓ | ✓ | ✓ | | 75 |
| Whiteboard | ✓ | ✓ | ✓ | | 75 |
| Text editor | | | ✓ | | 25 |
| Photoshop | | ✓ | ✓ | | 50 |
| Visio | ✓ | ✓ | ✓ | ✓ | 100 |
| Word | ✓ | ✓ | ✓ | ✓ | 100 |
| Excel | | ✓ | | | |
| Proprietary voice tools | ✓ | ✓ | ✓ | ✓ | |

† Participants 12 and 14 only finished the first session, after which they notified us that they no longer had time to complete the last two sessions.

## 6.3 Introduction to the Results

Here we report what the design process was like using Damask with and without patterns, the overall Damask experience, and detailed findings related to patterns and layers.

## 6.4 Results Concerning All Designs Created by the Participants

We looked at several aspects on how patterns affected the design process: the amount of time the participants took, how much they were able to accomplish within that time, and whether they preferred the designs with patterns and layers or without.

### *6.4.1 Time spent by participants*

All of the participants chose to design the desktop user interface first, although they were not instructed to do so, and Damask supports either ordering. There is a clear difference in how much time the designers spent in the non-pattern condition versus the pattern condition.

Among the desktop/smartphone participants, designers spent about an hour designing the desktop interface in both conditions, although in the patterns condition, an average of 9 minutes and 35 seconds of that hour was spent in the pattern browser while designing the desktop UI (see Table 6.5). However, they spent statistically significantly less time designing the smartphone user interface using patterns and layers (on average, 21 minutes with patterns versus 42 minutes without patterns), and only two designers out of 8 spent any time in the pattern browser while designing for the second device (average over all 8 designers, 12 seconds).

This shows that patterns and layers help cut down the amount of time spent retargeting an interface design after the first device design is mostly complete. Over extended use, we would expect the desktop time to go down as well, as designers would become more familiar with the patterns included in Damask and the ten-minute pattern browsing time might be eliminated.

**TABLE 6.5** Mean time spent by desktop/smartphone participants creating UI design, in hours and minutes (gray background signifies $p < 0.05$, 2-tailed t-test)

| | Desktop | Pattern Browser during Desktop design | Smartphone | Pattern Browser during Smartphone design | Total |
|---|---|---|---|---|---|
| Non-patterns | 1:07 | | 0:42 | | 1:49 |
| Patterns | 1:03 | 0:10 | 0:21 | 12 secs | 1:24 |

Among the desktop/voice participants, the results are not as statistically strong. Designers spent more time designing the desktop interface in the pattern condition (55 minutes) than the non-pattern condition (44 minutes), but the difference is not statistically significant, and in the patterns condition, an average of 5 minutes and 51 seconds of that time was spent in the pattern browser while designing the desktop UI (see Table 6.6). They spent less time designing the voice user interface using patterns and layers (on average, 34 minutes with patterns versus 39 minutes without patterns), but this is also not statistically significant ($p > 0.30$).

**TABLE 6.6** Mean time spent by desktop/voice participants creating UI design, in hours and minutes ($p > 0.30$, 2-tailed t-test)

| | Desktop | Pattern Browser during Desktop design | Voice | Pattern Browser during Voice design | Total |
|---|---|---|---|---|---|
| Non-patterns | 0:44 | | 0:39 | | 1:23 |
| Patterns | 0:55 | 0:06 | 0:34 | 0:00 | 1:29 |

Much of this can be explained by the fact that Designer 16 only used one pattern in the patterns condition, so he did not take full advantage of the patterns capabilities. If we remove this designer, then the time spent on the desktop design is almost the same in the two conditions (48 minutes for non-patterns versus 50 minutes for patterns), and the difference in the time spent in total and in voice design between the two conditions becomes bigger and closer to being statistically significant (29 minutes for non-patterns versus 16 minutes for patterns); see Table 6.7. It is difficult to see statistically significant differences when there only three participants' worth of data, and we feel confident that we would get significance with more data.

**TABLE 6.7** Mean time spent by desktop/voice participants creating UI design, without Designer 16, in hours and minutes

| | Desktop | Pattern Browser during Desktop design | Voice | Pattern Browser during Voice design | Total |
|---|---|---|---|---|---|
| Non-patterns | 0:48 | | 0:29 | | 1:17 |
| Patterns | 0:50 | 0:08 | 0:16 | 0:00 | 1:06 |
| *p* for 2-tailed t-test | > 0.50 | | 0.13 | | 0.09 |

### 6.4.2 *Extent of designs created by participants*

Using patterns, the designers were able to accomplish more during their design sessions. As a measure of this, we counted how many pages, elements within a page, and arrows between pages the designers created in the desktop and smartphone designs, and how many forms, prompts, and responses were created in the voice designs. This way, we can see whether or not the designers were faster in the patterns condition because their designs were simpler.

There are big differences between the non-patterns and patterns conditions, all of them statistically significant except for the voice UI designs (see Table 6.8). In general, in the patterns condition, the designers created about twice as many pages or forms, and about 2½ to 3 times as many other objects, compared to the non-patterns condition. Also, the vast majority of objects created in the patterns condition were part of a pattern, showing that the participants successfully incorporated the patterns in their designs.

**TABLE 6.8** Completeness of designs (gray background signifies $p < 0.05$, 2-tailed t-test)

| Condition | Desktop | | | Smartphone | | |
|---|---|---|---|---|---|---|
| | Pages | Elements | Connections | Pages | Elements | Connections |
| Non-patterns | 6 | 95 | 11 | 8 | 62 | 11 |
| Patterns | 13 | 281 | 29 | 17 | 215 | 27 |
| % from patterns | 88% | 92% | 76% | 88% | 92% | 77% |

| Condition | Voice | | |
|---|---|---|---|
| | Forms | Prompts | Responses |
| Non-patterns | 5 | 18 | 22 |
| Patterns | 10 | 72 | 90 |
| % from patterns | 77% | 88% | 90% |

If we remove Designer 8, who used no patterns, and Designer 16, who used only one pattern, then almost all of the differences are statistically significant (see Table 6.9).

**TABLE 6.9** Completeness of designs among designers who used more than 1 pattern in the patterns condition (gray background signifies $p < 0.05$, 2-tailed t-test)

| Condition | Desktop | | | Smartphone | | |
|---|---|---|---|---|---|---|
| | Pages | Elements | Connections | Pages | Elements | Connections |
| Non-patterns | 6 | 102 | 11 | 8 | 64 | 11 |
| Patterns | 14 | 324 | 34 | 19 | 240 | 30 |
| % from patterns | 94% | 95% | 79% | 93% | 94% | 80% |

| Condition | Voice | | |
|---|---|---|---|
| | Forms | Prompts | Responses |
| Non-patterns | 4 | 19 | 22 |
| Patterns | 11 | 91 | 115 |
| % from patterns | 88% | 92% | 71% |

### 6.4.3 *Preference of designs created by participants*

We asked designers whether they preferred the designs they made with patterns or without patterns (see Section C.7.7, Question 33). Seven out of the twelve participants said they preferred the design with patterns and layers. One of those participants, Participant 16, said, "The patterns made me realize I'd forgotten a lot of things in the first one, like all the functionality for shipping addresses/credit cards etc."

Three of the participants said they preferred the design without patterns and layers. However, two of these three used at most one pattern; they said they liked the designs they had created in the phase without patterns and layers because it was the second phase for them and they had become more comfortable with the tool. The other one explained that she had become more familiar with the tool and that the performance of the tool was better without patterns or layers.

Participant 6 said it depended on what the design sketch was for: the design without patterns was better for giving a high level overview because it had less detail, while the design with patterns was better for conducting a usability study or a scoping meeting with engineers. (Participant 5 misunderstood the question and did not respond to repeated requests to clarify, so his answer is not counted.)

## 6.5 Desktop Designs Created by the Participants

We analyzed the desktop designs in three ways: by asking other user interface designers to evaluate them, and by comparing the features and structure of the designs with those of two well-known e-commerce web sites.

### 6.5.1 Examples of desktop designs

Here are two desktop web designs that are typical of the ones our participants created. All of the designs can be found in Appendix D.

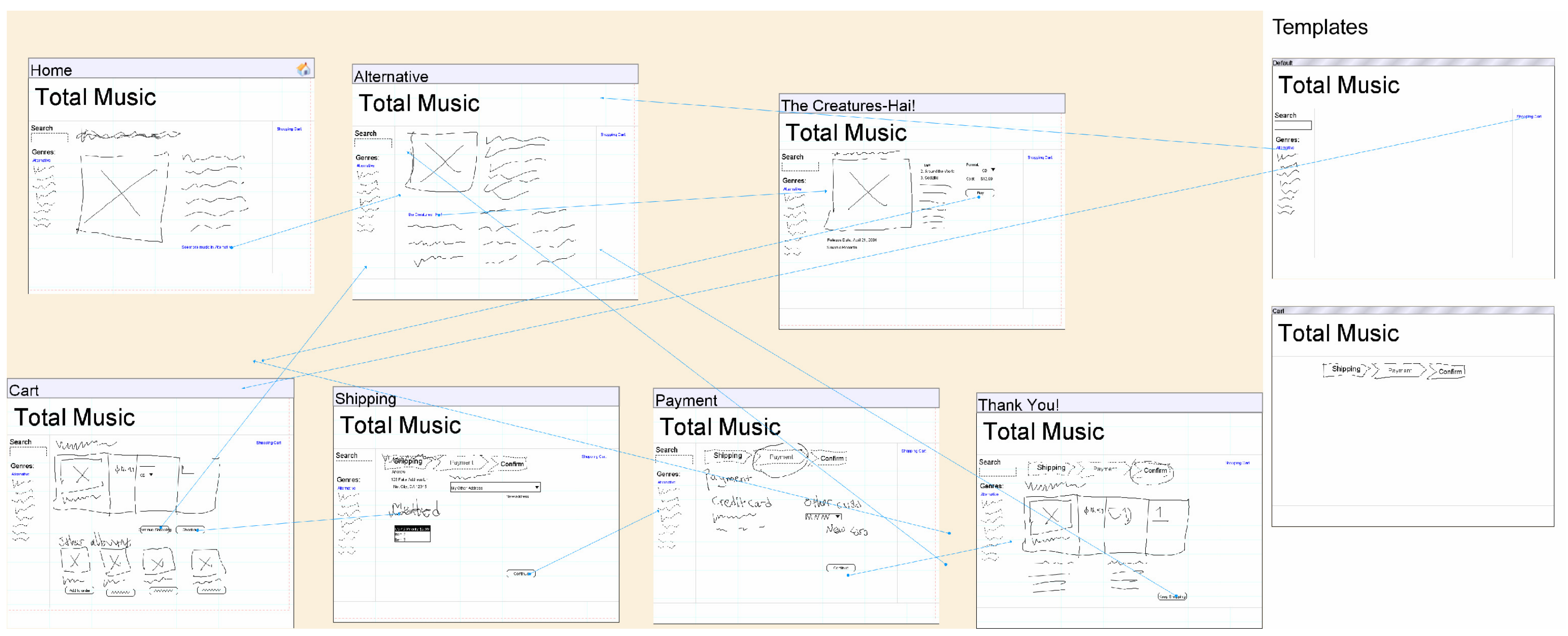


**FIGURE 6-2** Participant 6's desktop design for TotalMusic.com, created without patterns or layers.

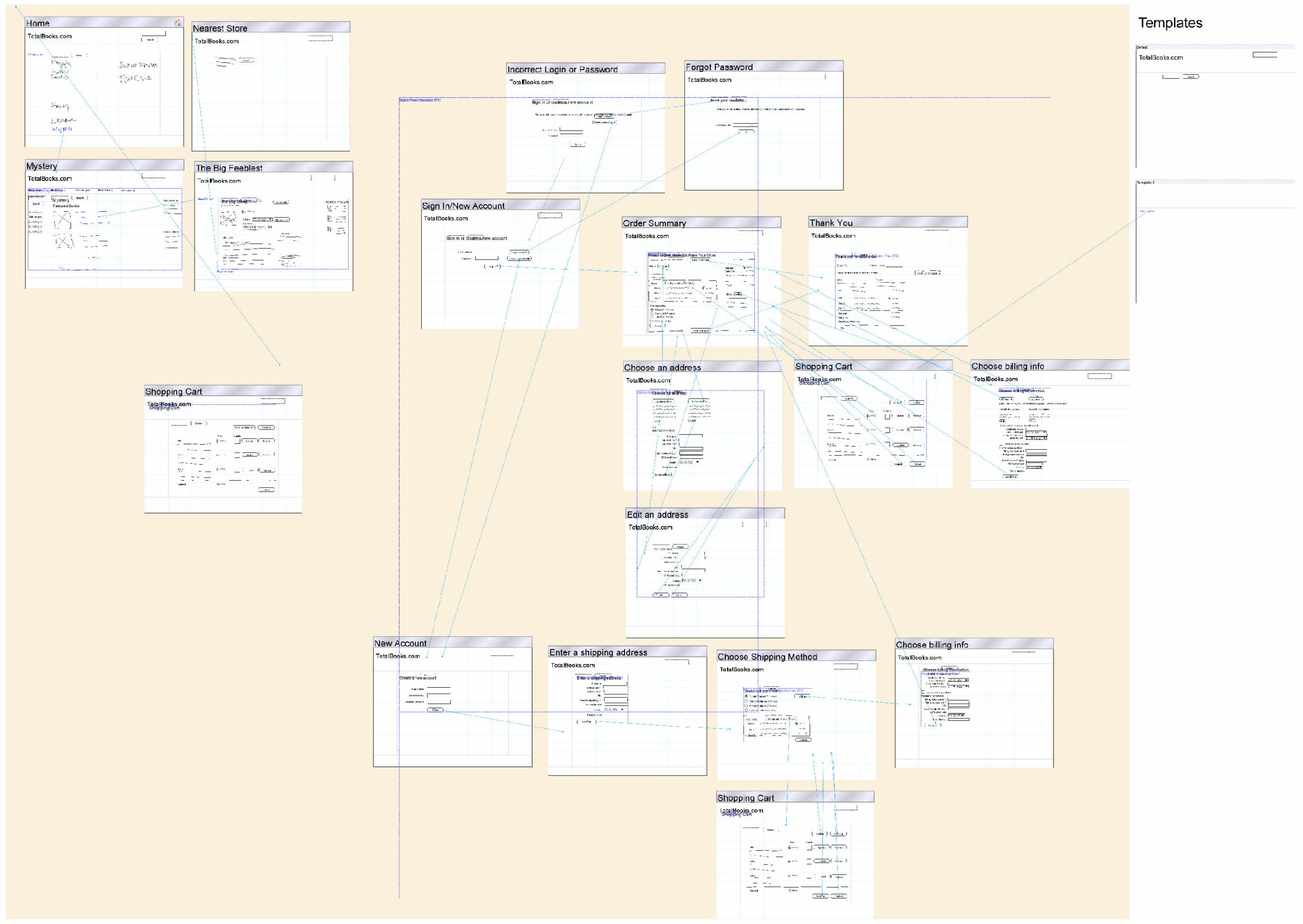


**FIGURE 6-3** Participant 6's desktop design for TotalBooks.com, created with patterns and layers.

Figure 6-2 shows a typical TotalMusic.com design created without patterns or layers, created by Participant 6. While it has pages for the basic functionality, it does not have much page detail. For example, the checkout pages do not allow the user to type in a new address or a new credit card, even though those were explicit design requirements; they have placeholders instead. Figure 6-3 shows a TotalBooks.com design created with patterns and layers by the same participant, after the TotalMusic.com design. It specifies features in more detail, especially the checkout process. However, it is somewhat harder to follow because of the detail.

Participant 6 believed each sketch had merits. He thought his TotalMusic.com sketch (Figure 6-2) would be useful if he "wanted to convey the basic design to a project team and help them envision it." His TotalBooks.com sketch (Figure 6-3) would be good "to run a usability test or rough project scoping meeting with engineers."

### 6.5.2 Quality analysis of desktop designs

We wanted to find out whether the patterns and layers affected the quality of the desktop designs created in that condition, compared to the designs without patterns or layers. Since the designs are not complete enough to do a full usability analysis, we instead created an online questionnaire for judges to fill out.

There are 24 desktop designs over the two conditions. Obviously, no single person would want to evaluate all 24 of them. Therefore, we recruited 18 professional designers online (via the mailing lists BayCHI Discussions and CHI-WEB) and randomly assigned four designs to each evaluator, so that each design was evaluated by three people. The four designs came from two designers (patterns and non-patterns), and the order of the four designs was randomized. The judges did not know that the four designs were from only two designers, nor were they made aware of anything related to patterns or Damask.

Table 6.10 shows the demographics of the evaluators, along with those of the designers for comparison. Overall, the evaluators are somewhat older on average and have more experience designing e-commerce web sites. See Appendix F, Questions 13–30 for the raw questionnaire data.

**TABLE 6.10** Demographics of the evaluators of the desktop designs. The demographics of the designers are included for comparison.

| | **Evaluators** | | | Designers | | |
|---|---|---|---|---|---|---|
| Total | | 18 | | | 12 | |
| *Gender* | | | | | | |
| Male | | 11 | 61% | | 6 | 50% |
| Female | | 7 | 39% | | 6 | 50% |
| *Age* | | | | | | |
| | 19 and under | 0 | 0% | under 20 | 0 | 0% |
| | 20-29 | 5 | 28% | 21-30 | 6 | 50% |
| | 30-39 | 11 | 61% | 31-40 | 5 | 42% |
| | 40-49 | 1 | 6% | 41-50 | 1 | 8% |
| | 50 or above | 1 | 6% | 51 or above | 0 | 0% |
| *Years of design experience* | | | | | | |
| < 1 | | 0 | 0% | | 0 | 0% |
| 1–2 | | 3 | 17% | | 0 | 0% |
| 3–4 | | 1 | 6% | | 1 | 8% |
| ≥ 5 | | 14 | 78% | | 11 | 92% |
| *Years of web design experience* | | | | | | |
| < 1 | | 0 | 0% | | 4 | 33% |
| 1–2 | | 4 | 22% | | 0 | 0% |
| 3–4 | | 2 | 11% | | 0 | 0% |
| ≥ 5 | | 12 | 67% | | 8 | 67% |
| *Number of e-commerce web sites designed* | | | | | | |
| 0 | | 2 | 11% | | 3 | 25% |
| 1–2 | | 4 | 22% | | 5 | 42% |
| 3–4 | | 6 | 33% | | 2 | 17% |
| ≥ 5 | | 6 | 33% | | 2 | 17% |

| | **Evaluators** | | Designers | |
|---|---|---|---|---|
| *Which platforms were the e-commerce sites designed for?* | | | | |
| Web | 12 | 67% | 9 | 75% |
| PDA or mobile phone | 4 | 22% | 3 | 25% |
| Voice | 1 | 6% | 1 | 8% |
| Other | 2 | 11% | 0 | 0% |
| *Design knowledge* | | | | |
| IA | 17 | 94% | 6 | 50% |
| Graphic design | 10 | 56% | 4 | 33% |
| Web design | 18 | 100% | 9 | 75% |
| GUI design | 14 | 78% | 9 | 75% |
| Mobile phone UI design | 3 | 17% | 2 | 17% |
| Voice UI design | 1 | 6% | 4 | 33% |
| Other | 2 | 11% | 4 | 33% |
| *How many times have you bought anything online in past 12 months?* | | | | |
| Never | 0 | 0% | 0 | 0% |
| 1–3 | 0 | 0% | 1 | 8% |
| 4–6 | 5 | 28% | 1 | 8% |
| 7–11 | 5 | 28% | 4 | 33% |
| ≥ 12 | 8 | 44% | 6 | 50% |
| *How did you buy those items?* | | | | |
| Web | 18 | 100% | 11 | 92% |
| Mobile phone display | 1 | 6% | 0 | 0% |
| Automated voice | 2 | 11% | 1 | 8% |
| Other | 1 | 6% | 0 | 0% |
| No answer | 0 | 0% | 1 | 8% |
| *How many times have you bought books online in past 12 months?* | | | | |
| Never | 0 | 0% | 0 | 0% |
| 1–3 | 8 | 44% | 5 | 42% |
| 4–6 | 7 | 39% | 3 | 25% |
| 7–11 | 1 | 6% | 2 | 17% |
| ≥ 12 | 2 | 11% | 2 | 17% |

| | **Evaluators** | | Designers | |
|---|---|---|---|---|
| *How did you buy those books?* | | | | |
| Web | 18 | 100% | 12 | 100% |
| Mobile phone display | 0 | 0% | 0 | 0% |
| Automated voice | 0 | 0% | 0 | 0% |
| Other | 1 | 6% | 0 | 0% |
| *How many times have you bought albums online in past 12 months?* | | | | |
| Never | 7 | 39% | 1 | 8% |
| 1–3 | 6 | 33% | 3 | 25% |
| 4–6 | 3 | 17% | 6 | 50% |
| 7–11 | 1 | 6% | 0 | 0% |
| ≥ 12 | 1 | 6% | 2 | 17% |
| *How did you buy those albums?* | | | | |
| Web | 11 | 61% | 10 | 83% |
| Mobile phone display | 0 | 0% | 0 | 0% |
| Automated voice | 0 | 0% | 0 | 0% |
| Other | 1 | 6% | 0 | 0% |
| No answer | 0 | 0% | 1 | 6% |

For each design, we asked judges to evaluate the information architecture and the page layout for finding a product, checking out a shopping cart, and overall. We also asked how skilled they think the designer is and how complete the design is (see Appendix F for the actual questionnaire and the raw answers).

The judges did not look at the original Damask designs. Instead, we converted the sketches to "sketchy"-looking HTML. This removed the participants' sketching and handwriting abilities as a confounding factor. We also added a collapsible site map to the left-hand side, which shows the titles of all of the pages in the site and the current page's title in bold (see Figure 6-4 and Appendix D).

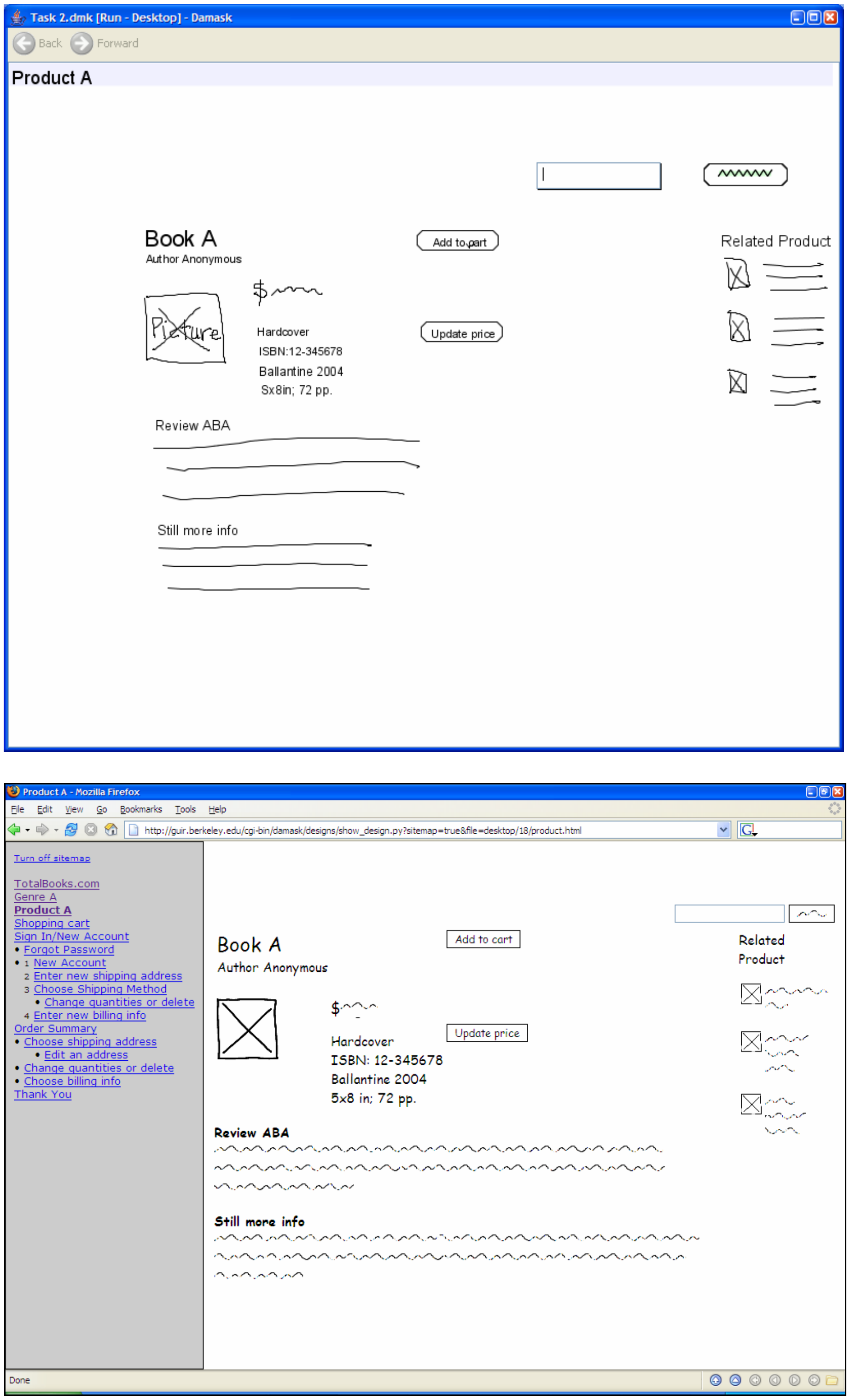


**FIGURE 6-4** A desktop Damask design (above) and its cleaned-up HTML version (below) used in the online evaluation.

We wanted to find out if there was an overall effect of the availability of patterns on the answers to the quantitative questions in the questionnaire (questions 1, 3, 5, 7, and 9-11). First, for each condition, designer, and quantitative question, we calculated the mean of the responses from the three evaluators. We then conducted a multivariate analysis of variance (MANOVA) to determine the effect of the availability of patterns on the answers to the quantitative questions in the questionnaire. The availability of patterns and layers was the independent variable, and each question was a dependent variable. We found no statistically significant effect (Wilks' $\lambda = .62$, $F_{7,16} = 1.39$, $p = 0.28$).

However, we noticed that for each question, the designs with patterns were rated higher than the designs without patterns, on average. To see if any of the questions had significant differences, we conducted analyses of variances (ANOVA) on each question. Using Fisher's PLSD method, each ANOVA was tested at the 0.05 level. The ANOVAs were statistically significant for two of the questions (ratings for the page layout and design of the shopping cart and checkout process, and completeness of the design) and not significant for the other five. See Table 6.11.

**TABLE 6.11** Average ratings for the desktop designs (1 = low, 5 = high), along with *F* and *p* values from ANOVAs.

| Question | No patterns | Patterns | $F_{1,22}$ | *p* |
|---|---|---|---|---|
| 1 Please give a rating for how well the pages for browsing for a (CD/book) were *linked together.* | 3.02 | 3.11 | 0.12 | 0.73 |
| 3 Please give a rating for the page *layout and design* for (CD/book) browsing. | 2.72 | 3.11 | 2.42 | 0.13 |
| 5 Please give a rating for how well the pages for shopping cart and checkout were *linked together*. | 3.14 | 3.36 | 0.58 | 0.46 |
| 7 Please give a rating for the page *layout and design* for shopping cart and checkout. | 2.53 | 3.22 | 6.12 | 0.02 |
| 9 Please give an overall rating. | 2.86 | 3.33 | 2.85 | 0.11 |
| 10 How complete do you consider this design to be? | 2.44 | 3.06 | 5.81 | 0.02 |
| 11 How skilled do you think this designer is? | 2.69 | 3.02 | 1.38 | 0.25 |

We then did the same analysis without the two designers that used one or no patterns, Designers 8 and 16, so that we could see the effect of pattern usage, as opposed to just exposure to patterns. Overall there was still no overall statistically significant effect from the availability of patterns (MANOVA, Wilks' $\lambda$ = .53, $F_{7,12}$ = 1.54, $p$ = 0.24). However, after running ANOVAs on each question, we found statistically significant differences for *three* of them: ratings for the page layout and design of the shopping cart and checkout process, and completeness of the design (as above), and the overall rating. See Table 6.12.

**TABLE 6.12** Average ratings for the desktop designs (1 = low, 5 = high), excluding Designers 8 and 16, along with *F* and *p* values from ANOVAs.

| Question | | No patterns | Patterns | $F_{1,18}$ | *p* |
|---|---|---|---|---|---|
| 1 | Please give a rating for how well the pages for browsing for a (CD/book) were *linked together*. | 2.87 | 3.12 | 1.87 | 0.19 |
| 3 | Please give a rating for the page *layout and design* for (CD/book) browsing. | 2.63 | 3.13 | 3.66 | 0.07 |
| 5 | Please give a rating for how well the pages for shopping cart and checkout were *linked together*. | 3.03 | 3.40 | 1.35 | 0.26 |
| 7 | Please give a rating for the page *layout and design* for shopping cart and checkout. | 2.43 | 3.30 | 8.33 | <0.01 |
| 9 | Please give an overall rating. | 2.73 | 3.37 | 5.55 | 0.03 |
| 10 | How complete do you consider this design to be? | 2.33 | 3.17 | 9.96 | <0.01 |
| 11 | How skilled do you think this designer is? | 2.63 | 3.13 | 2.76 | 0.11 |

Overall, the results of the questionnaire show that the design patterns had their biggest effect on two areas. The fact that the layout and designs of the pages in the shopping cart and checkout process were rated higher is not surprising. It is difficult to adequately design and layout shopping cart and checkout pages that handle changing the quantities in a cart, shipping options, shipping and billing addresses, credit card numbers, and so on in only a couple of hours. Designers using the Damask shopping cart and checkout patterns were able to add this functionality in very little time.

For example, two of the designs that Evaluator 16 evaluated were Designer 13's desktop designs. For the design without layers or patterns, he said about the shopping cart and checkout, "The ordering of items on the page doesn't seem to work that well," and goes on to make detailed suggestions on how to improve it. In contrast, for the design with layers and patterns, he said, "This is really good. The flow is sensible, and the final invoice-like form that you accept works really well. It would seem this design has done a better job at understanding the details of information flow on a page."

Similarly, the ease of adding pre-built functionality from patterns contributed to the higher completeness rating for the designs with patterns. This reinforces the completeness metrics that were described in Section 6.4.2.

However, patterns did not make a significant difference on how well the shopping cart/checkout pages were linked together. This may be because in the designs without patterns, the shopping cart/checkout pages usually were linked into a simple linear structure, which made it as easy to navigate as the more complete shopping cart/checkout pages in the designs with patterns. For example, Evaluator 2 evaluated the desktop designs by Designer 7. He was positive about both the design without patterns ("The checkout process is simple, one click done. But user can't easily add another item or change the cart. The best of all, user doesn't need to go to separate page to enter the address and credit card information.") and with patterns ("Very good check out process. Can change the address and credit card in one page. Can navigate to continue shopping.").

Patterns also did not make a significant difference on how well the product browsing pages were linked together. This is probably because the relevant Damask patterns mainly addressed the content of those pages, not how they are linked.

Patterns also did not make a significant difference on the ratings for the design and layout of the product browsing pages. This may be due to two factors. Our requirements for the product pages were not as extensive as for the shopping cart/checkout pages, making it easier to implement most of the functionality by hand. Also, the relevant Damask patterns originally had placeholder text that described how they should be filled in, such as “95% visitor link,” or UI elements that were not relevant for CDs and books, such as a drop-down menu labeled “Options.” This text often was not changed by the designers, causing the evaluators to become confused.

### 6.5.3 Functional analysis of desktop designs

We also looked at what features were implemented in the checkout process of the desktop designs, to see how they differed between the patterns and non-patterns conditions. We looked for several features that are standard across major e-commerce web sites and observed how many of the designs with and without patterns included them. We also made the same observations for the ten e-commerce web sites that had the most traffic during the week before Christmas 2004 according to Hitwise [68], and the five most usable e-commerce sites according to Keynote Systems [87], excluding those sites that did not have a checkout process (such as eBay and Yahoo Shopping).

We found that the designs with patterns were more likely to include these features than the designs without patterns, except for two of the features (see Table 6.13). None of the designs created in the patterns condition included the ability to edit the billing address or the payment information. This is because both of these features were unintentionally omitted from the pattern solutions, yet another example of how influential patterns were in the patterns condition.

**TABLE 6.13** Common features in the checkout process.

| | Non-Pattern (out of 12) | | Pattern (out of 12) | | Top 5 Usable (out of 4) | | Top 10 Popular (out of 7) | |
|---|---|---|---|---|---|---|---|---|
| Separate shopping cart | 10 | 83% | 10 | 83% | 4 | 100% | 7 | 100% |
| Order summary | 3 | 25% | 9 | 75% | 4 | 100% | 7 | 100% |
| Change shipping method | 9 | 75% | 11 | 92% | 4 | 100% | 7 | 100% |
| Change quantities during checkout | 0 | 0% | 7 | 58% | 4 | 100% | 4 | 57% |
| Edit shipping address | 1 | 8% | 10 | 83% | 4 | 100% | 7 | 100% |
| Edit billing address | 1 | 8% | 0 | 0% | 3 | 75% | 5 | 71% |
| Edit payment information | 1 | 8% | 0 | 0% | 3 | 75% | 5 | 71% |
| Create new shipping address | 8 | 67% | 11 | 92% | 3 | 75% | 5 | 71% |
| Create new billing address | 7 | 58% | 10 | 83% | 2 | 50% | 4 | 57% |
| Create new payment information | 9 | 75% | 10 | 83% | 2 | 50% | 4 | 57% |
| *Average* | *4.9* | *41%* | *7.8* | *65%* | *3.3* | *83%* | *5.5* | *79%* |

### 6.5.4 *Structural analysis of desktop designs*

To further characterize the desktop designs, we analyzed the designs' functionality and flow to see how they compare to that of popular, high-usability commercial web sites. To do this, we created abstract sitemaps that captured the functionality of each page and the flow between pages (see Figure 6-6 and Figure 6-7 and the key for the symbols in Figure 6-5).

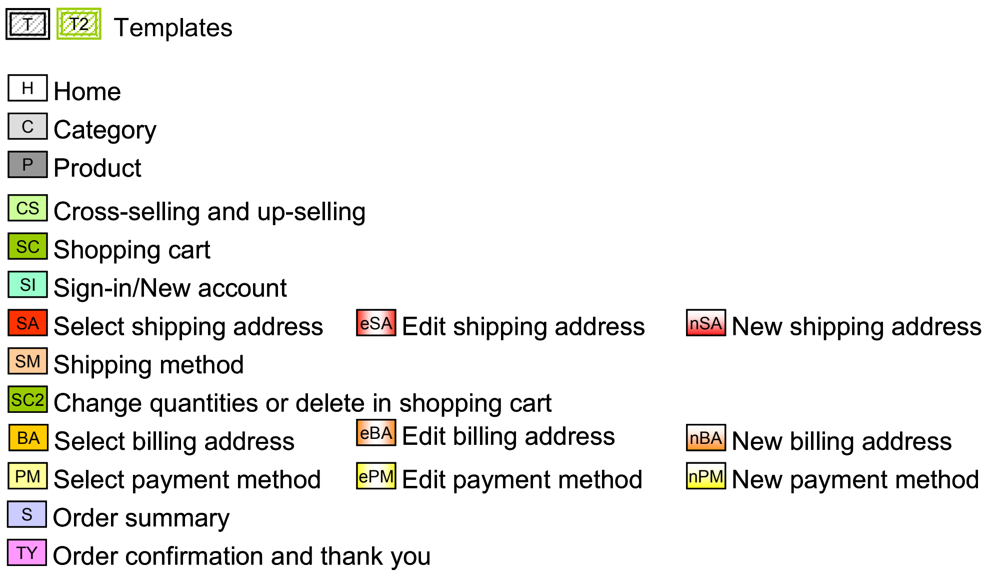


**FIGURE 6-5** The key to the symbols used in the abstract sitemaps.

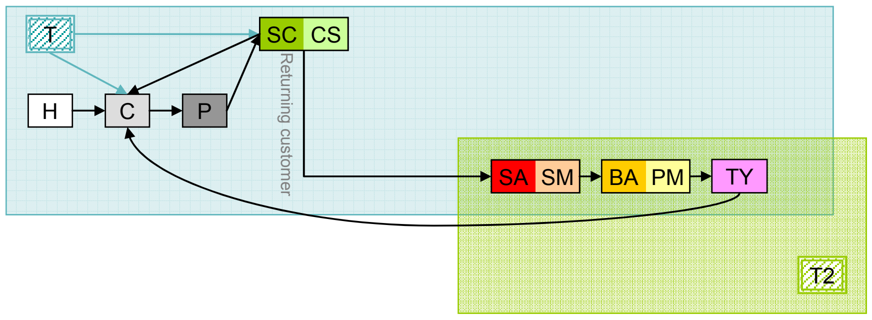


**FIGURE 6-6** The abstract sitemap for Participant 6's TotalMusic.com design for the desktop, shown in Figure 6-2, which was created without layers or patterns.Figure **6-2** Participant 6's desktop design for TotalMusic.com, created without patterns or layers.

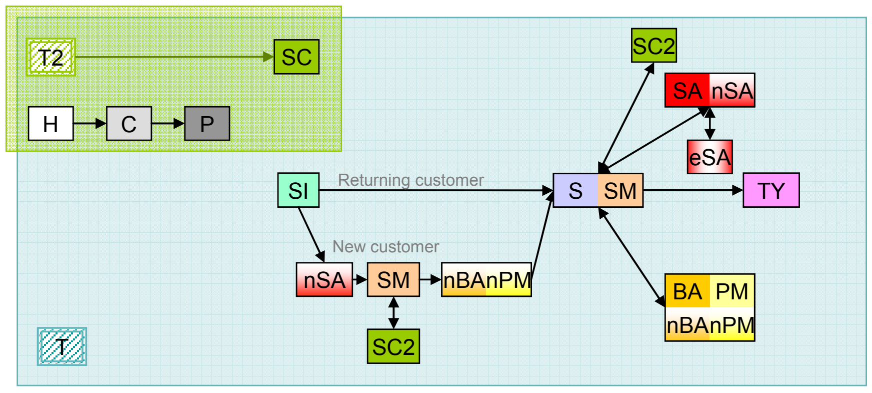


**FIGURE 6-7** The abstract sitemap for Participant 6's TotalBooks.com design for the desktop, shown in Figure 6-3.

Then we also created sitemaps that focused on the checkout processes of the web sites of Amazon and Barnes & Noble, which were the top two e-commerce sites in terms of usability, according to online usability studies run by Keynote Systems [87] (see Figure 6-8). The sitemaps primarily covered the checkout process because we had asked the designers to focus more on the checkout process than, say, browsing or searching for products.

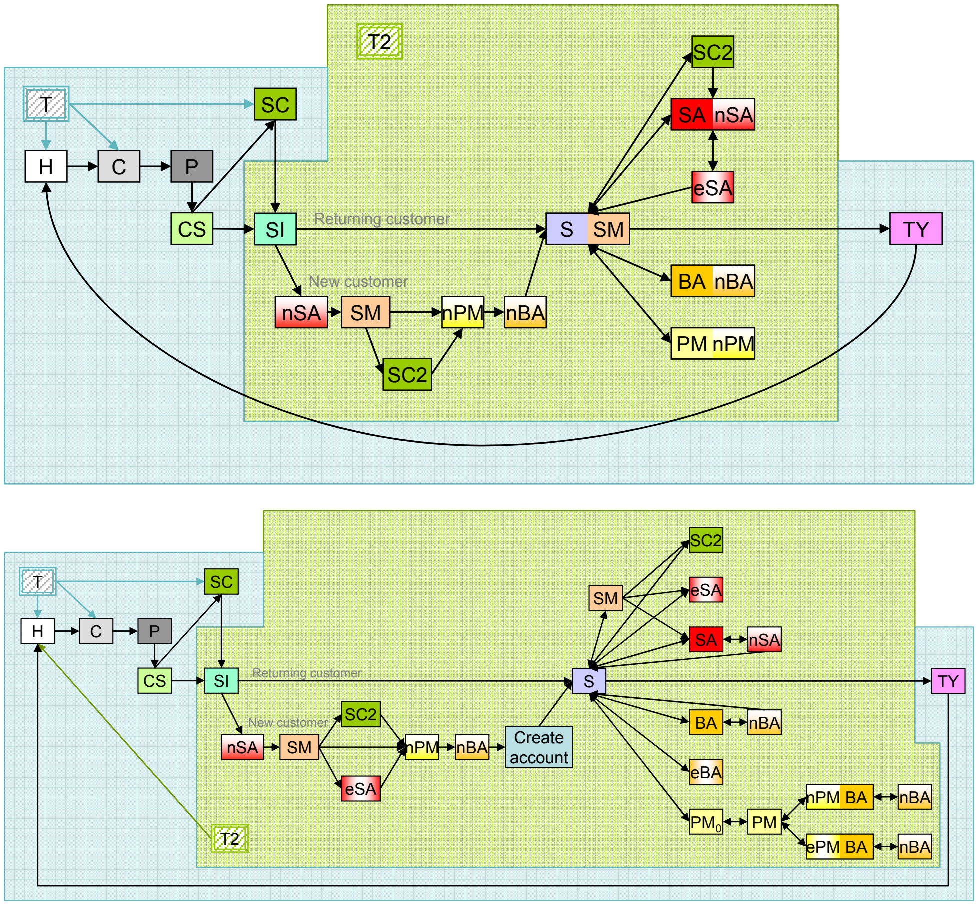


**FIGURE 6-8** The abstract sitemaps for Amazon.com (top) and BarnesAndNoble.com (bottom).

We removed the "New Customer" branch from all of the sitemaps, because our design instructions said the participants could assume the user already had an account with the e-commerce site. We then counted how many editing steps it would take to transform a sitemap of a design that a participant created into the sitemaps of Amazon and Barnes & Noble. Here are the possible types of edits:

- Creating a template
- Apply or unapplying a template to a page
- Adding or removing a link that originates from a normal page
- Adding or removing a link that originates from a template
- Adding or removing a page
- Merging or splitting pages

The fewer the steps, the closer the structure of the design is to that of a widely-accepted good design (see Figure 6-9). In graph theory, this metric is called the *graph edit distance*. See Appendix D for the sitemaps and the actual designs of the participants, and Appendix E for those of Amazon and Barnes & Noble.

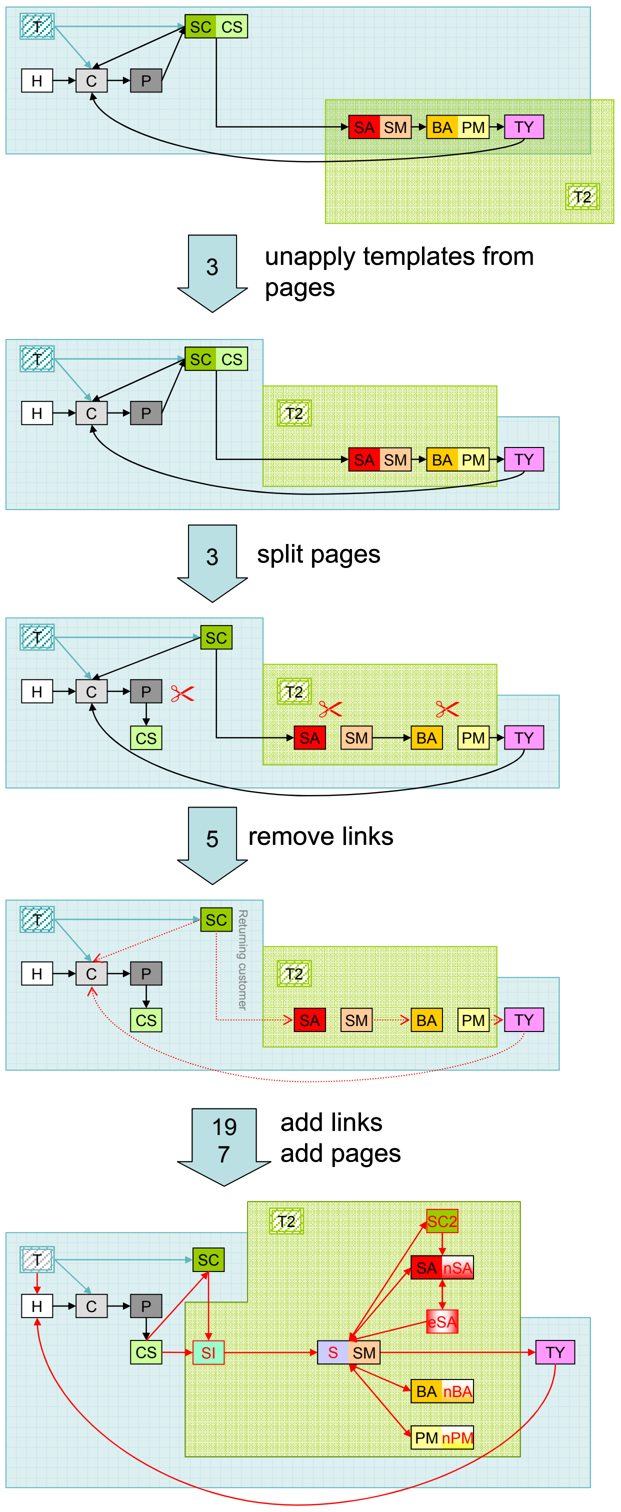


**FIGURE 6-9** Transforming the design shown in Participant 6's TotalMusic.com to the Amazon.com sitemap.

To see whether the patterns had an effect on the number of steps across types, we conducted multivariate analyses of variance (MANOVAs) where the availability of patterns and layers was the independent variable, and each edit type was a dependent variable (except for removing pages, which was 0 in every case). In both Amazon and Barnes & Noble, we found no overall statistically significant effect when we included all designs. However, if we exclude the outliers, we see that there is a significant effect on Barnes & Noble, and an effect that is close to significant on Amazon (see Table 6.14).

**TABLE 6.14** Results of MANOVAs on edit distances for desktop designs.

| | **Amazon** | **Barnes & Noble** |
|---|---|---|
| **All designs** | Wilks' $\lambda = 0.50$<br>$F_{9,14} = 1.55$<br>$p = 0.22$ | Wilks' $\lambda = 0.55$<br>$F_{9,14} = 1.28$<br>$p = 0.33$ |
| **All except Designers 8, 16** | Wilks' $\lambda = 0.27$<br>$F_{9,10} = 3.01$<br>$p = 0.05$ | Wilks' $\lambda = 0.25$<br>$F_{9,10} = 3.40$<br>$p = 0.03$ |

To see which types of edits had significant differences, we conducted analyses of variances (ANOVA) on each edit. Using Fisher's PLSD method, each ANOVA was tested at the 0.05 level. We did this over all of the designs (see Table 6.15) and all of the designs except Designers 8 and 16 (see Table 6.16).

**TABLE 6.15** Average edit distances for the desktop designs, along with *F* and *p* values from ANOVAS (NP = no patterns, P = patterns).

| Edit type | Amazon | | | | Barnes & Noble | | | |
|---|---|---|---|---|---|---|---|---|
| | NP | P | $F_{1,22}$ | *p* | NP | P | $F_{1,22}$ | *p* |
| Unapply template | 1.17 | 1.50 | 0.16 | 0.69 | 1.17 | 1.50 | 0.16 | 0.69 |
| New template | 1.08 | 1.08 | 0.00 | 1.00 | 1.08 | 1.08 | 0.00 | 1.00 |
| Apply template | 3.25 | 5.33 | 3.00 | 0.10 | 3.25 | 5.58 | 3.65 | 0.07 |
| Split pages | 2.33 | 1.25 | 7.41 | 0.01 | 3.83 | 4.25 | 0.52 | 0.48 |
| Remove template links | 0.17 | 0.33 | 0.58 | 0.45 | 0.17 | 0.33 | 0.58 | 0.45 |
| Remove normal links | 3.67 | 2.50 | 2.31 | 0.14 | 3.83 | 4.00 | 0.05 | 0.82 |
| Add template links | 2.08 | 2.17 | 0.06 | 0.80 | 3.33 | 3.08 | 0.53 | 0.47 |
| Add normal links | 17.00 | 11.83 | 17.3 | <0.01 | 38.59 | 34.59 | 11.88 | <0.01 |
| Add pages | 6.17 | 3.08 | 9.91 | <0.01 | 13.08 | 9.92 | 11.43 | <0.01 |

**TABLE 6.16** Average edit distances for the desktop designs, without the outliers from Designers 8 and 16, along with *F* and *p* values from ANOVAS (NP = no patterns, P = patterns).

| Edit type | Amazon | | | | Barnes & Noble | | | |
|---|---|---|---|---|---|---|---|---|
| | NP | P | $F_{1,18}$ | *p* | NP | P | $F_{1,18}$ | *p* |
| Unapply template | 1.30 | 1.70 | 0.17 | 0.68 | 1.30 | 1.70 | 0.17 | 0.68 |
| New template | 1.00 | 1.00 | 0.00 | 1.00 | 1.00 | 1.00 | 0.00 | 1.00 |
| Apply template | 2.90 | 5.70 | 5.39 | 0.03 | 2.90 | 6.00 | 6.41 | 0.02 |
| Split pages | 2.20 | 1.00 | 9.53 | 0.01 | 3.90 | 4.40 | 0.68 | 0.42 |
| Remove template links | 0.10 | 0.30 | 0.72 | 0.41 | 0.10 | 0.30 | 0.72 | 0.41 |
| Remove normal links | 3.60 | 2.10 | 3.01 | 0.10 | 3.80 | 4.10 | 0.13 | 0.73 |
| Add template links | 2.20 | 2.20 | 0.00 | 1.00 | 3.60 | 3.10 | 3.08 | 0.10 |
| Add normal links | 17.00 | 10.60 | 33.39 | <0.01 | 38.60 | 33.50 | 22.75 | <0.01 |
| Add pages | 6.10 | 2.10 | 26.77 | <0.01 | 13.00 | 8.90 | 38.89 | <0.01 |

If we look at the individual steps, we see that there is a significant difference for adding a page and adding normal links in all cases, for splitting a page in Amazon only, and applying a template only when outliers are not considered. We will now describe each in detail.

- *Adding a page.* On average, fewer pages were added to a design with patterns than a design without patterns. This indicates that the designs with patterns were more likely to have the same features as Amazon and Barnes & Noble than designs without patterns. This backs up the discussion in the previous section on functional analysis.
- *Adding a link that originates from a normal page (as opposed to a template).* On average, fewer links were added to a design with patterns than a design without patterns. This implies that designs with patterns have a more similar structure to Amazon and Barnes & Noble than designs without patterns.
- *Applying a template to a page.* Interestingly, templates were applied to *more* pages in the designs with patterns than those without. This is due to two factors. The patterns in Damask do not give much guidance on how and when to apply templates, so the designs with patterns were as likely as those without patterns to use templates differently than in Amazon and Barnes & Noble. Also, the designs with patterns had more pages than those without, so when a template needed to be applied to match Amazon and Barnes & Noble, there were more pages that need to be "fixed" in the designs with patterns than those without.
- In addition, in the case of Amazon.com, fewer pages were split (statistically significantly) in the designs with patterns than those without. This means that the contents of a page in the designs with patterns were more likely to match those of Amazon. This is not surprising since many of the patterns were inspired by Amazon.

This metric is complicated by the fact that not all of the features that Amazon and Barnes & Noble have in their checkout processes were required in our design tasks. These extra features include the ability to change the quantities of the items in your shopping cart during the checkout process, and the ability to edit your shipping and billing address. Therefore, we removed the optional features from the participants' sitemaps and the target sitemaps, and recounted. We then computed more MANOVAs, but they could not be computed, because of the characteristics of the data (the mean, standard deviation, and standard error for designs without patterns for three of the edit types is 0). Looking at the data, it is unlikely there is an overall statistically significant difference.

We conducted analyses of variances (ANOVA) on each edit to see which of them had significant differences. Using Fisher's PLSD method, each ANOVA was tested at the 0.05 level. We did this over all of the designs (see Table 6.17) and all of the designs except Designers 8 and 16 (see Table 6.18).

From this, we see that there are significant differences in roughly the same types of edits as in the measurements including the optional features: splitting a page, adding a normal link, and adding a page. We believe this is for the reasons stated above—that patterns do not currently give guidance about templates, and that designs with patterns have more pages that are affected by templates.

**TABLE 6.17** Average edit distances for the desktop designs, excluding optional features, along with *F* and *p* values from ANOVAS (NP = no patterns, P = patterns).

| Edit type | Amazon | | | | Barnes & Noble | | | |
|---|---|---|---|---|---|---|---|---|
| | NP | P | $F_{1,22}$ | *p* | NP | P | $F_{1,22}$ | *p* |
| Unapply template | 0.00 | 0.50 | 1.00 | 0.33 | 0.00 | 0.50 | 1.00 | 0.33 |
| New template | 0.00 | 0.08 | 1.00 | 0.33 | 0.00 | 0.08 | 1.00 | 0.33 |
| Apply template | 0.00 | 0.58 | 1.36 | 0.26 | 0.00 | 0.83 | 2.90 | 0.10 |
| Split pages | 2.33 | 1.25 | 7.41 | 0.01 | 3.83 | 4.25 | 0.52 | 0.48 |
| Remove template links | 0.08 | 0.33 | 1.48 | 0.24 | 0.08 | 0.33 | 1.48 | 0.24 |
| Remove normal links | 3.33 | 2.50 | 1.24 | 0.28 | 3.33 | 4.00 | 1.11 | 0.30 |
| Add template links | 1.92 | 2.17 | 0.44 | 0.51 | 3.17 | 3.08 | 0.04 | 0.83 |
| Add normal links | 11.33 | 9.08 | 5.32 | 0.03 | 26.58 | 23.75 | 9.04 | <0.01 |
| Add pages | 4.25 | 2.50 | 4.53 | 0.04 | 7.08 | 5.33 | 4.96 | 0.04 |

**TABLE 6.18** Average edit distances for the desktop designs, excluding optional features, without the outliers from Designers 8 and 16, along with *F* and *p* values from ANOVAS (NP = no patterns, P = patterns).

| Edit type | Amazon | | | | Barnes & Noble | | | |
|---|---|---|---|---|---|---|---|---|
| | NP | P | $F_{1,18}$ | *p* | NP | P | $F_{1,18}$ | *p* |
| Unapply template | 0.00 | 0.60 | 1.00 | 0.33 | 0.00 | 0.60 | 1.00 | 0.33 |
| New template | 0.00 | 0.10 | 1.00 | 0.33 | 0.00 | 0.10 | 1.00 | 0.33 |
| Apply template | 0.00 | 0.70 | 1.37 | 0.26 | 0.00 | 1.00 | 3.00 | 0.10 |
| Split pages | 2.20 | 1.10 | 9.53 | <0.01 | 3.90 | 4.40 | 0.68 | 0.42 |
| Remove template links | 0.10 | 0.30 | 0.72 | 0.41 | 0.10 | 0.30 | 0.72 | 0.41 |
| Remove normal links | 3.20 | 2.10 | 1.74 | 0.20 | 3.20 | 4.10 | 1.57 | 0.23 |
| Add template links | 2.10 | 2.20 | 0.14 | 0.71 | 3.50 | 3.10 | 1.95 | 0.18 |
| Add normal links | 11.20 | 8.20 | 11.07 | <0.01 | 26.60 | 22.90 | 17.78 | <0.01 |
| Add pages | 4.20 | 1.80 | 11.47 | <0.01 | 7.00 | 4.60 | 15.07 | <0.01 |

## 6.6 Smartphone Designs Created by the Participants

We analyzed the smartphone designs by asking other user interface designers to evaluate them. Unlike with the desktop designs, we did not compare them to mobile commerce (“m-commerce”) web sites, because there are no m-commerce sites at which one can buy books or music that have features comparable to those created by our participants.

### 6.6.1 *Examples of smartphone designs*

Figure 6-10 shows a typical TotalMusic.com design created by Participant 7 without patterns or layers. Figure 6-11 shows a TotalBooks.com design created with patterns and layers by the same participant, after she had created the TotalMusic.com design.

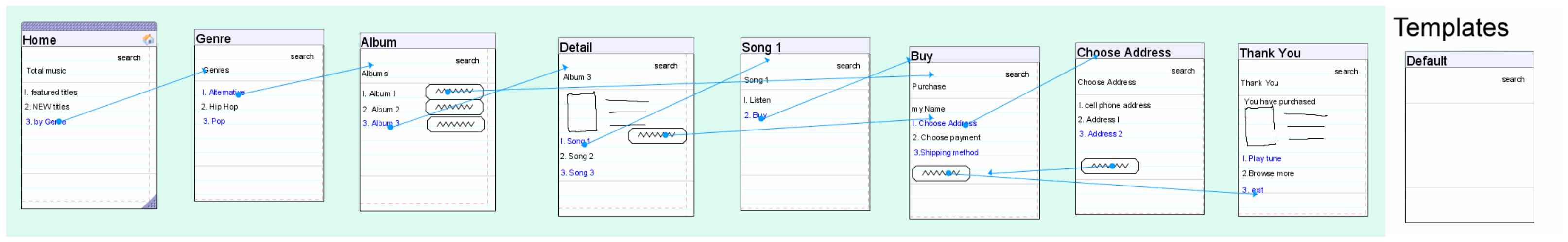


**FIGURE 6-10** Participant 7's smartphone design for TotalMusic.com, made without patterns or layers.

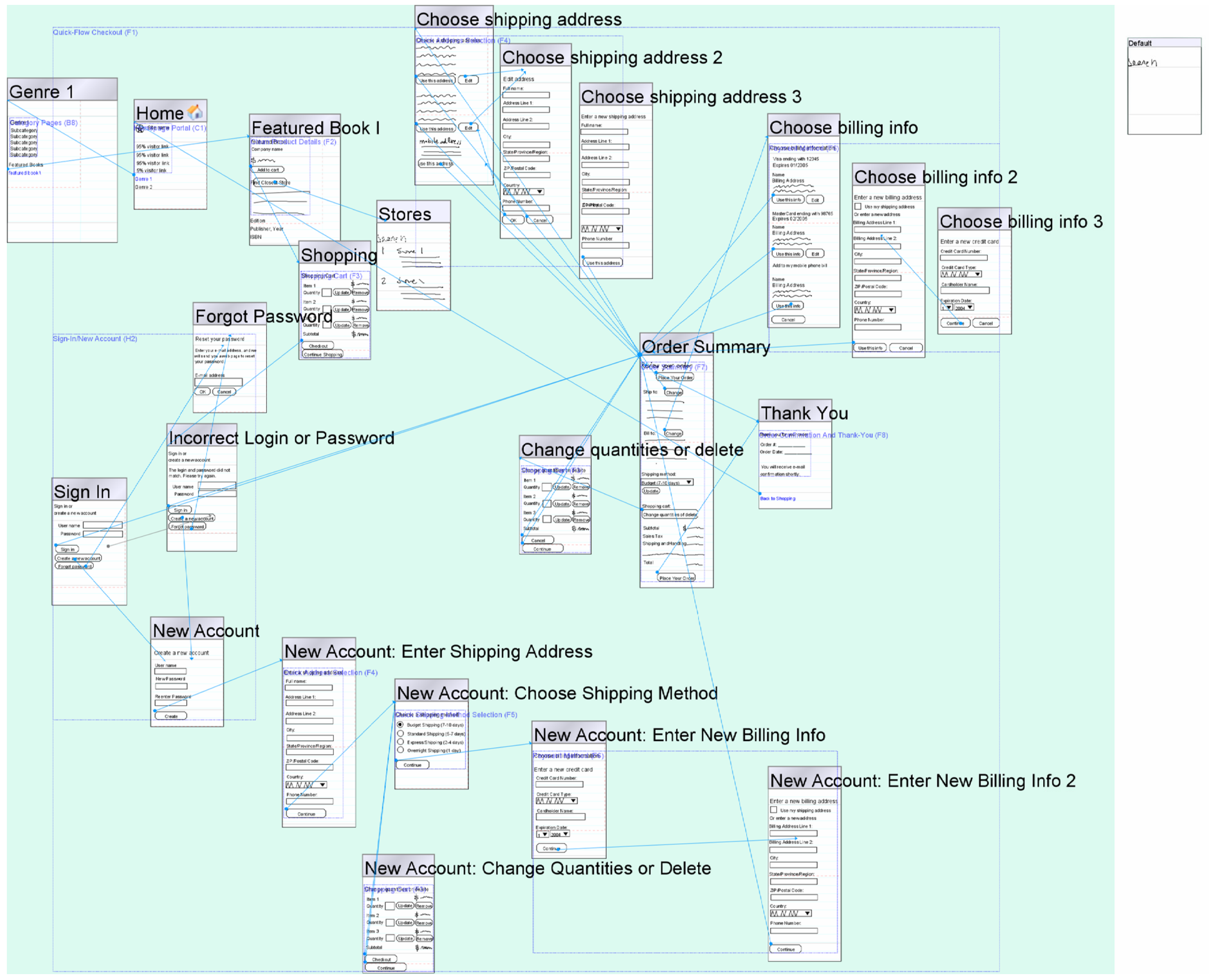

Quick-Flow Checkout (F1)
Sign-In/New Account (H2)
Default
Genre 1
Home
Featured Book I
Stores
Shopping
Forgot Password
Incorrect Login or Password
Sign In
New Account
Choose shipping address
Choose shipping address 2
Choose shipping address 3
Choose billing info
Choose billing info 2
Choose billing info 3
Order Summary
Thank You
Change quantities or delete
New Account: Enter Shipping Address
New Account: Choose Shipping Method
New Account: Enter New Billing Info
New Account: Enter New Billing Info 2
New Account: Change Quantities or Delete

**FIGURE 6-11** Participant 7's smartphone design for TotalBooks.com, made with patterns and layers.

Once again, the TotalBooks.com design has more detail and features but is harder to follow than the TotalMusic.com design. Participant 7 believed that in the limited time she had, "TotalBooks is the better design. I'm sure I missed some stuff in TotalMusic." As for the patterns, she found the amount of detail useful. "At some point I'd need all of that stuff, and I think it'd be easier to delete screens than remember to add them in, especially at this stage of the design."

### 6.6.2 *Quality analysis of smartphone designs*

We performed an online evaluation of the quality of the smartphone designs, using the same questionnaire and method as for the desktop designs, described in Section 6.5.2. There are 16 smartphone designs over the two conditions. We recruited 12 professional designers online and randomly assigned four designs to each evaluator, so that each design was evaluated by three people. The four designs came from two designers (patterns and non-patterns), and the order of the four designs was randomized.

Table 6.19 shows the demographics of the smartphone evaluators, along with those of the smartphone designers (Designers 4–11) for comparison. Overall, the evaluators are somewhat older on average and have more experience designing for mobile phones and buying ringtones. See Appendix F, Questions 13–30 for the raw questionnaire data.

**TABLE 6.19** Demographics of the evaluators of the smartphone designs. The demographics of the smartphone designers (Designers 4–11) are included for comparison.

| | **Evaluators** | | | Designers | | |
|---|---|---|---|---|---|---|
| Total | | 12 | | | 8 | |
| *Gender* | | | | | | |
| Male | | 8 | 67% | | 4 | 50% |
| Female | | 4 | 33% | | 4 | 50% |
| *Age* | | | | | | |
| | 19 and under | 0 | 0% | under 20 | 0 | 0% |
| | 20-29 | 3 | 25% | 21-30 | 4 | 50% |
| | 30-39 | 5 | 42% | 31-40 | 3 | 38% |
| | 40-49 | 3 | 25% | 41-50 | 1 | 13% |
| | 50 or above | 1 | 8% | 51 or above | 0 | 0% |
| *Years of design experience* | | | | | | |
| < 1 | | 0 | 0% | | 0 | 0% |
| 1–2 | | 0 | 0% | | 0 | 0% |
| 3–4 | | 0 | 0% | | 0 | 0% |
| ≥ 5 | | 12 | 100% | | 8 | 100% |
| *Years of web design experience* | | | | | | |
| < 1 | | 0 | 0% | | 0 | 0% |
| 1–2 | | 3 | 25% | | 0 | 0% |
| 3–4 | | 0 | 0% | | 0 | 0% |
| ≥ 5 | | 9 | 75% | | 8 | 100% |
| *Years of mobile phone design experience* | | | | | | |
| 0 | | 0 | 0% | | 1 | 13% |
| < 1 | | 0 | 0% | | 5 | 63% |
| 1–2 | | 9 | 75% | | 2 | 25% |
| 3–4 | | 1 | 8% | | 0 | 0% |
| ≥ 5 | | 2 | 17% | | 0 | 0% |
| *Number of e-commerce web sites designed* | | | | | | |
| 0 | | 1 | 8% | | 0 | 0% |
| 1–2 | | 3 | 25% | | 4 | 50% |
| 3–4 | | 4 | 33% | | 2 | 25% |
| ≥ 5 | | 1 | 8% | | 2 | 25% |
| *Which platforms were the e-commerce sites designed for?* | | | | | | |
| Web | | 11 | 92% | | 8 | 100% |
| PDA or mobile phone | | 4 | 33% | | 2 | 25% |
| Voice | | 1 | 8% | | 0 | 0% |
| Other | | 0 | 0% | | 0 | 0% |

| | **Evaluators** | | Designers | |
|---|---|---|---|---|
| *Design knowledge* | | | | |
| IA | 10 | 83% | 6 | 75% |
| Graphic design | 5 | 42% | 4 | 50% |
| Web design | 11 | 92% | 8 | 100% |
| GUI design | 7 | 58% | 7 | 88% |
| Mobile phone UI design | 4 | 33% | 1 | 13% |
| Voice UI design | 3 | 25% | 0 | 0% |
| Other | 1 | 8% | 3 | 38% |
| *How many times have you bought anything online in past 12 months?* | | | | |
| Never | 0 | 0% | 0 | 0% |
| 1–3 | 0 | 0% | 0 | 0% |
| 4–6 | 2 | 17% | 0 | 0% |
| 7–11 | 2 | 17% | 2 | 25% |
| ≥ 12 | 8 | 67% | 6 | 75% |
| *How did you buy those items?* | | | | |
| Web | 12 | 100% | 7 | 88% |
| Mobile phone display | 2 | 17% | 0 | 0% |
| Automated voice | 3 | 25% | 0 | 0% |
| Other | 0 | 0% | 0 | 0% |
| No answer | 0 | 0% | 1 | 13% |
| *How many times have you bought books online in past 12 months?* | | | | |
| Never | 0 | 0% | 0 | 0% |
| 1–3 | 5 | 42% | 3 | 38% |
| 4–6 | 5 | 42% | 1 | 13% |
| 7–11 | 0 | 0% | 2 | 25% |
| ≥ 12 | 2 | 17% | 2 | 25% |
| *How did you buy those books?* | | | | |
| Web | 12 | 100% | 8 | 100% |
| Mobile phone display | 0 | 0% | 0 | 0% |
| Automated voice | 0 | 0% | 0 | 0% |
| Other | 0 | 0% | 0 | 0% |

| | **Evaluators** | | Designers | |
|---|---|---|---|---|
| *How many times have you bought albums online in past 12 months?* | | | | |
| Never | 3 | 25% | 0 | 0% |
| 1–3 | 4 | 33% | 2 | 25% |
| 4–6 | 3 | 25% | 4 | 50% |
| 7–11 | 1 | 8% | 0 | 0% |
| ≥ 12 | 1 | 8% | 2 | 25% |
| *How did you buy those albums?* | | | | |
| Web | 8 | 67% | 7 | 88% |
| Mobile phone display | 1 | 8% | 0 | 0% |
| Automated voice | 0 | 0% | 0 | 0% |
| Other | 3 | 25% | 0 | 0% |
| No answer | 0 | 0% | 1 | 13% |
| *How many times have you bought ringtones online in past 12 months?* | | | | |
| Never | 7 | 58% | 6 | 75% |
| 1–3 | 4 | 33% | 2 | 25% |
| 4–6 | 1 | 8% | 0 | 0% |
| 7–11 | 0 | 0% | 0 | 0% |
| ≥ 12 | 0 | 0% | 0 | 0% |
| *How did you buy those ringtones?* | | | | |
| Web | 4 | 33% | 2 | 25% |
| Mobile phone display | 5 | 42% | 0 | 0% |
| Automated voice | 0 | 0% | 0 | 0% |
| Other | 0 | 0% | 0 | 0% |

Once again, for each design, we asked judges to evaluate the information architecture and the page layout for finding a product, checking out a shopping cart, and overall. We also asked how skilled they think the designer is and how complete the design is (see Appendix F for the questions and the raw answers). The judges judged “sketchy”-looking HTML versions of the designs (see Figure 6-12 and Appendix D).

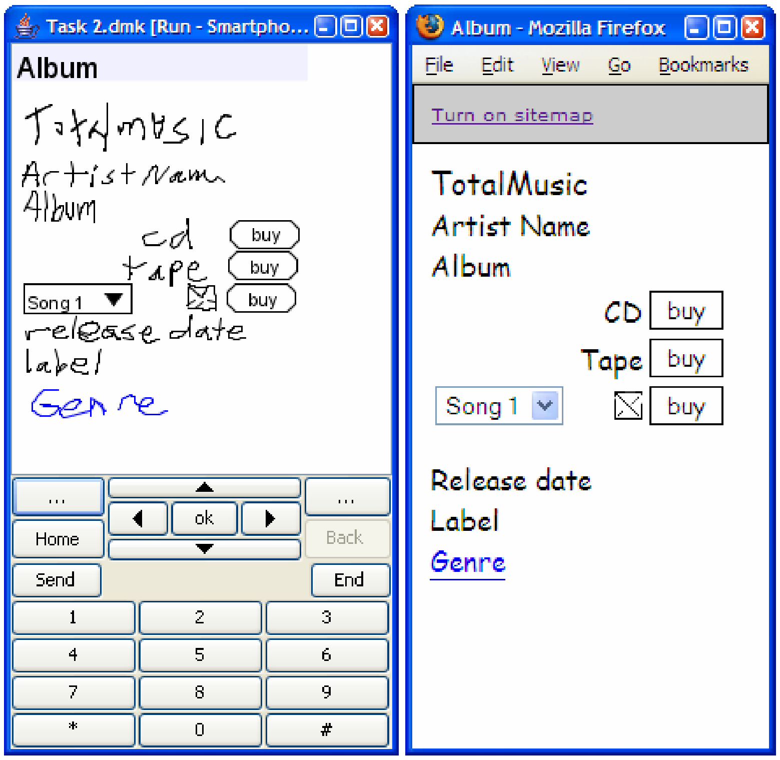


**FIGURE 6-12** A smartphone Damask design (left) and its cleaned-up HTML version (right) used in the online evaluation.

To analyze the questionnaire results, we first calculated the mean of the responses from the three evaluators, for each condition, designer, and quantitative question. We conducted a multivariate analysis of variance (MANOVA) to determine the effect of the availability of patterns on the answers to the quantitative questions in the questionnaire. The availability of patterns was the independent variable, and each question was a dependent variable. We found no overall statistically significant effect (Wilks' $\lambda = .52$, $F_{7,8} = 1.06$, $p = 0.46$).

To see if any of the questions had significant differences, we conducted analyses of variances (ANOVA) on each question as follow-up tests. Using Fisher's PLSD method, each ANOVA was tested at the 0.05 level. We found no statistically significant differences in any of the questions. See Table 6.20.

**TABLE 6.20** Average ratings for the smartphone designs (1 = low, 5 = high), along with *F* and *p* values from ANOVAS.

| Question | | No patterns | Patterns | $F_{1,14}$ | $p$ |
|---|---|---|---|---|---|
| 1 | Please give a rating for how well the pages for browsing for a (CD/book) were *linked together*. | 3.25 | 2.88 | 1.33 | 0.27 |
| 3 | Please give a rating for the page *layout and design* for (CD/book) browsing. | 2.79 | 2.46 | 0.94 | 0.35 |
| 5 | Please give a rating for how well the pages for shopping cart and checkout were *linked together*. | 3.00 | 3.33 | 0.93 | 0.35 |
| 7 | Please give a rating for the page *layout and design* for shopping cart and checkout. | 2.50 | 3.17 | 3.39 | 0.09 |
| 9 | Please give an overall rating. | 2.75 | 3.17 | 2.01 | 0.18 |
| 10 | How complete do you consider this design to be? | 2.12 | 2.79 | 3.51 | 0.08 |
| 11 | How skilled do you think this designer is? | 2.71 | 2.92 | 0.42 | 0.53 |

We then did the same analysis without the two designers that used one or no patterns, Designers 8 and 16, so that we could see the effect of pattern usage, as opposed to just exposure to patterns. Overall there was still no overall statistically significant effect from the availability of patterns (MANOVA, Wilks' $\lambda = .25$, $F_{7,12} = 2.61$, $p = 0.13$).

After running ANOVAS on each question, we found statistically significant differences for two of them: ratings for the page layout and design of the shopping cart and checkout process, and completeness of the design. See Table 6.21.

**TABLE 6.21** Average ratings for the smartphone designs (1 = low, 5 = high), excluding Designer 8, along with *F* and *p* values from ANOVAS.

| Question | No patterns | Patterns | $F_{1,12}$ | *p* |
|---|---|---|---|---|
| 1 Please give a rating for how well the pages for browsing for a (CD/book) were *linked together*. | 3.33 | 2.81 | 2.17 | 0.17 |
| 3 Please give a rating for the page *layout and design* for (CD/book) browsing. | 2.90 | 2.43 | 1.62 | 0.23 |
| 5 Please give a rating for how well the pages for shopping cart and checkout were *linked together*. | 3.05 | 3.43 | 1.01 | 0.33 |
| 7 Please give a rating for the page *layout and design* for shopping cart and checkout. | 2.48 | 3.33 | 5.37 | 0.04 |
| 9 Please give an overall rating. | 2.76 | 3.29 | 2.86 | 0.12 |
| 10 How complete do you consider this design to be? | 2.14 | 2.95 | 4.90 | 0.05 |
| 11 How skilled do you think this designer is? | 2.71 | 3.00 | 0.63 | 0.44 |

As with the desktop designs, the results of the questionnaire show that the design patterns had their biggest effect on two areas: higher ratings for the layout and designs of the pages in the shopping cart and checkout process, and the higher completeness rating for the designs with patterns. We believe this is due to similar reasons as the desktop designs: that the shopping cart and checkout process is complex enough that using patterns is a big advantage, and that the completeness of the patterns leads to a more complete design overall.

## 6.7 Voice Designs Created by the Participants

Similarly to the smartphone designs, we analyzed the voice designs by asking other user interface experts to evaluate them. We did not compare them to voice commerce ("v-commerce") sites, because there are no v-commerce sites at which one can buy books or music that have features comparable to those created by our participants.

### *6.7.1 Examples of voice designs*

Figure 6-13 shows a typical TotalMusic.com voice UI design created by Participant 15 without patterns or layers. Figure 6-14 shows a TotalBooks.com voice UI design created with patterns and layers by the same participant, prior to creating the TotalMusic.com design. The TotalMusic.com design is more linear, but the prompts and responses are well crafted for the task. The TotalBooks.com design has more features, but the complexity of the layout makes this design harder to grasp. Given the number of arrows in a voice UI design in Damask, an improved layout algorithm would greatly improve the readability of voice designs.

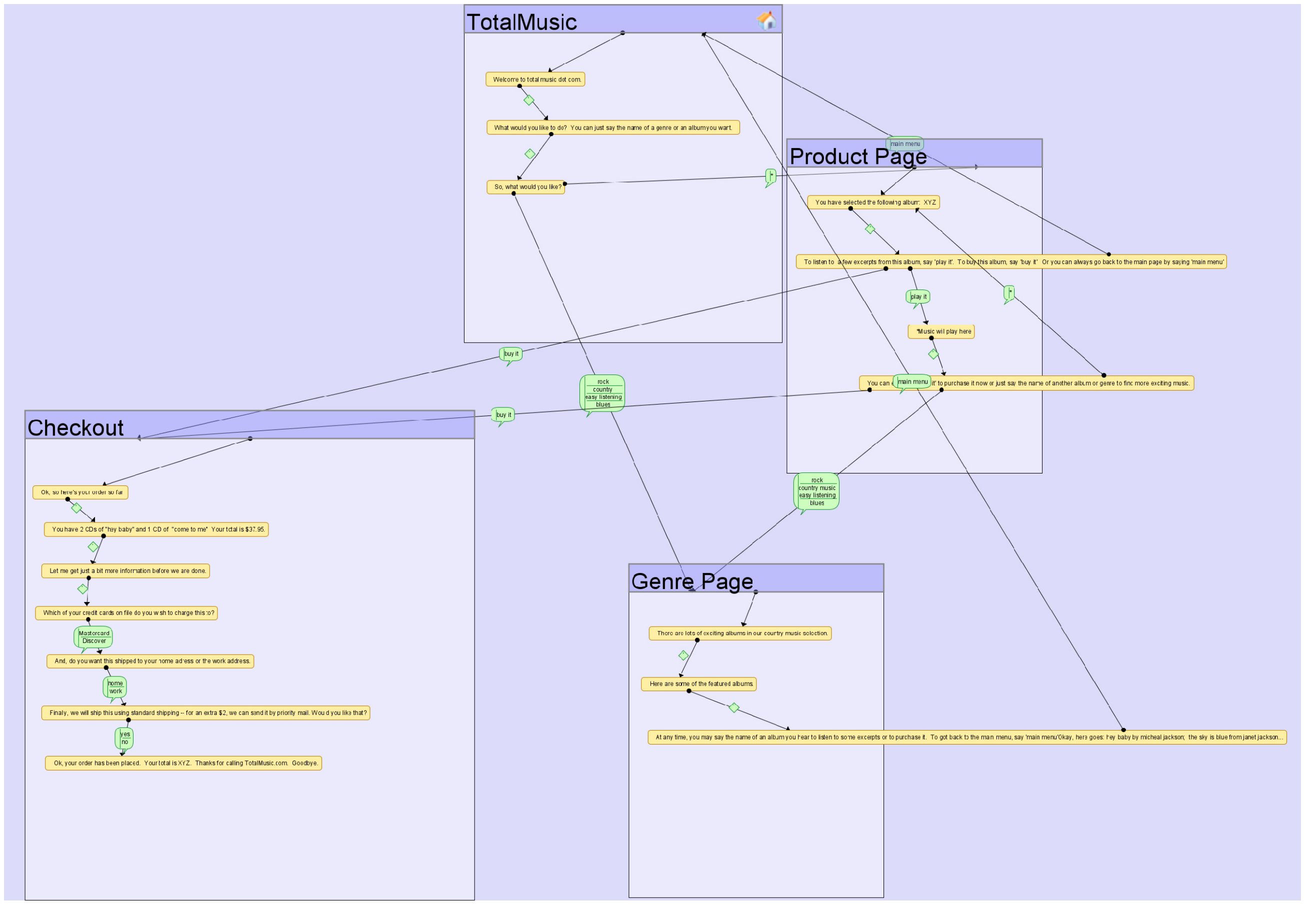

TotalMusic
Welcome to total music dot com.
What would you like to do? You can just say the name of a genre or an album you want.
So, what would you like?
*
rock
country
easy listening
blues
buy it
Product Page
main menu
You have selected the following album: XYZ
To listen to a few excerpts from this album, say 'play it'. To buy this album, say 'buy it' Or you can always go back to the main page by saying 'main menu'
play it
*Music wil play here
*
You can main menu it' to purchase it now or just say the name of another album or genre to finc more exciting music.
buy it
rock
country music
easy listening
blues
Checkout
Ok, so here's your order so far
You have 2 CDs of "hey baby" and 1 CD of "come to me" Your total is $37.95.
Let me get just a bit more information before we are done.
Which of your credit cards on file do you wish to charge this to?
Mastercard
Discover
And, do you want this shipped to your home adress or the work address.
home
work
Finaly, we will ship this using standard shipping -- for an extra $2, we can send it by priority mail. Would you like that?
yes
no
Ok, your order has been placed. Your total is XYZ. Thanks for calling TotalMusic.com. Goodbye.
Genre Page
There are lots of exciting albums in our courtry music selection.
Here are some of the featured albums.
At any time, you may say the name of an album you hear to listen to some excerpts or to purchase it. To got back to the main menu, say 'main menu'Okay, here goes: hey baby by micheal jackson; the sky is blue from janet jackson...

**FIGURE 6-13** Participant 15's voice design for TotalMusic.com, made without patterns or layers.

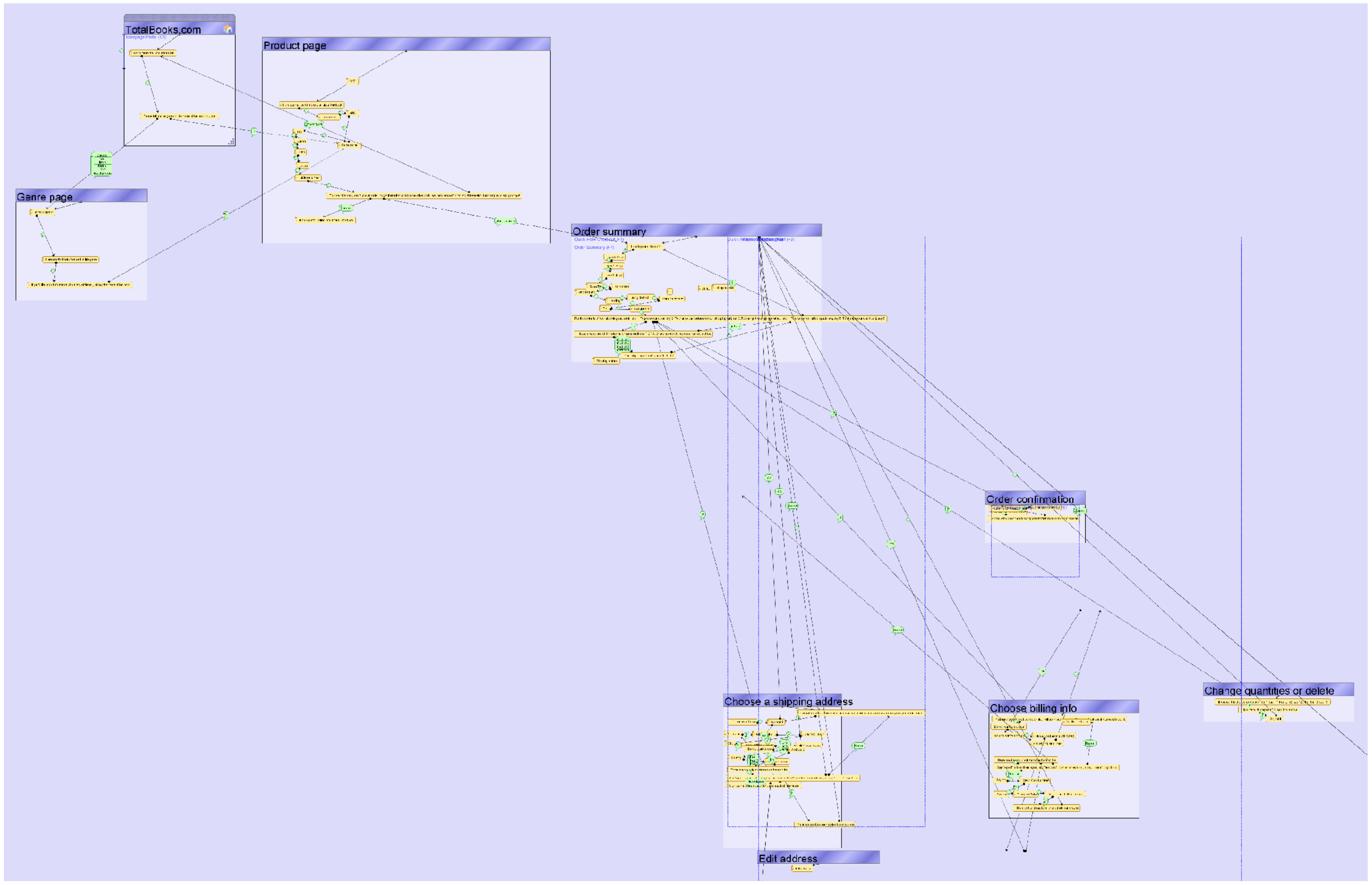


**FIGURE 6-14** Participant 15’s voice design for TotalBooks.com, made with patterns and layers.

### 6.7.1 *Quality analysis of voice designs*

To evaluate the voice designs, we used experts to judge the voice UI designs. We first modified the voice designs to fill in placeholders with actual items to give the illusion of finished designs. We then asked four people, three HCI researchers and one HCI engineer, to listen to each of the eight voice designs. The order in which each judge listened to the designs was random. For each design, they were given a specific book or music album to "buy," along with enough information to complete the purchase. They then answered five questions: the confirmation number and the total price of the purchase (to make sure they were paying attention to the interface), what they did and did not like about the voice interface, and then a rating of the design from 1 to 7, where 1 was "did not like" and 7 was "liked very much." See Appendix G for the experimental questions and raw data.

The results are mixed, as shown in Table 6.22. There is no statistically significant difference between the voice designs with and without patterns. (Order effects were not significant.) However, it is instructive to specifically look at what the judges did and did not like about the designs, and infer how patterns affected their opinions.

**TABLE 6.22** Average ratings of all eight voice designs (four non-patterns and four patterns) and the voice designs except those from Designer 16, who only used one pattern in the Patterns condition (1 = did not like, 7 = liked very much).

| | Participants | | | | | |
|---|---|---|---|---|---|---|
| | 13 | 15 | 16 | 17 | All designs | All designs except Designer 16 |
| Non-Patterns | 5.50 | 4.75 | 4.00 | 5.00 | 4.81 | 5.08 |
| Patterns | 3.00 | 5.00 | 6.00 | 4.50 | 4.63 | 4.17 |
| *p* (2-tailed *t*-test) | | | | | 0.85 | 0.38 |

Of the designs that include patterns, many of the evaluations described them as being efficient and fast to use. However, there were also many complaints about the excessive wordiness, the unnecessary repetition of items in the shopping cart, and the lack of a "goodbye." These comments stem directly from the parts of the design that came from patterns.

Since the patterns in *The Design of Sites* [188], on which the Damask patterns are based, do not contain voice-specific solutions, we created the solutions ourselves. Unfortunately, we did not have the opportunity to consult voice designers at the time they were designed. This led to voice solutions which were sometimes too similar to the desktop solutions. For example, in a desktop web site, it is fine to display the contents of a shopping cart after adding an item to it, because the user can quickly glance at the contents. But in a voice UI, adding an item to a cart should *not* result in the user listening to a possibly long list of items. Instead, there should only be a brief confirmation statement like, "OK, it's been added." Fixing the patterns would dramatically improve these designs without any more effort on the designers' part.

Interestingly, during the study of Damask itself, out of the four desktop/voice participants, only Participant 17 said that the voice user interface that she created with patterns was somewhat awkward. This may be because during Run mode, they did not actually listen to the interface, but instead read it in a dialog box. All of them knew the importance of actually listening to an interface to get it right, but thought the visual Run mode was adequate for this early phase of design.

The designs without patterns had their own pluses and minuses. In general they were praised for their efficiency, but some judges noticed that three of the four designs without patterns also did not have a "goodbye" message. One design did not include a confirmation number, and another did not say the final total cost, problems that would have been easily avoided by using the QUICK-FLOW CHECKOUT pattern.

## 6.8 Pattern Usage and Ratings

Table 6.23 contains a list of patterns that designers used, along with which designers used them. We only implemented solutions for the patterns that we thought the designers would use, so the participants were limited to using only the eleven patterns listed in the table.

**TABLE 6.23** Patterns used by the participants

| Participant # | # patterns explicitly used | # total patterns used | B8: Category Pages | C1: Homepage Portal | F1: Quick Flow Checkout | F2: Clean Product Details | F3: Shopping Cart | F4: Quick Address Selection | F5: Quick Shipping Method Selection | F6: Payment Method | F7: Order Summary | F8: Order Confirmation and Thank You | H2: Sign In/New Account |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 4 | 5 | 11 | ● | ● | ● | ● | ● | ⊙ | ⊙ | ⊙ | ⊙ | ⊙ | ⊙ |
| 5 | 6 | 6 | ○ | ● |  | ● | ● | ● | ○ | ● |  | ● | ○ |
| 6 | 4 | 10 | ● | ○ | ● | ● | ● | ⊙ | ⊙ | ⊙ | ⊙ | ⊙ | ⊙ |
| 7 | 5 | 11 | ● | ● | ● | ● | ● | ⊙ | ⊙ | ⊙ | ⊙ | ⊙ | ⊙ |
| 8 | 0 | 0 | ○ | ○ | ○ | ○ | ○ |  |  |  |  |  | ○ |
| 9 | 5 | 11 | ● | ● | ● | ● | ● | ⊙ | ⊙ | ⊙ | ⊙ | ⊙ | ⊙ |
| 10 | 7 | 8 | ●² | ○ |  | ● | ● | ● | ⊙ | ● | ● | ● | ○ |
| 11 | 4 | 10 | ●² | ○ | ● | ● | ● | ⊙ | ⊙ | ⊙ | ⊙ | ⊙ | ⊙ |
| 13 | 4 | 10 | ● | ○ | ● | ● | ● | ⊙ | ⊙ | ⊙ | ⊙ | ⊙ | ⊙ |
| 15 | 2 | 8 | ○ | ● | ● |  | ○ | ⊙ | ⊙ | ⊙ | ⊙ | ⊙ | ⊙ |
| 16 | 1 | 1 | ● | ○ |  |  |  |  |  | ○ |  |  | ○ |
| 17 | 9 | 9 | ○ | ● |  | ● | ● | ● | ● | ● | ● | ● | ● |
| Average | 4.3 | 8 |  |  |  |  |  |  |  |  |  |  |  |
| # designers explicitly used |  |  | 8 | 6 | 7 | 9 | 9 | 3 | 1 | 3 | 2 | 3 | 1 |
| # designers used |  |  | 8 | 6 | 7 | 9 | 9 | 10 | 10 | 10 | 9 | 10 | 8 |

● = used once
●² = used twice
⊙ = embedded within another pattern that was used
○ = viewed during the pattern learning phase but not used in the main task

Out of twelve participants, six used 10 or 11 patterns, four used 6 to 9 patterns, one used 1 pattern, and one used none. On average, the participants explicitly used 4.3 patterns and implicitly used 8 patterns, since QUICK-FLOW CHECKOUT (F1) and ORDER SUMMARY (F7) contain other patterns. Also, out of the eleven patterns that had

Damask solutions, all of them were used by at least six out of twelve participants, and nine of them were used by at least eight. This indicates that most designers were able to find the patterns in Damask that were relevant to their design task, even though they only had fifteen minutes to look through the pattern collection (containing 90 patterns) and the e-commerce patterns were not explicitly pointed out to the participants.

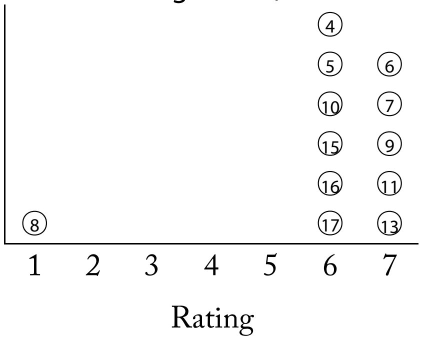
Although we did not ask most of the participants why they did not use certain patterns, we noticed that when participants did not use a pattern that we expected them to use, it was usually because they did not look through the pattern during the learning task or the main design task. Therefore, it is likely that the participants simply did not know about the pattern and its title was not descriptive enough to catch their eye; this was the reason we got when we asked participants 15, 16, and 17 about not using particular patterns during the post-test debriefing. Since the participants were given only fifteen minutes to learn as much they could about ninety patterns, this is not surprising, and the fact that designers used 8 patterns on average is a very positive sign. In the case of HOMEPAGE PORTAL, some designers did look at it but decided not to use it, because its solution did not look like an e-commerce home page.

The participants who used design patterns extensively said that patterns saved them time because they would not have to "reinvent the wheel." Participant 9 said, "They let you skip the step of creating them and just pick up stuff that's already been proven to work." Participant 10 said, "They help enforce consistency and they also save a bunch of time when designing something that doesn't really require a whole bunch of innovation. Nice that the patterns are flexible, too. I can delete portions if I don't want to use them."

As for the two participants who used one pattern or less, Participant 16 said that he did not feel familiar enough to use the patterns and did not want to take more time to look them up, besides CATEGORY PAGES, which he had already used in the layers and patterns warm-up task. Participant 8 thought that the task was too simple to need to use patterns, but said that the patterns overall had "lots of possibilities—would be good for client meetings/needs analysis." Note that not only did Participants 8 and 16 not use many patterns, they barely looked through the patterns during the main design task, if at all.

Most of the participants liked the patterns that Damask provided for the given task. The average rating for the patterns' usefulness in the given task was 6 out of 7 (only 1 out of 12 ranked usefulness less than a 6, and that participant did not use the patterns for the task; see Table 6.24).

**TABLE 6.24** Participants' ratings on the usefulness of the patterns for the TotalBooks.com design task (1 = not useful, 7 = very useful)

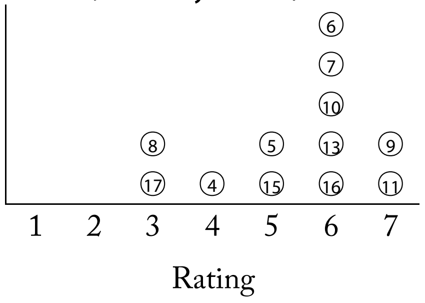


mean = 6, median = 6, std dev = 1.65

For general usefulness, the opinion was less strong but still positive overall. Seven of the twelve participants gave a rank of 6 or 7 (average = 5.33). Many of them talked about how patterns saved them time. Participant 9 said, "They let you skip the step of creating them and just pick up stuff that's already been proven to work." Participant 10 noted, "They help enforce consistency and they also save a bunch of time when designing something that doesn't really require a whole bunch of innovation. Nice that the patterns are flexible, too. I can delete portions if I don't want to use them."

The other five of the twelve participants gave the patterns' usefulness a rank of 3, 4, or 5. Two of them mentioned how the patterns' solutions seemed to assume too much. Participant 4 said, "Sometimes they assumed I was using them in a particular way." Participant 15 said, "Even with 80 [sic] patterns, sometimes the pattern you pick does not match what you want" (see Table 6.25).

**TABLE 6.25** Participants' ratings on the general usefulness of the patterns (1 = not useful, 7 = very useful)



mean = 5.33, median = 6, std dev = 1.37

Besides the patterns we expected the participants to use, there were several patterns that the participants tried to use but which did not have any Damask-based solutions that they could directly incorporate into their designs. On average, the designers tried to use 3.5 patterns that were unimplemented, 4.2 if we do not consider Participants 8 and 16 (see Table 6.26). The time spent trying to use these patterns also cost these designers time on the task. One could imagine the task times for the patterns condition would have been even lower had these patterns been implemented.

The three patterns most frequently tried were PERSONAL E-COMMERCE, FEATURED PRODUCTS, and SEARCH ACTION MODULE. Had we implemented the latter two, it would have shortened the participants' design time even more, but it is not as clear what the impact of using the PERSONAL E-COMMERCE pattern would be.

PERSONAL E-COMMERCE is a genre pattern that references all of the lower-level patterns in the E-Commerce genre. It would have included more functionality than the designers would have needed, but it also would have been so overwhelming that designers might have decided not to use it or might have had trouble deciding what to remove from the pattern instance for the purposes of the design task.

**TABLE 6.26** Patterns that participants tried to use but had no Damask solutions. The highlighted columns indicate the patterns most frequently tried.

| Participant # | # patterns explicitly used | A1: Personal E-Commerce | A7: Valuable Company Sites | B1: Multiple Ways to Navigate | B2: Browsable Content | B3: Hierarchical Organization | B4: Task-Based Organization | B5: Alphabetical Categorization | B7: Popularity-Based Organization | C2: Up-Front Value Proposition | D1: Page Templates | D2: Content Modules | G1: Featured Products | G2: Cross-Selling and Up-Selling | G4: Recommendation Community | G5: Multiple Destinations | J1: Search Action Module | J2: Straightforward Search Forms | K1: Unified Browsing Hierarchy | K2: Navigation Bar | K10: Obvious Links |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **4** | 2 | ⊘ | | | | | | | | | | | | | | | ⊘ | | | | |
| **5** | 5 | | | | | | | | | | | | ⊘ | ⊘ | ⊘ | | ⊘ | ⊘ | | | |
| **6** | 7 | ⊘ | | | | ⊘ | | ⊘ | | | | | ⊘ | | | | | | ⊘ | ⊘ | ⊘ |
| **7** | 10 | ⊘ | | ⊘ | ⊘ | ⊘ | | | | ⊘ | ⊘ | ⊘ | ⊘ | | ⊘ | | ⊘ | | | | |
| **8** | 0 | | | | | | | | | | | | | | | | | | | | |
| **9** | 0 | | | | | | | | | | | | | | | | | | | | |
| **10** | 4 | ⊘ | | | | | ⊘ | | | | | | | | | | ⊘ | ⊘ | | | |
| **11** | 8 | ⊘ | | | ⊘ | | | | | ⊘ | ⊘ | ⊘ | ⊘ | | ⊘ | ⊘ | | | | | |
| **13** | 4 | ⊘ | ⊘ | | | | | | | | | | ⊘ | ⊘ | | | | | | | |
| **15** | 1 | | | | | | | | ⊘ | | | | | | | | | | | | |
| **16** | 0 | | | | | | | | | | | | | | | | | | | | |
| **17** | 1 | ⊘ | | | | | | | | | | | | | | | | | | | |
| Total | | 7 | 1 | 1 | 2 | 2 | 1 | 1 | 1 | 2 | 2 | 2 | 5 | 2 | 3 | 1 | 4 | 2 | 1 | 1 | 1 |

While all of the designers liked the amount of detail in the patterns, they found some of the larger patterns intimidating and hard to use. For example, the layout of the solution of the QUICK-FLOW CHECKOUT pattern made it hard for designers to figure out how to link the instance with the rest of their design. Participant 6 said, "A somewhat cleaner layout of pages would help, to make it easier to see what you've just pasted in."

Also, when we first created the solution, some of the page titles were not descriptive enough, which also resulted in a couple of designers not using the pattern as intended. We tried to mitigate this by including a Preview button right above the solutions, which allows designers to interact with the solutions in Run mode, but it was rarely used. Adding descriptive features, such as annotations to describe parts of the design and making important arrows stand out more, would help designers understand the patterns faster. Also, making parts of patterns collapsible would allow designers to learn patterns more gradually.

Adding to the intimidation factor was that after designers instantiate a pattern, they cannot move all of the pages in the pattern instance at the same time. This made them a bit more hesitant to instantiate a large pattern, and it was a chore to easily move an instance of a large pattern such as QUICK-FLOW CHECKOUT to another part of the canvas. This could be alleviated by letting designers drag the box surrounding the pattern instance, thus moving the entire instance.

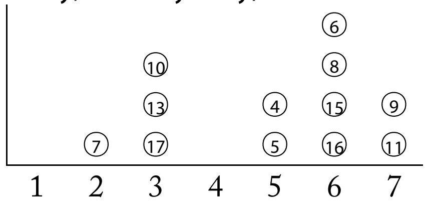
Some of the participants voiced concerns about the broadness of the individual patterns. For example, the participants who did not use the HOMEPAGE PORTAL pattern said that the introductory photo or the solution did not look like a typical e-commerce site. One way to address this is to have more than one solution per pattern, each tailored to particular genre. For example, a solution for a news site homepage would be different than one for an e-commerce site homepage.

Another interesting aspect is how the participants interacted with the Pattern Explorer during the design task. Most of them concluded that a design pattern would be useful only if it had a solution, so instead of reading a pattern carefully, they quickly scrolled through the pattern, and if they didn't find a solution, they moved on to another pattern. This behavior may have stemmed from the somewhat artificial scenario, but we believe that many designers will not bother to read the entire pattern in real life, just like most users do not read documentation. This may require that the patterns be written in a more bulleted outline format, using hyperlink expansion for more detailed discussions.

## 6.9 Layers Usage and Ratings

All of the participants understood layers well enough to perform the task. Six out of eight desktop/smartphone designers said that the concept of layers was easy or somewhat easy to understand (rated at least 5 out of 7), and two said they were hard or somewhat hard to understand (rated 2 and 3). Two out of four voice designers found the concept easy to understand (rated 6), and the other two found it somewhat hard (rated 3). The overall average was 4.92 on a scale of 1 to 7 (7 being best), but it is clearly bimodal (standard deviation = 1.73). See Table 6.27.

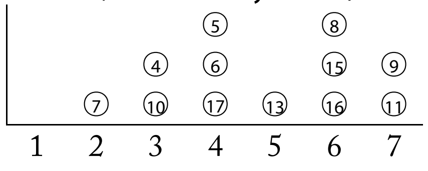
**TABLE 6.27** Participants' ratings on how easy to understand layers are (1 = not easy, 7 = very easy).



mean = 4.92, median = 5.5, std dev = 1.73

The participants' preferences for layers were also mixed. Three out of eight desktop/smartphone designers and three out of four voice designers said they liked layers (at least 5 out of 7); the overall average was 4.75 (see Table 6.28).

**TABLE 6.28** Participants' ratings on how much they like layers (1 = do not like, 7 = like very much).



mean = 4.75, median = 4.5, std dev = 1.66

The most frequent mistake that participants made was forgetting which layer they were currently in, and therefore making changes that did not propagate the way they expected. One participant suggested that since most of the participants start with the desktop design first and then modify the other generated designs, the default layer in the smartphone and voice designs should be This Device instead of All Devices.

Another usability problem that most of the participants ran into was moving objects between layers. Originally, the Move Object to This Device button actually took an object that was in all devices, and made a separate object for *each* device. None of the seven participants (Participants 4–10) who used Damask with this behavior understood that. They assumed that if they were in the desktop view and used the Move Object to This Device button on an object, then the object would be removed from the smartphone and voice views. We later changed Damask to do this, and the remaining five participants had no problems understanding the new behavior.

However, using that button still was not always natural. For example, suppose there was a button that appeared in all devices. If designers wanted to remove the button from the smartphone user interface only, their natural inclination was to try to use the eraser and erase the object in smartphone view, rather than going back to the desktop view and moving the object from All Devices to This Device.

We discuss possible improvements to layers in Section 8.3.

## 6.10 Usability of Damask

Overall, we found that designers had little problem understanding the basic concepts of Damask's user interface. Although they all ran into Damask's quirks, by the second session, they had all gotten much more comfortable using the interface. Many of the participants verbally commented how they liked the informal, fluid interaction style of Damask. This was a new experience for most of the participants—four of the participants had only used DENIM a little bit, and the other eight had never used DENIM.

For the two main design tasks, every designer started designing the desktop web UI first, including the voice designers who did not consider themselves web designers. They said that they considered that the "main" interface.

After each session, the designers were asked to write down what they liked about Damask, without being prompted for anything specific. All of the answers can be found in Section C.6.3, Question 3 and Section C.7.7, Question 3; also see Section C.8.1 for a detailed summary. We will give the highlights below.

- Seven out of twelve designers mentioned design patterns. Participant 9 said she liked "being able to grab ready-made plans, because it helps avoid reinventing the wheel." Participant 17 said, "For building the web prototype it was very useful to have the patterns, so I didn't have to do some things from scratch (like login, shopping cart, etc.)"
- Seven designers mentioned Run mode. Participant 6 said that "the 'runtime' mode for simulating user experience" is "easily" one of his "favorite aspects."
- Five designers mentioned either layers specifically or being able to create a user interface for more than one device simultaneously. Participant 11 said she particularly liked, "the ability to enter device-specific info in the form of layers." Participant 7 said that the "ability to reuse elements from desktop view to smartphone view saved me time."
- Five designers mentioned either the sketching interface or the pen user interface. Participant 4 said he liked "sketching non-standard UI widgets (tables, copy

blocks) without spending time drawing lines and getting the right number of characters." Participant 13 said, "I also like the idea of how 'sketchy' the windows seem—it really allows for informal brainstorming to be formalized a bit, and easily shared."

- Four designers mentioned the templates. Participant 6 said he liked "The ability to quickly templatize site elements."
- Four designers mentioned how they were able to create a prototype quickly. Participant 10 said, "I felt like I could pretty quickly create both a flow chart and a usable prototype. I like that a lot."
- Two designers mentioned the zooming interface. Participant 11 said she liked "the ability to quickly generate a top-down view of the site and then zoom in on the details of each page."
- Two designers mentioned being able to link sketches together with arrows. Participant 7 said, "I liked the showing the flow through linking on Damask."

The designers were asked what they did not like about Damask and what was missing. All of the answers can be found in Section C.6.3, Questions 4 and 5, and Section C.7.7, Questions 4 and 7; see Section C.8.2 for a detailed summary. The range of topics was much broader than in the previous question, but the top three concerns, each mentioned by four designers, were:

- Layers. Participant 6 said, "I'm still on the fence about the layers." Participant 10 said, "The concept of layers was a little hard to grasp. With more experience I could pick it up, but I had to keep thinking about it."
- The awkwardness of the sketching interface as implemented in Damask. Participant 7 said, "Writing with a pen didn't translate well."

- The difficulty of panning and zooming around the canvas. Participant 6 said, "The most frustrating part was trying to move about the workspace. Many tasks involved this laborious traversal of space, from copying and pasting elements between pages to linking distant pages to objects on the current page." Participant 10 said, "Certain features could probably be moved more in the direction of how other applications do them. An example would be zooming (i.e., Photoshop)."

Note that layers were mentioned both positively and negatively.

There were also some usability issues that the participants experienced. Most participants were confused that one tool, the pencil, had three uses: creating pages, drawing inside pages, and linking pages together, especially because every other tool performed only one function. This design decision originated with Damask's predecessor, DENIM, where the pencil was the only tool available to create pages and objects. There are several possible ways to fix this. Some involve making the user interface more pen-centric. One way is to eliminate the other control tools, keeping only the pencil, and rely on recognition for inserting controls, like SILK [99]. Another is to use a "stamping" metaphor for the control tools, similar to the augmented DENIM that we described in Chapter 4, which would reinforce the overall sketching metaphor. A completely different route is to make the user interface *less* pen-centric and more conventional, by adding another tool for creating a page. We found that Visio was the most commonly used tool among our participants (11 out of 12), so making the tool more similar to Visio may ease the learning curve.

Screen real estate was another usability issue. Since the Pattern Browser was in its own large window, it was hard to have it on-screen at the same time as the main Damask window. One designer specifically suggested that Damask should include a mini-browser, with thumbnails of the pattern solutions, which would be a sidebar just next to the main window.

The template pane had similar problems: its zoom level was tied to the main window, and its width was fixed depending on the zoom level. If the designers zoomed in to modify a template, the template became so wide that most of the rest of the canvas was not visible. This made it hard to do actions such as finding a page to link to the template. This can be fixed by making the template pane's width and zoom level independent of the main canvas. It could possibly be put into another window, although this would make linking template elements to the main canvas difficult.

Designers had problems remembering how to link template elements to other pages. Instead of drawing an arrow from a link in the template pane to the link's destination, they instead drew an arrow from a page that used the template in the main window to the link's destination. This unwittingly created an organizational arrow instead of a navigational arrow, which caused designers to be confused when their link did not work. They did not notice that the organizational arrow was a different color. Damask should allow designers to link template elements from within normal pages in addition to the template pane, but indicate that the link is global, perhaps by animating the source of the link to the template pane.

Designers experienced some performance problems with Damask. It had trouble keeping up with participants' writing many letters by hand. Using a framework that is optimized for pen input, such as the Digital Ink APIs in Windows XP Tablet PC Edition, would greatly alleviate that. Also, the system slowed down and became less responsive if the participants inserted many pattern instances. Since the code has not been optimized, there are plenty of opportunities for improving the performance of Damask.

## 6.11 Summary

Our experiment with Damask shows that designers are enthusiastic about the concept of patterns and can use them effectively. Using patterns during the early stages of design saves time and results in design concepts that include more functionality with a more standard interaction flow. However, there are issues with learning and using large patterns with lots of functionality. We have also shown that layers are a powerful metaphor for distinguishing UI elements for all devices versus those for only one device. Even though the current implementation of layers is complex and somewhat confusing, eight of the twelve designers still prefer this metaphor over an alternate metaphor that we presented (which is described in Section 8.3.3), implying that the concept is on the right track, but the details and implementation need to be improved.

# 7 Related Work

This chapter contrasts Damask's approach to other related work, including model-based UI tools, tool support for patterns, combinations of model-based and pattern-based approaches, tools to transform UIs from one device or modality to another, and user interface design tools in general.

## 7.1 Model-Based UI Tools

Szekely [178] identifies five approaches that model-based UI tools have taken: automatic interface design, specification-based model-based interface development environments, help generation, tools to help designers create models, and design critics and advisors. We address all of these except help generation, which is not part of the target domain of Damask.

### *7.1.1 Automatic interface design tools*

Automatic interface design tools [16, 26, 49, 66, 89, 102, 157, 199] strive to automatically create the user interface of an application, given a task or domain model of the application.

As Szekely describes [178], an automatic design tool typically takes the following steps to generate a user interface:

1 Determine the presentation units. The tool figures out the windows that will be used and the contents of those windows.

2 Determine the navigation between presentation units. The tool constructs a graph of presentation units that defines which presentation units can be reached from other units.

3 For each presentation unit, determine the abstract interaction objects, which define the behavior for each element in a presentation unit in an abstract manner, for example, "select one from a list."

4 Map abstract interaction objects into concrete interaction objects, which are actual widgets available in a toolkit.

5 Determine the window layout, in other words, where the widgets are placed in the window.

The first three steps build the abstract UI specification, and the last two build the concrete interface.

As Szekely discusses, each of these steps is difficult to automate, especially steps 1 and 3, which require a deep understanding of the user's tasks. For example, it is hard for a tool to tell whether a set of data is better displayed as a table or as a graphical display like a map. Consequently, designers have not accepted the tools since it is harder to create a model and guess what the tool will generate than it is to design it directly themselves. Some tools, such as Tadeus [164], explicitly involve the designer in each step instead of trying to do each step automatically, as a way to address this problem.

In Damask, we sidestep the automation problems since Damask does not require explicit definition of a domain or task model. Instead, a designer using Damask designs a concrete interface for one device embedded with design patterns. Damask uses information in the existing concrete design and the design patterns it uses to generate an appropriate UI for the other devices.

### 7.1.2 Specification-based model-based interface development environments

Unlike automatic interface design tools, specification-based model-based interface development environments (MB-IDEs) do not try to automatically generate a user interface from task or domain models. Instead, designers directly create and interact with task, domain, or presentation models, which the MB-IDE then uses to generate a final UI. Letting designers directly interact with models enables them to more easily specify a design, change it, retarget it, and so on.

Early specification-based MB-IDEs [85, 165, 179, 180, 197] use modeling languages that tend to look like traditional programming languages, which are inappropriate for designers, who often have little programming background. Also, the languages are at a level of abstraction that we feel is inappropriate at the early stages of design and prototyping.

A newer generation of specification-based MB-IDEs [5, 11, 36, 37, 45, 75, 84, 92, 110, 132, 134, 147, 159, 194, 201] use XML-based languages for describing user interface models. (Souchon and Vanderdonckt [173] have a good overview of many of these languages.) Many of these languages were designed with targeting multiple types of devices in mind. Designers are more likely to be comfortable with the syntax of these languages, since they are superficially similar to HTML. However, they are still more similar to programming than to existing early-stage design practices, and they are still at a level of abstraction not appropriate for early-stage design and prototyping, in the same way that most web designers do not create their designs initially in HTML.

Damask takes a different approach. It leverages the existing work practices of designers, who sketch rather than program, to generate cross-device UIs. Damask builds the interaction model of a design from both the designer's sketch and the design patterns it uses.

### 7.1.3 Modeling tools

Some MB-IDEs include a modeling tool to help a UI designer create a model-based UI without creating the model directly. FUSE [107] and Adept [199] have simple form interfaces to edit models, but they have not been extensively evaluated. We also believe that a form-based interface is not a good match for UI designers' existing work practices, which involve freeform sketching of UI designs. Inference Bear [53] and Grizzly Bear [52] use a programming-by-demonstration interface builder as a front-end to creating a model; however, the model itself is exposed to the designer through a special-purpose modeling language. More recent modeling tools, such as TERESA [132], PLANES [117], Windows Transition Notation UIDA [190], and XIML Task Simulator and DialogGraphEditor [50], allow the creation of models using a graphical language such as ConcurTaskTrees [153] or Windows Transition Notation [190]. Several projects [23, 117, 152] have proposed using the Unified Modeling Language (UML) for UI modeling.

In contrast, Damask does not expose the abstract model directly to designers, since they do not usually think about their interfaces in terms of abstract models.

UI Pilot [158] takes a different approach to modeling. Designers using UI Pilot create a user-task model in a simple outline mode, including the types of data and users in the application. Then they construct a "wireframe" for each screen or web page of the system, by dragging the appropriate tasks, user types, and data into a blank wireframe. They can also add links to related wireframes. Designers can also construct wireframes without directly constructing a model, by creating data on the fly within a wireframe. In this way it is similar to Damask, which also does not require explicitly building a model. UI Pilot and Damask are complementary—currently, UI Pilot does not handle actual screen layout of user interface controls, while in Damask, the dominant design paradigm is sketching out UI controls within a page.

### 7.1.4 Design critics and advisors

Design critics and advisors use information provided by models to give an analysis of the user interface design. There are three basic types [178]:

- Property verifiers [49, 151, 154] verify that a design satisfies certain properties, such as whether all parts of the design are reachable.
- End-user simulators [88] simulate users using the application and predict task times learning times, and errors. Some tools, such as critique [73] and the CogTool [83], create predictive models of a task based on a person demonstrating the task.
- Summative evaluators [35, 166] analyze a design and give it a score based on a set of criteria or a theory of, say, layout quality.

Generally, these tools have been hampered by the fact that it is currently difficult to encode high-level design guidelines into a precise set of rules that a tool can check. Damask takes a different approach, by collecting various types of actual usage data during a "Run mode" and then using other tools, such as WebQuilt [70] and SUEDE [91], to display that data in an "Analysis mode" for designers to evaluate. In this way, designers can draw upon their design experience when analyzing their designs. In addition, the included patterns encode design guidelines and best practices.

### 7.1.5 *Prototyping vs. finished interfaces*

The philosophy of most model-based UI research is that the model-based tools would be the primary way to create the finished user interface, although many tools expect the user interface to be modified somewhat by a designer. In contrast, Damask is targeted towards prototyping. We do not expect the designer to use Damask to create the final user interface, nor do we expect Damask's generated user interfaces to be used without modification. Since we are targeting prototyping, the generated user interface does not need to be ideal, since in the early stages of design, the designer is concerned more with the user's interaction flow rather than the details of the interface [195, 200].

## 7.2 UI Tools and Languages for Multiple Devices

Many model-based tools and languages specifically address the issue of creating user interfaces targeted at multiple devices [22, 23, 33, 36, 37, 39-41, 62, 67, 75, 92, 109, 110, 117, 119, 120, 130, 132, 134, 159, 194]. They all take the same basic approach: the designer or developer describes a user interface using one or more abstract models, either directly in the textual modeling language or by using a GUI tool. These models are then rendered into a concrete user interface, either during development or at run-time using a supporting renderer. Sometimes, the models are annotated to give hints to the renderer about how to display the user interface for a particular device.

Using these tools, it is generally difficult to create an application where parts of the presentation model are different from one device to the next, even if the same overall task or concept is in common. For example, to specify that an interface should use a graphical map of locations on a PC but a textual list of addresses on a mobile phone display, the designer would essentially need to create two interfaces from scratch, and somehow link the two interfaces together if he or she cared about keeping the abstract models consistent. There are no tools that can recognize a map and automatically generate a list for use on another device. Damask uses patterns for this purpose: a pattern represents a general concept where the abstract models for each device may be quite different. So for the previous example, Damask would have a pattern containing a map for the PC and a list for the mobile phone already linked up, which designers could then use directly in their designs.

GADGET [48] and SUPPLE [55] take a different approach: they view user interface generation as an optimization problem. GADGET is actually an optimization toolkit for interface layout. One can use GADGET to take an existing collection of user interface controls and generate a layout for those controls. This makes GADGET potentially useful to Damask for creating, for example, a smartphone UI from a desktop UI, since many of the controls would be the same, but makes it less useful for generating voice UIs from desktop UIs, where the controls may be very different. SUPPLE takes an abstract model (called a *functional specification*), a device model, and a trace of UI events created by a user to generate an interface. Damask could potentially use SUPPLE's algorithms to generate device-specific UIs, provided there is a robust way to create a functional specification from a concrete user interface that a designer creates for a specific device.

There have been several projects that aim to create a platform for creating universal remote controls [69, 80, 142, 143, 155, 201]. These projects envision appliances that export high-level descriptions of a remote control user interface to another device, such as a PDA or a Braille reader, which then renders that description into a concrete UI. The UI would take the user's input to the remote control UI and send it back to the appliance for processing. There are two important distinctions between the problems these projects are solving and Damask's problem area. The target domain of universal remote controls is narrower (remote controls for appliances vs. web interaction), but the UIs that are rendered from the abstract remote control description must be appealing and useful immediately, without additional tweaking. Damask, on the other hand, is targeting a broader set of UIs (general web-style interaction on PCs, mobile phones, and voice applications) but the interfaces that are generated will most likely be modified by the UI designers before being released.

Calvary, Coutaz, and Thevenin [31] discuss a process framework for developing plastic interfaces, which can adapt to different devices. In addition to the typical model-based approach, in which a designer creates a series of models from top-level abstract models to a concrete interface, the framework also covers translations between platforms, which may happen at any model abstraction level. This framework provides a useful way of thinking about how to develop cross-device user interfaces, although with Damask, top-level abstract models are not directly exposed, so the framework is not directly applicable.

Wiecha et al [198] discusses the possibility of factoring web services so that issues such as device, navigation style, localization, and personal preferences are separated into transforms that are then applied, one by one, to an abstract application definition. Each transform in this chain of transforms could then be implemented as proxies or intermediaries between content providers and consumers. This paper discusses run-time issues, which are actually orthogonal to the early-stage design and prototyping issues that Damask addresses. Once a cross-device UI is designed with Damask, Wiecha et al's chain of transforms could be used to implement such a user interface.

## 7.3 Tool Support for Patterns

There has been much discussion about using design patterns in HCI [27, 182, 183, 188, 189], but few HCI tools have been created that support patterns. Paternò [153] describes extending a task and architecture model editor to support patterns that are made up of model fragments themselves. Paternò focuses on abstract task and architecture patterns. A task pattern describes what steps a user performs to execute a particular task, such as searching, independent from a particular user interface. An architecture pattern describes how the program implements a task, such as how a program accesses a database to perform a search. On the other hand, since designers create concrete UI designs using Damask, the patterns in Damask also consist of concrete UI fragments.

In computer science, patterns have made the most impact in software engineering. Software engineers use patterns to talk about how classes in object-oriented programs are organized and how they communicate with each other. Patterns were first used in this way by Beck and Cunningham [20], and this approach was popularized in a book by Gamma, Helm, Johnson, and Vlissides [56], commonly known as the "Gang of Four." Consequently, software tools that support patterns have mostly targeted object-oriented software development.

Budinsky et al [29] describes a system that generates design pattern code automatically, using pattern templates and application-specific information provided by the programmer. The tool also provides an online version of [56] to allow programmers to quickly browse and access information about patterns.

Florijn et al [47] describes a tool that allows programmers to view their programs in three different views: pattern, design (i.e., class diagrams), and code. This tool allows programmers to instantiate patterns from a repository, to bind existing code to a pattern, and to check whether their code still conforms to a pattern's constraints.

Pagel and Winter [150] describe a pattern metamodel that can describe all object-oriented design patterns known up to when the paper was written, how to instantiate an abstract pattern from a pattern repository into a concrete pattern used in a design, and a tool that supports the use of patterns in software design.

FACE [118] is a system in which a developer builds an application by directly customizing abstract design patterns, which are represented with a representation of classes and their relationships similar to the Object-Modeling Technique (OMT) [160], a modeling language that is a predecessor of UML.

Rational Software Architect [78], Software Modeler [79], and XDE [77]; ModelMaker [127]; OmniBuilder [148]; and objectiF [126] are CASE tools that allow developers to use design patterns in developing their applications. These tools typically let developers browse patterns, take existing designs and instantiate patterns in them, and check the design to make sure it still fits a pattern's specification. In objectiF's pattern catalog, each pattern is structured using the template structure found in [56]. Rational Software Architect can also detect patterns and anti-patterns in code.

Pattern-Lint [167] lets programmers determine whether a section of code conforms to a pattern, through static analysis of the code, and a visualization of the classes and their relationships during runtime.

These tools only address software engineering issues, not user interface issues, but there are some aspects of these tools which address issues that any pattern-based tool needs to support, such as browsing and searching for patterns, and customizing patterns for a particular application. However, customizing patterns with these tools usually involves a form-based interface, which would fit awkwardly with Damask's sketch-based interface. Also, if a developer wants to use a pattern, some of these tools force the developer to change his solution to make it fit the pattern. Damask will not do this. One of the most important aspects of patterns is their flexibility: using the Alexandrian definition of patterns, a developer should be able to use a pattern many times but never the same way twice. Designers using Damask are free to greatly modify how a pattern is used in their particular design.

## 7.4 Combining Models and Patterns

Other groups have proposed combining patterns and model-based approaches. Hussey and Carrington [74] discuss designing user interfaces by starting out with an abstract UI specification, and then methodically applying transformation patterns to it to create a concrete UI specification. In contrast, designers using Damask interact with concrete UI specifications that contain UI design patterns. Paternò [153] and Sinnig et al [169] describe extending a model editor to support patterns that are made up of model fragments themselves. Trætteberg [184] discusses using fragments of models to help define design patterns, which in turn could help us understand UI models better. Damask also uses model fragments to represent patterns internally, although unlike Trætteberg's proposal, Damask does not expose the models directly to the designer.

Javahery et al [82] describe a manual process of redesigning an existing UI for another device using design patterns. The designer identifies the patterns in the UI and mechanically transforms each pattern into the appropriate form for the target device. However, they have not yet produced a tool that would allow the designer to automatically extract patterns and replace them with device-appropriate examples, as is done in Damask.

## 7.5 User Interface Transformation Tools

There has been much work on automatically transforming an existing user interface meant for one device or modality to another. Many of these projects have focused on *transcoding* finished desktop web interfaces to PDA interfaces at run-time [18, 19, 25, 30, 51, 59, 97, 108, 111]. However, automatically shrinking interfaces from large desktop displays to such small PDA displays often results in awkward interaction. Others have worked on converting GUIs to audio interfaces mostly to benefit the blind and visually impaired. With many of these systems [6, 54, 138] designers cannot modify the results of the interface transformation process. Since Damask is a prototyping tool, not a tool to adapt finished UIs, designers are free to modify the generated user interface design. Other systems [12, 72, 146] require external data to assist in translation and are not meant to be used by user interface designers.

Ultraman [171] provides a way for designers to control the transformations, but it assumes they are comfortable with the concept of trees, grammars, and writing code in Java. Damask is targeting a different audience for a different part of the design cycle: designers who have little or no experience programming, and early-stage design, before any interface is completely specified and ready to run. One of Damask's key aspects is its informal interface, which allows designers to defer unimportant design details until later in the process and focus on their tasks without having to worry about precision. Many research systems have taken this direction in recent years, either by not processing the ink [187] or by processing the ink internally while displaying the unprocessed ink [38, 61, 98, 131, 162, 163].

## 7.6 User Interface Design Tools

There are many tools, both from the research and commercial communities, which are related to the design of user interfaces.

### 7.6.1 *Research web and GUI design tools*

Damask is closely related to SILK [99], a sketch-based user interface prototyping tool. Using SILK, individual screens can be drawn, with certain sketches recognized as interface widgets. These screens can be linked to form storyboards [100], which can be tested in a run mode.

Damask is also closely related to DENIM [105, 141], SILK's successor. DENIM takes many of the ideas in SILK and extends them to the domain of web site design. However, DENIM de-emphasizes recognition of sketched widgets, focusing instead on the creation of whole web sites. Furthermore, instead of the separate screen and storyboard views in SILK, all of the views in DENIM are integrated through zooming. Also, SILK attempts to recognize the user's sketches and display its interpretation as soon as possible. DENIM intentionally avoids doing much recognition in order to support more free-form sketching. Damask's storyboarding behavior is very similar to that of DENIM. The main difference is that Damask supports multiple devices, while DENIM concentrates on desktop web design.

Damask's use of storyboarding for behaviors is based on showing what the user interface looks like before and after a user's action and is similar to the techniques used in DENIM and SILK. Other design systems that use storyboarding include DEMAIS

[14, 15], Anecdote [63] and PatchWork [187]. Similarly, Chimera [94-96] and Pursuit [128, 129], tools for graphic editing and end-user shell programming respectively, are based on a before-and-after comic strip metaphor. These systems infer what action causes a transition from examples, while in Damask, DENIM and SILK, the user interface designer states explicitly what user action causes a transition.

WebStyler [65] is a simple sketch-based tool for prototyping individual web pages. However, Damask addresses many more aspects of web site design, including designing the site structure and being able to interact with the sketches, as well as the use of design patterns and support for multiple devices.

### 7.6.2 *Commercial web and GUI design tools*

There is a lack of early-stage, commercial prototyping tools for web interfaces or desktop GUIs. Newman and Landay's study of web designers [140] shows that designers use other tools to fill this gap. Macromedia Director [112] and Flash [114] are often used to assemble storyboards, while Visio [122] is used for modeling the high-level information architecture of a web site. However, Director and Flash are multimedia authoring tools, and Visio is a general purpose diagramming tool. This makes using them for such high-level web site design awkward at best, since they were not designed for those tasks.

Axure RP [13] and iRise Studio [81] are prototyping tools for web sites that are aimed at designers and business analysts. They allow users to draw wireframes of web sites and create functioning prototypes without programming. Unlike Damask, they do not have a sketch-based interface, they do not include patterns for encapsulating reusable solutions, and they are focused on desktop web design, not cross-device UI design.

Currently, the most popular tools for creating web sites include Microsoft FrontPage [121] and Macromedia Dreamweaver [113]. These tools focus on designing page layout rather than the site architecture. Admittedly, each of them has a "site structure view" of a web site. However, this view often constrains any edits so that the tree structure remains intact. Furthermore, you cannot edit the site structure and the page layout in the same view. Most importantly, these tools focus on producing high-fidelity representations, which is inappropriate in the early stages of design, and they focus only on desktop web design. These are all important issues that we addressed in Damask.

Popular and influential tools for prototyping desktop GUIs include Microsoft Visual Basic [123] and Visual Studio [124], Apple HyperCard [9], Macromedia Director [112], and the NeXT Interface Builder (now the Mac OS X Interface Builder [10]). All of these tools allow designers to visually layout their interfaces, instead of writing code. However, these tools also focus on producing high-fidelity representations, and if designers want to test the interaction of their designs, they still have to write code to define all but the simplest behaviors. Because many designers do not have strong technical backgrounds, we have designed the visual language of Damask to be accessible to such designers, while still allowing them to specify enough interaction to evaluate their designs.

Virtual toolkits [135] are systems that allow one to create a GUI that runs on multiple desktop platforms. Some toolkits, such as XVT [156], AppWare [144], wxWidgets [170], Java AWT [57, 174], and SWT [42], use the widgets of the underlying platform, while others, such as Qt [185], Tk [149], Galaxy [7], Amulet [137], GTK+ [116], Java Swing [58, 175], and XUL [133], implement the widgets themselves. Unlike Damask, these frameworks only try to bridge differences within the desktop world and do not support the creation of mobile phone or voice UIs, and they target developers, not designers. There are visual interface builders for these toolkits that allow designers to lay out a UI design (e.g., NetBeans [139] and the Eclipse Visual Builder [43]), but they have the same drawbacks as the tools mentioned in the previous paragraph.

### 7.6.3 Commercial mobile UI design tools

Current tools for designing user interfaces for PDAs and mobile phones include IBM WebSphere Studio Device Developer [76], Microsoft Visual Studio with the .NET Compact Framework [125], and AppForge Crossfire [8]. These tools allow developers to target many mobile platforms at the same time from one code base. However, these tools are geared mostly to developers, not user interface designers, and assume substantial programming expertise. They also do not handle creating desktop or voice applications from the same design.

### 7.6.4 Voice UI design tools

Damask's voice view is a simplified version of SUEDE [91, 168], a voice UI prototyping tool. SUEDE lets designers create a lo-fi prototype of a voice user interface, collect usability data for the design using a Wizard-of-Oz (WOz) [86] mode, and then analyze the data within the context of the original design. Damask currently does not have many of SUEDE's features, including WOz and usability data collection, but a full version of Damask should include them.

Other voice design tools include CSLU Rapid Application Developer [176, 177], Unisys' Natural Language Speech Assistant [186] and Nuance's V-Builder [145]. These tools are focused more on creating and implementing finished voice user interfaces, allowing fine-grained control over voice recognition grammars, for example. While they are powerful, they are not suitable for lo-fi prototyping for the same reason Dreamweaver is not suitable for lo-fi web prototyping: the designer is more likely to start fiddling with details instead of concentrating on the overall design. However, one can imagine a designer exporting a Damask design into a format that could then be imported into one of the above tools, so that the designer can add details to create the finished design.

## 7.7 Summary

Most existing projects in the area of cross-device user interface design either require designers to specify a user interface at an abstract level, which does not match the designers' skills or current work practices, or they retarget an existing interface to other devices, which often results in awkward interaction. These tools are also focused on creating finished user interfaces. In contrast, Damask uses design patterns to help designers bridge the gap between devices. It also concentrates on early stage design and prototyping, employing a sketch-based interface so that designers can focus on important structural and navigational issues, instead of details that are better left for later stages.

# 8 Future Work

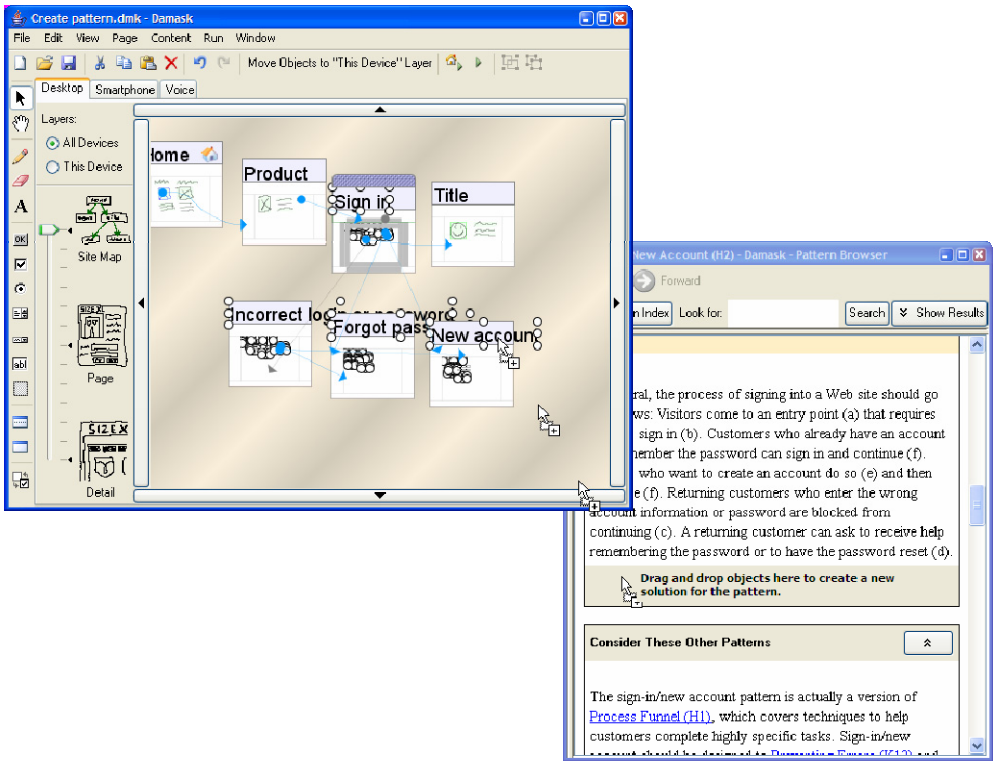
While we have established the usefulness of patterns in early-stage design of cross-device user interfaces, there are still many open research issues in this area. Here we discuss four areas for future work: patterns, annotations, layers, and other cross-device design issues in general.

## 8.1 Patterns

While we have focused mostly on using patterns in the design process, there are other aspects of design patterns that are as important.

### *8.1.1 Creating and sharing patterns*

An important aspect of supporting design patterns is allowing designers to create their own patterns and share them with a community of designers. To create a pattern, we imagine that inside the Pattern Browser, there would be a New Pattern button that would create an empty pattern. Fields such as name, background, and context would be filled in simply by typing and importing images and text. To add the Damask solution part, the designer would take parts of an existing design and drag it into the Solution section of the pattern browser. In other words, it basically would be instantiating a pattern in reverse (see Figure 8-1).



ral, the process of signing into a Web site should go
ws: Visitors come to an entry point (a) that requires
sign in (b). Customers who already have an account
nember the password can sign in and continue (f).
who want to create an account do so (e) and then
e (f). Returning customers who enter the wrong
account information or password are blocked from
continuing (c). A returning customer can ask to receive help
remembering the password or to have the password reset (d).

**Drag and drop objects here to create a new solution for the pattern.**

**Consider These Other Patterns**

The sign-in/new account pattern is actually a version of Process Funnel (H1), which covers techniques to help customers complete highly specific tasks. Sign-in/new

**FIGURE 8-1** Mockup of creating a solution for a new pattern.

For sharing patterns, a Damask patterns community web site could be set up for designers to submit patterns, and then a shepherding process [64] would give the pattern author a chance to refine the pattern before having it published. Or a pattern could simply be published, and other people could rate it and comment on it, similar to the process used at the Python Cookbook web site [115]. The pattern could be published to a web site internal to a group or company, or to a public web site.

### 8.1.2 *Showing previous uses of patterns*

It would also be useful to show how a pattern has been used in previous projects. We could do this by adding an "Example" section to a description of a pattern in the Pattern Browser that would show an instance of the pattern within existing designs, and would also download examples from the patterns community web site. These examples could then act as alternate solutions, allowing designers to "instantiate" the example into their current design (see Figure 8-2).

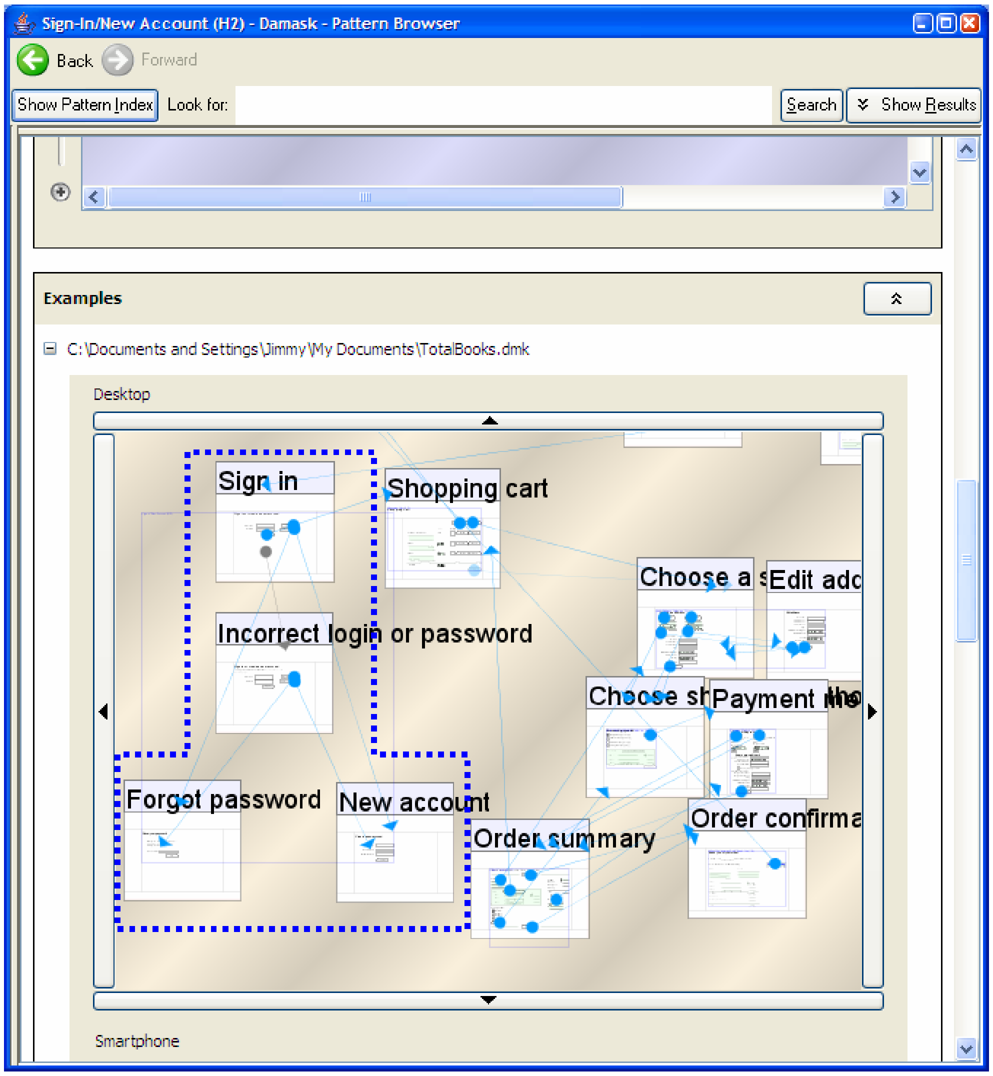
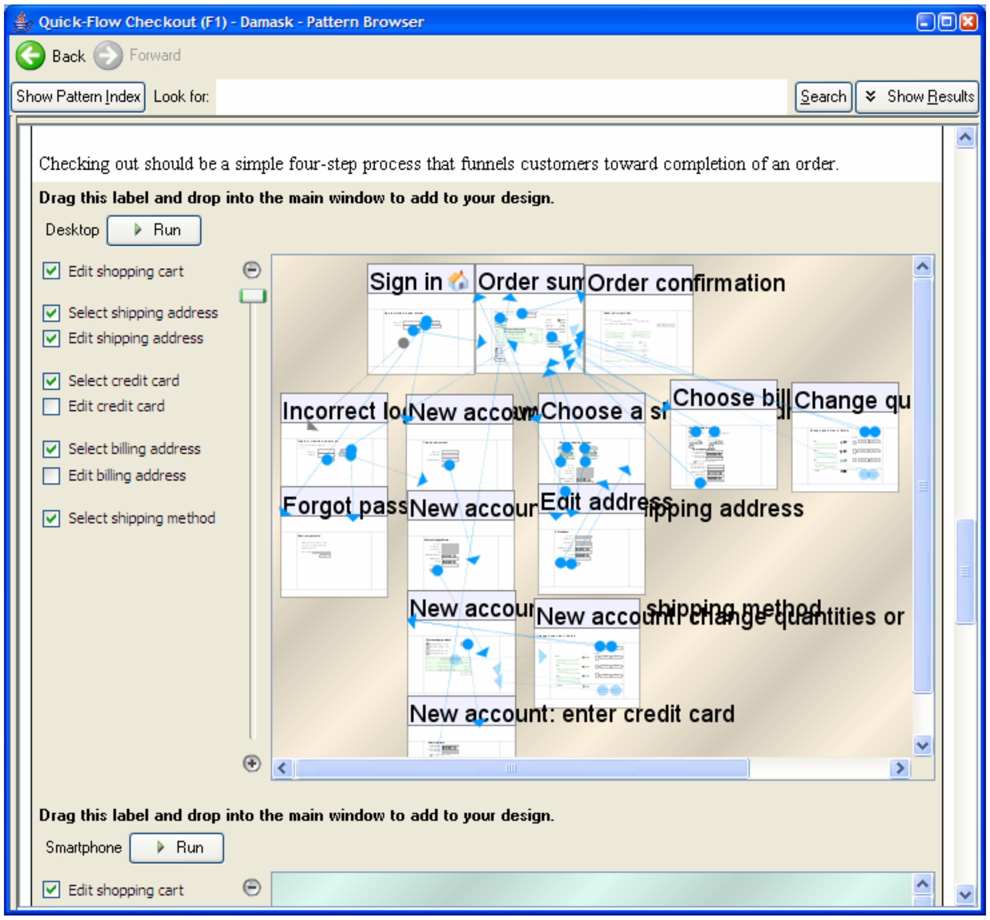


**FIGURE 8-2** Mockup of an Examples section in the Pattern Browser.

### *8.1.3 Handling large patterns*

Understanding and interacting with large patterns is an issue that needs to be addressed. Designers had a hard time understanding the scope of a large pattern such as QUICK-FLOW CHECKOUT. Typically, a pattern will explain the various tradeoffs in using it in the Forces section, which Damask did not include. However, we found that the designers in our study did not read any of the included text anyway. Also, designers may not need all of the functionality of a pattern but may have a hard time removing parts of it. If we expand Damask to include very large patterns such as PERSONAL E-COMMERCE, the issue will only become worse.

The solution section of such a large pattern could be augmented by check boxes that would turn on or off various features. For example, a full-fledged QUICK-FLOW CHECKOUT pattern would include check boxes for adding or editing shipping addresses, handling multiple destination addresses, gift giving options, and so on. The check boxes can be on a per-device basis, so that designers can choose which features will be available on which devices (see Figure 8-3). The labels of the check boxes can be supplemented with a more detailed description of the forces at work in the pattern. This mechanism could also be useful for pattern instances themselves. Within the design itself on the main canvas, sections of a pattern instance could also be made collapsible, or the entire pattern instance could be collapsed to one screen, to make it easier to see the main flow of the interaction.



Checking out should be a simple four-step process that funnels customers toward completion of an order.

Drag this label and drop into the main window to add to your design.

- [✓] Edit shopping cart
- [✓] Select shipping address
- [✓] Edit shipping address
- [✓] Select credit card
- [ ] Edit credit card
- [✓] Select billing address
- [ ] Edit billing address
- [✓] Select shipping method

Drag this label and drop into the main window to add to your design.

- [✓] Edit shopping cart

**FIGURE 8-3** Mockup of a checklist for choosing features within a pattern solution.

### 8.1.4 *Recognizing patterns*

Designing an interface often does not begin from scratch. Designers often take existing user interfaces, whether their own or someone else's, as a starting point for their design. Being able to take an existing interface, import it into Damask, and automatically detect patterns within the interface would greatly improve the utility of Damask. However, there are many unsolved issues in this area, including how patterns would be recognized, how designers would fix errors in the recognition process, and whether the recognition process would be accurate enough for designers to want to use it.

### 8.1.5 Versioning of patterns

Patterns evolve over time. As technologies change, the pattern solutions bundled with Damask may not remain the best ones. For example, Lazlo Systems has demonstrated a checkout process using Flash that is unlike most checkout processes for the web and is more similar to a traditional GUI interaction [101]. Setting aside whether Laszlo's system is better than current web-based systems, what would happen if designers could take their design with an instance of the original checkout pattern and replace it with an instance of a Laszlo-based pattern? Any changes made to the original instance would be lost if it were simply replaced, but the changes might not be relevant to the new instance. These issues need to be studied in greater detail.

### 8.1.6 Scope of patterns

Since the patterns in *The Design of Sites* [188], on which the Damask patterns are based, do not contain voice-specific solutions, we created the solutions ourselves. However, we have almost no experience designing voice UIs, and we did not have an opportunity to refine our solutions with the help of experienced voice UI designers. Improving our voice-specific solutions with the help of professional voice UI designers would go a long way towards making Damask more suitable for voice UI design.

Also, the patterns that Damask currently includes are most suitable for desktop web sites and not as appropriate for mobile phone or voice user interfaces. The patterns do not take advantage of the mobility and location-awareness that mobile devices have [34], or that people often do different tasks with different devices, even if it is the same high-level activity. For example, one could imagine a pattern that involves using standard web technologies for notifications on the desktop, but using SMS on mobile phones.

## 8.2 Annotations

Allowing designers to sketch annotations that are not directly part of the user interface is an important feature that is currently unaddressed in Damask. Annotations serve as design rationale and design history, and are valuable when presenting the idea to other people. Damask could be extended to include an annotation pen, which would not be interpreted by the system, as is done in SILK [99] (see Figure 8-4). Damask could also infer which object the annotation is attached to, so that it moves when the object does, as is done in The Designers' Outpost [90]. A menu item could turn annotations on or off.

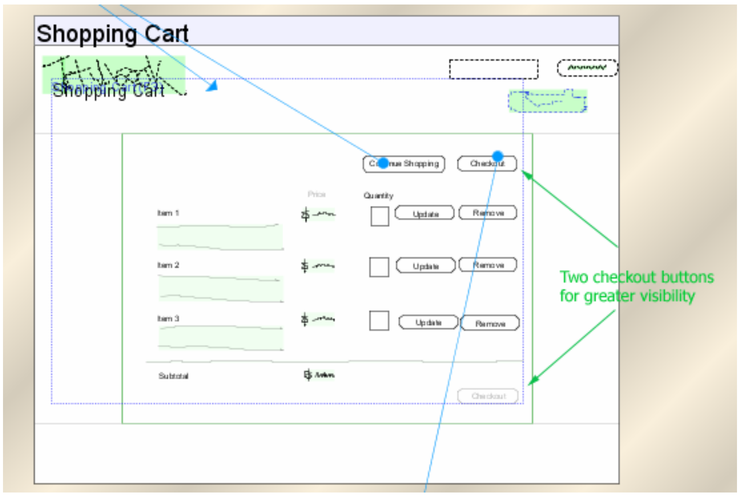


**FIGURE 8-4** Mockup of annotations (in green)

It would also be useful to add annotations to pattern solutions. This would make it easier to understand the solution, and the annotations could be merged directly into the main design when the pattern is instantiated. Embedding the rationale of the pattern within the design would allow the designer to see it in context, instead of in a separate window. However, too many annotations could also overwhelm designers; finding the right balance is critical.

## 8.3 Layers

Layers provide a powerful mechanism to control which devices an object appears in, but the current implementation is confusing. We have proposed several extensions and alternatives to layers. One of these ideas was presented to our Damask evaluation participants for some early feedback.

### *8.3.1 The "All" tab*

Currently, each tab has two layers, All Devices and This Device, and the devices in which an object appears depend on which layer you create the object on. Several designers suggested eliminating the layers, and instead adding a tab next to the three device tabs called All. An object created in the All tab would appear on all devices, while one created in, say, the Desktop tab would appear only on the desktop.

What layout would be displayed in the All tab? One option is to simply show the Desktop layout; to see the layout of the other devices, you would need to go to that device's tab. Another option is to have child tabs within the All tab for each device (see Figure 8-5).

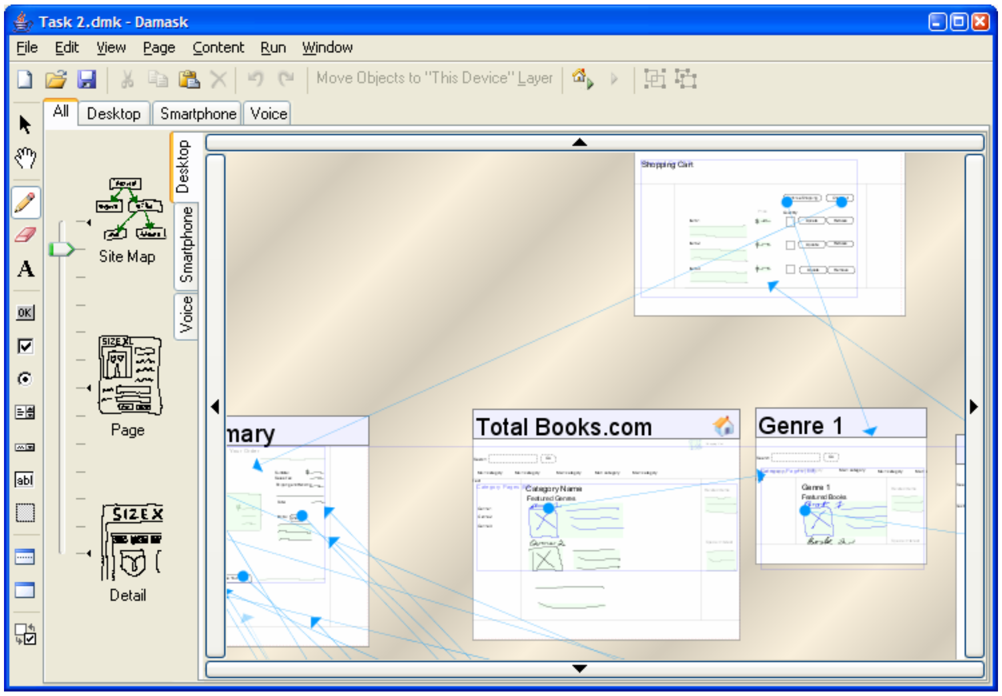

Task 2.dmk - Damask
File Edit View Page Content Run Window
Move Objects to "This Device" Layer
All Desktop Smartphone Voice
Desktop Smartphone Voice
Site Map
Page
Detail
Shopping Cart
mary
Total Books.com
Genre 1

**FIGURE 8-5** Mockup of an "All" tab in Damask.

### *8.3.2 Device-specific tools*

Another proposal is to have two sets of tools. One set would be for all devices, the other would be for one device. For example, in Section 6.9, we discussed how a designer who wanted to remove an all-device-button from the smartphone would have to go to Desktop view, select the button, and click Move to This Device. With device-specific tools, the designer would stay in Smartphone view, select the device-specific eraser, and tap on the button to erase it. Similarly, the all-device button tool would create a button visible on all devices, while the device-specific button would appear only on the current device. The device-specific text tool would change the text of an object for only one device, leaving the text for the other devices alone (see Figure 8-6).

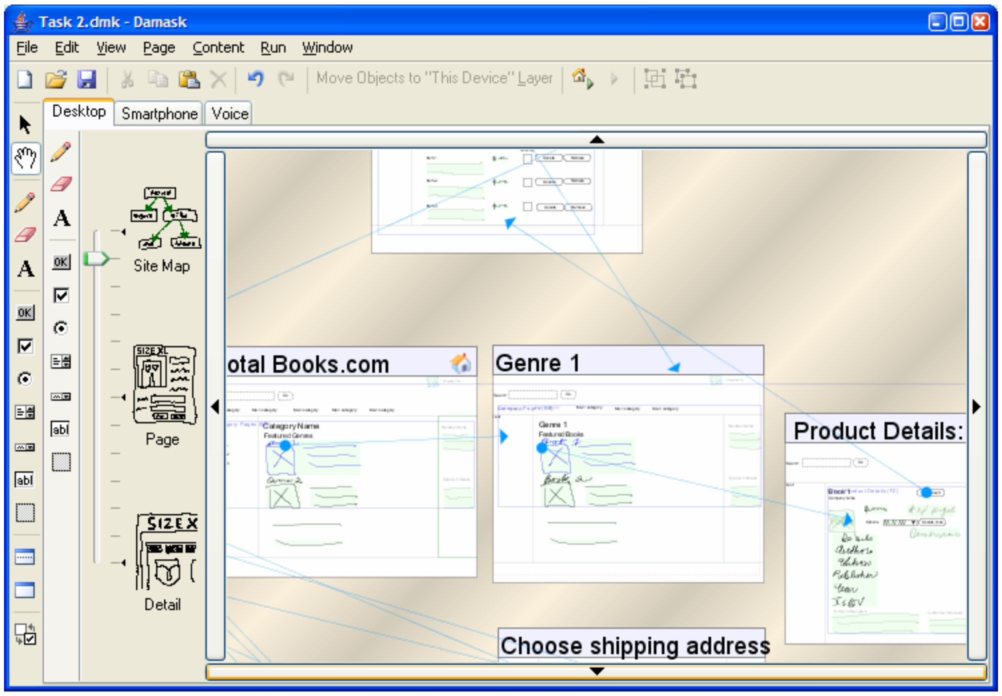
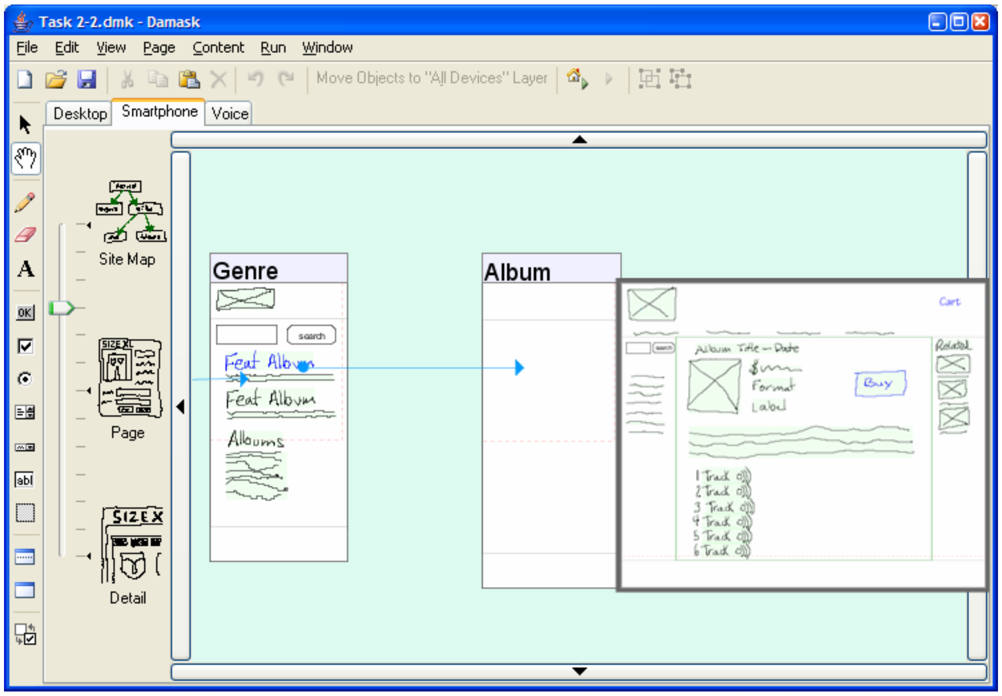


**FIGURE 8-6** Mockup of device-specific tools within the lighter-colored toolbox in the Desktop tab.

The objects would still need to be in different colors so that the designer could tell which objects were visible on all devices versus one device, and there would need to be a special button to convert a device-specific object to an all-devices object. Also, having two sets of tools may be confusing.

### *8.3.3 Trays*

In the Trays design, every page would have a "tray" associated with it, which the user could expand or collapse. When designers add a control to a page in one device type, the control would be added to the trays of the corresponding pages in the other device types. Designers could then go to another device type, look at the elements in a tray, and drag them onto the page itself if they decided they want those elements in the UI design for that device type. They could also click on the element to let Damask auto-position it (see Figure 8-7).



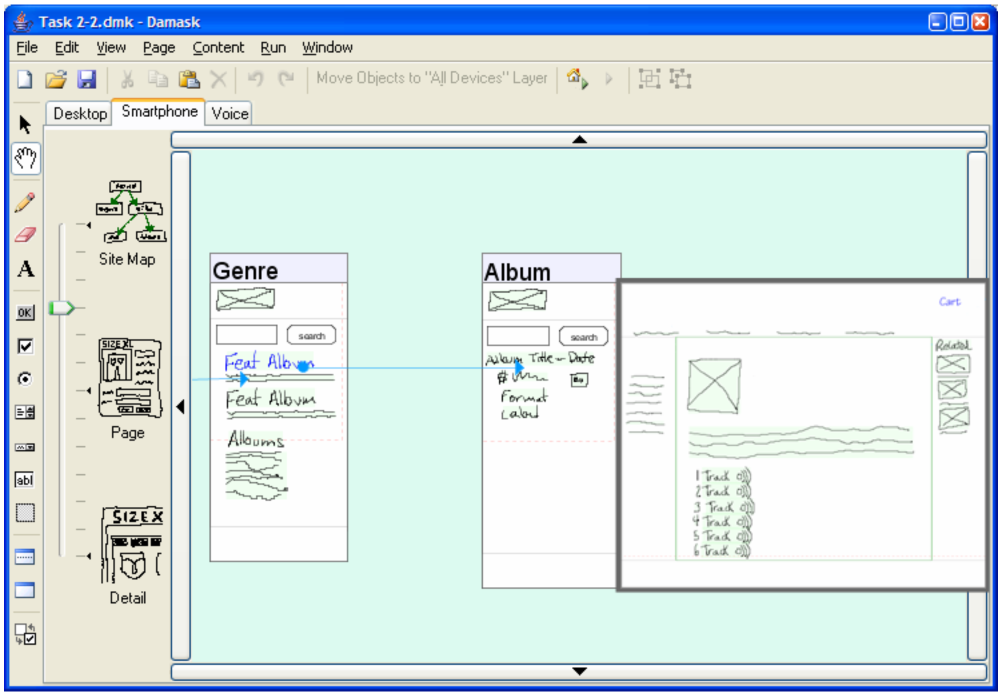


**FIGURE 8-7** *Top* A tray for the Album page, containing desktop objects to add. *Bottom* The Album page with objects added from the tray.

A variation of this idea is for elements to appear on all device types when added, just like how the All Devices layer currently works, and then designers would drag an element into the tray if they did *not* want it for that device type.

Compared to our current implementation of layers, trays allow finer control over which device type an object can appear in. In layers, an object can either be for All Devices or This Device only. In trays, an object can be, for example, in a page for desktop and smartphone views but not in the corresponding form for voice view. Trays can also be used to help designers learn how a control in desktop or smartphone views corresponds to one in voice view: as a control is dragged from the tray into the form, the control's corresponding voice version can be shown as a ghost image.

One challenge with the trays idea is letting designers know how editing an object in one device type affects the other device types. An object that is visible on more than one device type would need to be differentiated in some way, perhaps with a different color.

After each participant had finished the layers and patterns phase of the Damask evaluation, we presented the trays concept. When we asked whether they preferred layers or trays, they were evenly split.

Four designers preferred trays to layers. They thought trays would be less confusing than layers, and it would be easier for them to keep track of which elements were for all devices and which were for only one device. Interestingly, three out of the four designers who preferred trays were voice designers. It is much harder to figure out which controls in the desktop view correspond to the voice view and vice-versa, than it is to figure out the correspondence between a desktop control and a smartphone control. Therefore, the voice designers liked dragging a control from a tray to a voice form; it gave them more control. Designer 16 said, "I like [the tray] a lot better than having [controls] go over automatically."

Four designers preferred layers to trays; they found the trays idea a little too unfamiliar, or that it would require them to keep track mentally of more concepts. Designer 17 said, "It takes some getting used to the layers, but once you're used to them, it's clearer to see what [colors] means what and [on which devices] things are."

The last four did not have a strong preference. Designer 9 said that while trays offer "more control," the layers concept was still fine. However, she added that improving the automatic layout algorithm, as discussed in Section 4.4.1, would be necessary to make Damask more useful overall.

There is not a significant correlation between whether a designer preferred layers and whether that designer currently uses a tool that has layers, such as Photoshop or Visio.

## 8.4 Synchronization Between Voice and Graphical UI Designs

The degree of synchronization between voice and desktop/smartphone controls remains an open issue. Creating controls that exist on both the desktop and the smartphone makes sense, since they are visually similar. But creating controls that exist across desktop, smartphone, and voice are more problematic, since the voice controls and their corresponding desktop/smartphone controls look very different, and the mapping between them is not necessarily obvious. Therefore, the tight synchronization that now exists between desktop/smartphone and voice may not be necessary. For example, when a desktop control is created, the corresponding voice control could be created at the same time, so that the designer has something to start with. But any subsequent changes to one of them would not affect the other.

## 8.5 Extending Damask's Visual Language

Another issue is adding more capabilities to Damask's visual language of pages, components, and connections, while not detracting from its simplicity. Damask's visual language is very similar to DENIM's. In previous work, we extended DENIM's visual language to include custom components and conditional transitions [106]. Custom components are components that designers create themselves. Conditional transitions are transitions between pages where the endpoint of the transition depends on the state of the components in a page, such as the state of a check box.

Extending this to multiple devices presents new challenges, especially concerning synchronization. For example, if a transition is dependent on a certain set of component states in desktop view, should it also be dependent on the same states in the smartphone and voice views? If they should not, how does the designer indicate that?

Damask's visual language also does not handle user interface designs where certain parts are generated from a data source. There is no way to take the text typed into a text box, and display it in another page, or to display different data depending on that text. One of the voice interface designers said it is one of the primary reasons he would not use Damask in his daily tasks. Enabling more dynamic interfaces within the existing visual language paradigm will take more research.

## 8.6 Summary

Damask's approach to handling cross-device user interface design can be extended in many ways. This ranges from more complete support for using and creating design patterns and letting designers annotate their designs, to more sophisticated mechanisms for handing which parts of a UI design occur on which devices.

# 9 Conclusion

Here we summarize the benefits and limitations of Damask and review the contributions of the research.

## 9.1 Benefits of Damask's Approach

In current practice, designers who want to design a user interface that targets more than one type of device have essentially two choices. They can either design one UI for each device, or they can create one UI and use a tool to automatically generate the UIs for the other devices.

Damask allows designers to pursue a different path. Designers create a user interface design for one device by sketching the design, using layers to control which parts of the design appear on other devices, and instantiating design patterns. Damask creates user interface designs for other devices, giving designers a head start in designing for those devices. The informal interface and Run mode allow designers to quickly test and refine their ideas. The user studies I conducted show that designers can produce higher-quality user interfaces for multiple devices in less time than designing each interface independently, which is the current practice. Overall, designers especially liked the fluid interaction style and the design patterns in Damask.

## 9.2 Limitations of Damask's Approach

There were several issues that were outside Damask's research scope. As discussed in Section 4.4.1, Damask's algorithm for taking a UI in one device and automatically laying out the UI for the other devices is very simplistic and would need to be improved to ensure more widespread acceptance, possibly by using a combination of inferring which controls are grouped together and a more robust automatic layout algorithm.

The patterns that included interactive solutions were limited to one domain, e-commerce, and were originally designed for the desktop web. While these patterns turned out to be useful in the cross-device domain, discovering patterns related to issues specifically for cross-device applications, such as location-based applications or interacting with ubiquitous computing environments, would further improve the usefulness of Damask.

Damask's desktop and smartphone views are focused on web-style interaction. It is not clear whether they are appropriate for certain highly dynamic and interactive application domains, such as 3-D visualization and games. These domains are clearly beyond the scope of this work. Damask's voice view is geared towards guided prompt-and-response voice systems, as opposed to open-ended natural conversational systems. The latter would require significant new innovations, as that problem has not been solved even in voice-only design tools.

## 9.3 Contributions

Damask supports my thesis that:

> **A tool that uses design patterns to bridge the gap between device-specific UIs will enable designers to create cross-device UIs with at**

**least the same functionality as if the designer designed each device-specific UI separately, but in less time.**

Here we discuss each of the specific contributions of the research in more detail.

- *An understanding of current work practices for cross-device user interface design*

  As discussed in Chapter 2, we found that for designers currently doing cross-device UI design, consistency and dealing with multiple devices were major burdens, and that teams were usually organized so that a designer works on the same set of features across devices.

- *The novel application of the following techniques to cross-device user interface design:*

  1 *Design patterns allow designers to describe their designs at a high level of abstraction and, by capturing interaction semantics, make it easier for a design tool to create interfaces appropriate for different devices.*

  Damask provides a Pattern Browser for designers to explore a collection of design patterns and add them to their designs, as described in Section 4.5. The evaluation results described in Sections 6.3–6.10, especially 6.8, show patterns achieved their goal.

  2 *Layers can provide a clean conceptual model for designers to keep track of which UI elements are consistent across devices and which are device specific.*

  The Damask canvas provides layers for designers to add UI objects. These layers determine in which devices an object is visible, as described in Section 4.4. The evaluation results described in Section 6.3–6.10, especially Section 6.9, show that layers achieved their goal, albeit with some difficulty with the execution.

- *A data model to represent cross-device UIs that incorporates design patterns and layers and links corresponding elements across devices*

  As discussed in Section 4.6, Damask uses a single scenegraph of objects to represent a design. Each object contains both overall information and information specific to each device. The scenegraph also has pattern instance objects that point to the objects that make up the instance.

- *A user interface design tool called Damask that improves the design of cross-device UIs by implementing the concepts above*

Chapter 4 describes Damask's user interface and how it incorporated design patterns and layers throughout the application.

- *An understanding of how cross-device user interface design is influenced by design patterns and layers*

We discuss how designers used design patterns and layers in Chapter 6. Among other findings, we found designers using design patterns and layers for cross-device UI design completed their design tasks in less time than when they designed a UI for each device separately. Compared to designs created without patterns and layers, those created with patterns and layers were at least as good in terms of layout and interaction flow and were more complete. Also, the designers overall found design patterns to be generally useful.

## 9.4 Final Remarks

As computers become more pervasive [196], they will continue to take new shapes and forms. Applications that take advantage of the wide variety of devices will have an advantage over those limited to a PC, but the challenge of designing such applications is much higher. Design patterns and layers show great promise in helping user interface designers tackle the challenge of cross-device user interface design.

Damask is available for download at http://bid.berkeley.edu/damask and http://dub.washington.edu/damask.

# Bibliography

1. Amazon going mobile, *CNNfn*, 2000. http://money.cnn.com/2000/02/28/technology/amazon/
2. Adobe, *Illustrator*, 1985. Adobe Systems Inc.: San Jose, CA. http://www.adobe.com/products/illustrator/
3. Adobe, *Photoshop*, 1990. Adobe Systems Inc.: San Jose, CA. http://www.adobe.com/products/photoshop/
4. Alexander, Christopher, Sara Ishikawa, Murray Silverstein, Max Jacobson, Ingrid Fiksdahl-King, and Shlomo Angel, *A Pattern Language*. New York: Oxford University Press, 1977.
5. Ali, Mir Farooq, Manuel A. Pérez-Quiñones, Marc Abrams, and Eric Shell. Building Multi-Platform User Interfaces With UIML. In Proceedings of *2002 International Workshop of Computer-Aided Design of User Interfaces: CADUI'2002*. Valenciennes, France. pp. 225-236, May 15-17, 2002.
6. Alva, *outSPOKEN*, 1988. Alva Access Group (originally by Berkeley Systems, Inc.): Oakland, CA.
7. Ambiencia Information Systems, Inc., *Galaxy Application Environment*, 1992. Ambiencia Information Systems, Inc. (originally by Visix Software, Inc): Breckenridge, CO. http://www.ambiencia.com/
8. AppForge, *Crossfire*, 2004. AppForge, Inc.: Atlanta, GA. http://www.appforge.com/crossfire/
9. Apple, *HyperCard*, 1987. Apple Computer, Inc.: Cupertino, CA.
10. Apple, *Mac OS X Interface Builder*, 1991. Apple Computer Inc. (originally by NeXT Computer, Inc.): Cupertino, CA. http://developer.apple.com/tools/interfacebuilder.html
11. Arsanjani, Ali, David Chamberlain, Dan Gisolfi, Ravi Konuru, Julie Macnaught, Stephane Maes, Roland Merrick, David Mundel, T.V. Raman, Shankar Ramaswamy, Thomas Schaeck, Rich Thompson, Angel Diaz, John Lucassen,

and Charles F. Wiecha, *Web Service Experience Language, Version 2*: IBM Corporation, 2002. http://www.ibm.com/developerworks/library/ws-wsxl/

12. Asakawa, Chieko and Hironobu Takagi. Annotation-Based Transcoding for Nonvisual Web Access. In Proceedings of *The Fourth International ACM SIGCAPH Conference on Assistive Technologies: ASSETS 2000*. Arlington, VA. pp. 172-179, November 13-15, 2000.

13. Axure Software Solutions, *Axure RP*, 2003. Axure Software Solutions, Inc.: San Francisco, CA. http://www.axure.com/

14. Bailey, Brian P., *DEMAIS: A Behavior-Sketching Tool for Early Multimedia Design*, Unpublished Ph.D., Computer Science Department, University of Minnesota, Minneapolis, MN, 2002. http://orchid.cs.uiuc.edu/publications/tech-report-02-020.pdf

15. Bailey, Brian P. and Joseph A. Konstan, Are Informal Tools Better? Comparing DEMAIS, Pencil and Paper, and Authorware for Early Multimedia Design. *CHI 2003, ACM Conference on Human Factors in Computing Systems, CHI Letters*, 2003. **5**(1): pp. 313-320.

16. Balzert, Helmut, Frank Hofmann, Volker Kruschinski, and Christoph Niemann. The JANUS Application Development Environment—Generating More than the User Interface. In Proceedings of *1996 International Workshop of Computer-Aided Design of User Interfaces: CADUI '96*. Namur, Belgium: Namur University Press. pp. 183-205, June 5-7, 1996.

17. Banavar, Guruduth, Lawrence D. Bergman, Yves Gaeremynck, Danny Soroker, and Jeremy Sussman, Tooling and System Support for Authoring Multi-Device Applications. *Journal of Systems and Software (Special Issue on Ubiquitous Computing)*, 2004. **69**(3): pp. 227-242.

18. Banerjee, Somnath, Arobinda Gupta, and Anupam Basu. Online Transcoding of Web Pages for Mobile Devices. In Proceedings of *The 5th International Symposium on Human Computer Interaction with Mobile Devices and Services: Mobile HCI 2003*. Udine, Italy: Springer. pp. 271-285, September 8-11, 2003.

19. Baudisch, Patrick, Xing Xie, Chong Wang, and Wei-Ying Ma. Collapse-to-Zoom: Viewing Web Pages on Small Screen Devices by Interactively Removing Irrelevant Content. In Proceedings of *UIST 2004, ACM Symposium on User Interface Software and Technology*. Santa Fe, NM. pp. 91-94, 2004.

20. Beck, Kent and Ward Cunningham, *Using Pattern Languages for Object-Oriented Programs*. Technical Report CR-87-43, Tektronix, Inc., 1987.

21. Bederson, Benjamin B., Jesse Grosjean, and Jon Meyer, Toolkit Design for Interactive Structured Graphics. *IEEE Transactions on Software Engineering*, 2004. **30**(8): pp. 1-12.

22. Bergman, Lawrence D., Guruduth Banavar, Danny Soroker, and Jeremy Sussman. Combining Handcrafting and Automatic Generation of User-Interfaces for Pervasive Devices. In Proceedings of *2002 International Workshop of Computer-Aided Design of User Interfaces: CADUI'2002*. Valenciennes, France: May 15-17, 2002.

23. Bertini, Enrico and Giuseppe Santucci. Modelling Internet Based Applications for Designing Multi-Device Adaptive Interfaces. In Proceedings of *The Working Conference on Advanced Visual Interfaces: AVI 2004*. Gallipoli, Italy. pp. 252-256, May 25-28, 2004.

24. BeVocal, *BeVocal Café*, 1999. BeVocal, Inc.: Mountain View, CA. http://cafe.bevocal.com/

25. Bickmore, Timothy, Andreas Girgensohn, and Joseph W. Sullivan, Web Page Filtering and Re-Authoring for Mobile Users. *The Computer Journal*, 1999. **42**(6): pp. 534-546.

26. Bodart, François, Anne-Marie Hennebert, Jean-Marie Leheureux, and Jean Vanderdonckt. Computer-Aided Window Identification in TRIDENT. In Proceedings of *Fifth IFIP TC13 Conference on Human-Computer Interaction: INTERACT'95*. Lillehammer, Norway: Chapman & Hall. pp. 331-336, June 25-29, 1995.

27. Borchers, Jan, *A Pattern Approach to Interaction Design*. Chicester, England: John Wiley & Sons. 268 pp., 2001.

28. British Broadcasting Corporation, BBC Mobile. http://www.bbc.co.uk/mobile/

29. Budinsky, Frank J., Marilyn A. Finnie, John M. Vlissides, and Patsy S. Yu, Automatic Code Generation from Design Patterns. *IBM Systems Journal*, 1996. **35**(2): pp. 151-171.

30. Buyukkokten, Orkut, Hector Garcia-Molina, Andreas Paepcke, and Terry Winograd, Power Browser: Efficient Web Browsing for PDAs. *CHI 2000, ACM*

*Conference on Human Factors in Computing Systems, CHI Letters*, 2000. **2**(1): pp. 430-437.

31. Calvary, Gaëlle, Joëlle Coutaz, and David Thevenin. A Unifying Reference Framework for the Development of Plastic User Interfaces. In Proceedings of *Engineering for Human-Computer Interaction: EHCI 2001*. Toronto, ON, Canada: Springer-Verlag. pp. 173-192, May 11-13, 2001.

32. Charny, Ben and Jim Hu, Yahoo battles Google for the cell phone, *CNET News.com*, 2004. http://news.com.com/2100-1038_3-5428926.html

33. Chu, Hao-hua, Henry Song, Candy Wong, Shoji Kurakake, and Masaji Katagiri, Roam, a Seamless Application Framework. *Journal of Systems and Software (Special Issue on Ubiquitous Computing)*, 2004. **69**(3): pp. 209-226.

34. Chung, Eric, Jason I. Hong, James Lin, Madhu K. Prabaker, James A. Landay, and Alan L. Liu. Development and Evaluation of Emerging Design Patterns for Ubiquitous Computing. In Proceedings of *Designing Interactive Systems: DIS 2004*. Cambridge, MA. pp. 233–242, August 1–4, 2004.

35. Comber, Tim and John Maltby. Investigating Layout Complexity. In Proceedings of *1996 International Workshop of Computer-Aided Design of User Interfaces: CADUI '96*. Namur, Belgium: Namur University Press. pp. 211-229, June 5-7, 1996.

36. CONSENSUS Project, *RIML Language Specification, Version 2*. 105 pp., 2004. http://www.consensus-online.org/publicdocs/20040317-RIML-II-Final-public.pdf

37. Cover, Robin, *Cover Pages: Multi-Channel Access XML (MAXML)*, January 6, 2001. http://xml.coverpages.org/maxml.html

38. Davis, Richard C., James A. Landay, Victor Chen, Jonathan Huang, Rebecca B. Lee, Francis C. Li, James Lin, III Charles B. Morrey, Ben Schleimer, Morgan N. Price, and Bill N. Schilit. NotePals: Lightweight Note Sharing by the Group, for the Group. In Proceedings of *Human Factors in Computing Systems: CHI 99*. Pittsburgh, PA. pp. 338-345, May 15-20, 1999. http://dub.washington.edu/projects/notepals/pubs/notepals-chi99-final.pdf

39. Dery-Pinna, Anne-Marie, Jérémy Fierstone, and Emmanuel Picard. Component Model and Programming: A First Step to Manage Human Computer Interaction Adaptation. In Proceedings of *The 5th International Symposium on*

*Human Computer Interaction with Mobile Devices and Services: Mobile HCI 2003*. Udine, Italy: Springer. pp. 456-465, September 8-11, 2003.

40. Ding, Yun, Heiner Litz, and Dennis Pfisterer. A Graphical Single-Authoring Framework for Building Multi-platform User Interfaces. In Proceedings of *The 9th International Conference on Intelligent User Interfaces: IUI '04*. Madeira, Funchal, Portugal. pp. 235-237, January 13-16, 2004.

41. Dourish, Paul and André van der Hoek. Émigré: Metalevel Architecture and Migratory Work. In Proceedings of *The 4th International Symposium on Human Computer Interaction with Mobile Devices: Mobile HCI 2002*. Pisa, Italy: Springer. pp. 281-285, September 18-20, 2002.

42. Eclipse Foundation, *Standard Widget Toolkit*, 2001. Eclipse Foundation. http://www.eclipse.org/swt/

43. Eclipse Foundation, *Visual Editor Project*, 2004. Eclipse Foundation. http://www.eclipse.org/vep/

44. Ecma International, *Standard ECMA-262: ECMAScript Language Specification*. 3rd ed. Geneva. 172 pp., 1999. http://www.ecma-international.org/publications/standards/Ecma-262.htm

45. Elting, Christian, Stefan Rapp, Gregor Möhler, and Michael Strube. Architecture and Implementation of Multimodal Plug and Play. In Proceedings of *Fifth International Conference on Multimodal Interfaces: ICMI-PUI '03*. Vancouver, BC, Canada. pp. 93-100, November 5-7, 2003.

46. Fincher, Sally, Perspectives on HCI Patterns: Concepts and Tools (Introducing PLML). *Interfaces*, 2003(56): pp. 26-28.

47. Florijn, Gert, Marco Meijers, and Pieter van Winsen. Tool Support for Object-Oriented Patterns. In Proceedings of *European Conference for Object-Oriented Programming: ECOOP 97*. Jyväskylä, Finland: Springer-Verlag. pp. 472-495, June 9-13, 1997.

48. Fogarty, James and Scott E. Hudson, GADGET: A Toolkit for Optimization-Based Approaches to Interface and Display Generation. *UIST 2003, ACM Symposium on User Interface Software and Technology, CHI Letters*, 2003. 5(2): pp. 125-134.

49. Foley, James D. and Piyawadee “Noi” Sukaviriya. History, Results and Bibliography of the User Interface Design Environment (UIDE), an Early

Model-Based System for User Interface Design and Implementation. In Proceedings of *Design, Specification and Verification of Interactive Systems: DSV-IS'94*. Carrara, Italy. pp. 3-14, June 8-10, 1994.

50. Forbrig, Peter, Anke Dittmar, Daniel Reichart, and Daniel Sinnig. User-Centred Design and Abstract Prototypes. In Proceedings of *Perspectives in Business Informatics Research: BIR 2003*. Berlin, Germany: Shaker Verlag. pp. 132-145, September 18-20, 2003.

51. Fox, Armando, Ian Goldberg, Steven D. Gribble, David C. Lee, Anthony Polito, and Eric A. Brewer. Experience With Top Gun Wingman: A Proxy-Based Graphical Web Browser for the 3Com PalmPilot. In Proceedings of *IFIP International Conference on Distributed Systems Platforms and Open Distributed Processing: Middleware '98*. Lake District, UK, September 15-18, 1998.

52. Frank, Martin R. Grizzly Bear: A Demonstrational Learning Tool for a User Interface Specification Language. In Proceedings of *ACM Symposium on User Interface Software and Technology: UIST '95*. Pittsburgh, PA. pp. 75-76, November 15-17, 1995.

53. Frank, Martin R., Piyawadee “Noi” Sukaviriya, and James D. Foley. Inference Bear: Designing Interactive Interfaces through Before and After Snapshots. In Proceedings of *ACM Symposium on Designing Interactive Systems: DIS '95*. Ann Arbor, MI. pp. 167-175, August 23-25, 1995.

54. Freedom Scientific, *JAWS for Windows*, 1995. Freedom Scientific, Inc. (originally by Henter-Joyce, Inc.): St. Petersburg, FL. http://www.freedomscientific.com/fs_products/software_jaws.asp

55. Gajos, Krzysztof and Daniel S. Weld. SUPPLE: Automatically Generating User Interfaces. In Proceedings of *The 2004 International Conference on Intelligent User Interfaces: IUI '04*. Madeira, Funchal, Portugal. pp. 93-100, January 13-16, 2004.

56. Gamma, Erich, Richard Helm, Ralph Johnson, and John Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley Professional Computing Series. Reading, MA: Addison-Wesley. 395 pp., 1995.

57. Geary, David M., *Graphic Java 1.2, Volume 1: AWT*. 3rd ed: Prentice Hall PTR. 970 pp., 1998.

58. Geary, David M., *Graphic Java 2, Volume 2: Swing*. 3rd ed: Prentice Hall PTR. 970 pp., 1998.

59. González-Castaño, Francisco J., Luis E. Anido-Rifón, and Enrique Costa-Montenegro. A New Transcoding Technique for PDA Browsers, Based on Content Hierarchy. In Proceedings of *4th International Symposium on Human Computer Interaction with Mobile Devices: Mobile HCI 2002*. Pisa, Italy: Springer. pp. 69-80, September 18-20, 2002.

60. Gould, John D. and Clayton Lewis, Designing for Usability: Key Principles and What Designers Think. *Communications of the ACM*, 1985. **28**(3): pp. 300-311.

61. Gross, Mark D. and Ellen Yi-Luen Do. Ambiguous Intentions: A Paper-like Interface for Creative Design. In Proceedings of *ACM Symposium on User Interface Software and Technology: UIST '96*. Seattle, WA. pp. 183-192, November 6–8, 1996.

62. Grundy, John and Biao Yang. An Environment For Developing Adaptive, Multi-Device User Interfaces. In Proceedings of *The 4th Australasian User Interface Conference: AUIC2003*. Adelaide, Australia, Feburary 4-7, 2003.

63. Harada, Komei, Eiichiro Tanaka, Ryuichi Ogawa, and Yoshinori Hara. *Anecdote*: A Multimedia Storyboarding System with Seamless Authoring Support. In Proceedings of *ACM International Multimedia Conference 96*. Boston, MA. pp. 341-351, November 18-22, 1996.

64. Harrison, Neil B. The Language of Shepherding: A Pattern Language for Shepherds and Sheep. In Proceedings of *The 6th Annual Conference on Pattern Languages of Programming: PLoP 1999*. Monticello, IL, August 15-18, 1999. http://hillside.net/language-of-shepherding.pdf

65. Hearst, M. A., M. D. Gross, J. A. Landay, and T. E. Stahovich, Sketching Intelligent Systems. *IEEE Intelligent Systems*, 1998. **13**(3): pp. 10-19.

66. Hinrichs, Tom, Ray Bareiss, Lawrence Birnbaum, and Gregg Collins. An Interface Design Tool Based on Explicit Task Models. In Proceedings of *CHI '96 Conference Companion on Human Factors in Computing Systems*. Vancouver, BC, Canada: ACM Press. pp. 269-270, April 13-18, 1996.

67. Hinz, Michael, Zoltán Fiala, and Frank Wehner. Personalization-Based Optimization of Web Interfaces for Mobile Devices. In Proceedings of *The 6th International Symposium on Human Computer Interaction with Mobile Devices and Services: Mobile HCI 2004*. Glasgow, Scotland: Springer. pp. 204-215, September 13-16, 2004.

68. Hitwise Pty. Ltd., *A Merry Christmas For E-Commerce: 2004 Online Holiday Shopping Up 26%*, December 28, 2004. http://www.hitwise.com/info/news/hitwiseHS2004/wrapUp_Issue6.html

69. Hodes, Todd, Mark Newman, Steven McCanne, Randy Katz, and James Landay. Shared Remote Control of a Videoconferencing Application: Motivation, Design, and Implementation. In Proceedings of *SPIE Multimedia Computing and Networking: MMCN '99*. San Jose, CA. pp. 17-28, January 25-27, 1999.

70. Hong, Jason I., Jeffrey Heer, Sarah Waterson, and James A. Landay, WebQuilt: A Proxy-based Approach to Remote Web Usability Testing. *ACM Transactions on Information Systems*, 2001. **19**(3): pp. 263-285.

71. Hong, Jason I. and James A. Landay, SATIN: A Toolkit for Informal Ink-based Applications. *UIST 2000, ACM Symposium on User Interface Software and Technology, CHI Letters*, 2000. **2**(2): pp. 63-72.

72. Huang, Anita W. and Ned Sundaresan. Aurora: A Conceptual Model for Web-Content Adaptation to Support the Universal Usability of Web-based Services. In Proceedings of *ACM Conference on Universal Usability: CUU 2000*. Arlington, VA. pp. 124-131, November 16-17, 2000.

73. Hudson, Scott E., Bonnie E. John, Keith Knudsen, and Michael D. Byrne, A Tool for Creating Predictive Performance Models from User Interface Demonstrations. *UIST 99, ACM Symposium on User Interface Software and Technology, CHI Letters*, 1999. **1**(1): pp. 93-102.

74. Hussey, Andrew and David Carrington, *Using Patterns in Model-based Design*. Technical Report 99-15, Software Verification Research Centre, School of Information Technology, University of Queensland, Queensland, Australia, March, 1999. http://svrc.it.uq.edu.au/Bibliography/svrc-tr.html?99-15

75. IBM, *Abstract User Interface Markup Language Toolkit*, 2004. IBM Corp.: Armonk, NY. http://www.alphaworks.ibm.com/tech/auiml/

76. IBM, *IBM WebSphere Studio Device Developer*, 2002. IBM Corp.: Armonk, NY. http://www.ibm.com/software/wireless/wsdd/

77. IBM, *Rational Rose XDE Developer*, 2002. IBM Corp., (originally by Rational Software Corp.): Armonk, NY. http://www.ibm.com/software/awdtools/developer/rosexde/

78. IBM, *Rational Software Architect*, 2004. IBM Corp.: Armonk, NY. http://www.ibm.com/software/awdtools/architect/swarchitect/

79. IBM, *Rational Software Modeler*, 2004. IBM Corp.: Armonk, NY. http://www.ibm.com/software/awdtools/modeler/swmodeler/

80. International Committee for Information Technology Standards (INCITS), V2 Technical Committee on Information Technology Access Interfaces. http://www.incits.org/tc_home/v2.htm

81. iRise, *iRise Studio*, 2003. iRise: El Segundo, CA. http://www.irise.com/

82. Javahery, Homa, Ahmed Seffah, Daniel Engelberg, and Daniel Sinnig, Migrating User Interfaces Across Platforms Using HCI Patterns, in *Multiple User Interfaces: Cross-Platform Applications and Context-Aware Interfaces*, Ahmed Seffah and Homa Javahery, Editors. John Wiley & Sons: Chichester, England, UK. pp. 241-259, 2003.

83. John, Bonnie E., Konstantine Prevas, Dario D. Salvucci, and Ken Keodinger, Predictive Human Performance Modeling Made Easy. *CHI 2004, ACM Conference on Human Factors in Computing Systems, CHI Letters*, 2004. **6**(1): pp. 466-473.

84. Katsurada, Kouichi, Yusaku Nakamura, Hirobumi Yamada, and Tsuneo Nitta. XISL: A Language for Describing Multimodal Interaction Scenarios. In Proceedings of *Fifth International Conference on Multimodal Interfaces: ICMI-PUI '03*. Vancouver, BC, Canada. pp. 281-284, November 5-7, 2003.

85. Kawai, Shiro, Hitoshi Aida, and Tadao Saito. Designing Interface Toolkit with Dynamic Selectable Modality. In Proceedings of *International ACM Conference on Assistive Technologies: ASSETS '96*. Vancouver, BC, Canada. pp. 72-79, April 11-12, 1996.

86. Kelley, John F., An Iterative Design Methodology for User-Friendly Natural Language Office Information Applications. *ACM Transactions on Office Information Systems*, 1984. **2**(1): pp. 26-41.

87. Keynote Systems, *Online Retail Competition Heating Up as Consumers Become Savvier on the Web, Says Keynote*, February 16, 2005. http://www.keynote.com/news_events/releases_2005/05feb16.html

88. Kieras, David, A Guide to GOMS Model Usability Evaluation Using NGOMSL, in *The Handbook of Human-Computer Interaction*, Martin Helander,

Thomas Landauer, and Prasad Prabhu, Editors. North-Holland: Amsterdam. pp. 733-766, 1996.

89. Kim, Won Chul and James D. Foley. Providing High-level Control and Expert Assistance in the User Interface Presentation Design. In Proceedings of *Human Factors in Computing Systems: INTERCHI '93*. Amsterdam, The Netherlands: ACM Press. pp. 430-437, April 24-29, 1993.

90. Klemmer, Scott R., Mark W. Newman, Ryan Farrell, Mark Bilezikjian, and James A. Landay, The Designers Outpost: A Tangible Interface for Collaborative Web Site Design. *UIST 2001, ACM Symposium on User Interface Software and Technology, CHI Letters*, 2001. **3**(2): pp. 1–10.

91. Klemmer, Scott R., Anoop K. Sinha, Jack Chen, James A. Landay, Nadeem Aboobaker, and Annie Wang, SUEDE: A Wizard of Oz Prototyping Tool for Speech User Interfaces. *UIST 2000, ACM Symposium on User Interface Software and Technology, CHI Letters*, 2000. **2**(2): pp. 1-10.

92. Kost, Stefan, Dynamically Generated Multimodal Application Interfaces. Position paper for AVI 2004 Workshop: Developing User Interfaces with XML: Advances on User Interface Description Languages, 2004. http://krishna.imn.htwk-leipzig.de/ensonic/publications/paper-avi2004.pdf

93. Krasner, Glenn E. and Stephen T. Pope, A Description of the Model-View-Controller User Interface Paradigm in the Smalltalk-80 System. *Journal of Object-Oriented Programming*, 1988. **1**(3): pp. 26-49.

94. Kurlander, David, Chimera: Example-Based Graphical Editing, in *Watch What I Do: Programming by Demonstration*, Allen Cypher, Editor. MIT Press: Cambridge, MA. pp. 271-290, 1993.

95. Kurlander, David, *Graphical Editing by Example*, Unpublished Ph.D., Department of Computer Science, Columbia University, New York, 1993.

96. Kurlander, David and Eric Bier, Graphical Search and Replace. *Computer Graphics: Proceedings of SIGGRAPH 88*, 1988. **22**(4): pp. 113-120.

97. Lam, Heidi and Patrick Baudisch, Summary Thumbnails: Readable Overviews for Small Screen Web Browsers. *CHI 2005, ACM Conference on Human Factors in Computing Systems, CHI Letters*, 2005. **7**(1): pp. 681-690.

98. Landay, James A., *Interactive Sketching for the Early Stages of User Interface Design*, Unpublished Ph.D., Computer Science Department, Carnegie Mellon

University, Pittsburgh, PA, 1996.
http://www.cs.berkeley.edu/~landay/research/publications/Thesis.pdf

99. Landay, James A. and Brad A. Myers, Sketching Interfaces: Toward More Human Interface Design. *IEEE Computer*, 2001. **34**(3): pp. 56-64.

100. Landay, James A. and Brad A. Myers. Sketching Storyboards to Illustrate Interface Behavior. In Proceedings of *CHI '96 Conference Companion on Human Factors in Computing Systems*. Vancouver, BC, Canada. pp. 193-194, April 13–18, 1996.

101. Laszlo Systems, Amazon Store Demo, 2004.
http://www.laszlosystems.com/lps/sample-apps/amazon/amazon2.lzx?lzt=html

102. Lawson, Jeff. Automated Rich-Client Generation from XML Schemas. In Proceedings of *XML Conference & Exposition 2004*. Washington, DC, November 15-19, 2004. http://xchainj.com/XML2004/XchainJ3.zip

103. Lewis, Clayton and John Rieman, *Task-Centered User Interface Design: A Practical Introduction*. Boulder, CO: University of Colorado, 1993.
ftp://ftp.cs.colorado.edu/pub/cs/distribs/clewis/HCI-Design-Book/

104. Lin, James and James A. Landay. Damask: A Tool for Early-Stage Design and Prototyping of Multi-Device User Interfaces. In Proceedings of *The 8th International Conference of Distributed Multimedia Systems (2002 International Workshop on Visual Computing)*. San Francisco, CA. pp. 573-580, Sept. 26-28, 2002.

105. Lin, James, Mark W. Newman, Jason I. Hong, and James A. Landay, DENIM: Finding a Tighter Fit Between Tools and Practice for Web Site Design. *CHI 2000, ACM Conference on Human Factors in Computing Systems, CHI Letters*, 2000. **2**(1): pp. 510-517.

106. Lin, James, Michael Thomsen, and James A. Landay, A Visual Language for Sketching Large and Complex Interactive Designs. *CHI 2002, ACM Conference on Human Factors in Computing Systems, CHI Letters*, 2002. **4**(1): pp. 307-314.

107. Lonczewski, Frank and Siegfried Schreiber. The FUSE-System: An Integrated User Interface Design Environment. In Proceedings of *1996 International Workshop of Computer-Aided Design of User Interfaces: CADUI '96*. Namur, Belgium: Namur University Press. pp. 37-56, June 5-7, 1996.

108. Lopez, Juan F. and Pedro Szekely, Web Page Adaptation for Universal Access, in *Universal Access in HCI: Towards an Information Society for All (Proceedings of 1st International Conference on Universal Access in Human-Computer Interaction, New Orleans, LA, August 8-10, 2001)*, Constantine Stephanidis, Editor. Lawrence Erlbaum Associates: Mahwah, NJ. pp. 690-694, 2001.

109. López-Jaquero, Victor, Francisco Montero, José Pascual Molina, Antonio Fernández-Caballero, and Pascual González. Model-Based Design of Adaptive User Interfaces Through Connectors. In Proceedings of *The 10th International Workshop on the Design, Specification, and Verification of Interactive Systems: DSV-IS 2003*: Springer. pp. 245-257, 2003.

110. Luyten, Kris, Chris Vandervelpen, and Karin Coninx. Migratable User Interface Descriptions in Component-Based Development. In Proceedings of *The 9th International Workshop on the Design, Specification, and Verification of Interactive Systems: DSV-IS 2002*. Rostock, Germany. pp. 44-58, June 12-14, 2002.

111. MacKay, Bonnie, Carolyn Watters, and Jack Duffy. Web Page Transformation When Switching Devices. In Proceedings of *The 6th International Symposium on Human Computer Interaction with Mobile Devices and Services: Mobile HCI 2004*. Glasgow, Scotland: Springer. pp. 228-239, September 13-16, 2004.

112. Macromedia, *Director*, 1989. Macromedia, Inc. (originally by MacroMind, Inc.): San Francisco, CA. http://www.macromedia.com/software/director/

113. Macromedia, *Dreamweaver*, 1997. Macromedia, Inc.: San Francisco, CA. http://www.macromedia.com/software/dreamweaver/

114. Macromedia, *Flash*, 1996. Macromedia, Inc. (originally by FutureWave Software, Inc.): San Francisco, CA. http://www.macromedia.com/software/flash/

115. Martelli, Alex, Anna Ravenscroft, and David Ascher, *Python Cookbook*. 2nd ed. Sebastopol, CA: O'Reilly Media. 807 pp., 2005. Based on the online Python Cookbook at http://aspn.activestate.com/ASPN/Python/Cookbook/

116. Mattis, Peter, Spencer Kimball, Josh MacDonald, and GTK+ Team, *GTK+: The GIMP Toolkit*, 1997. GTK+ Team. http://www.gtk.org/

117. Mayora-Ibarra, Oscar, Edgar Cambranes-Martínez, Carlos Miranda-Palma, Alejandro Fuentes-Penna, and Oscar De la Paz-Arroyo. UML Modelling of Device-Independent Interfaces and Services for a Home Environment Application. In Proceedings of *The 4th International Symposium on Human*

*Computer Interaction with Mobile Devices: Mobile HCI 2002*. Pisa, Italy: Springer. pp. 296-301, September 18-20, 2002.

118. Meijler, Theo Dirk, Serge Demeyer, and Robert Engel. Making Design Patterns Explicit in FACE, a Framework Adaptive Composition Environment. In Proceedings of *European Software Engineering Conference and ACM SIGSOFT Symposium on the Foundations of Software Engineering: ESEC/FSE '97*: Springer-Verlag LNCS 1301. pp. 94-110, 1997.

119. Menkhaus, Guido and Wolfgang Pree. User Interface Tailoring for Multi-Platform Service Access. In Proceedings of *International Conference on Intelligent User Interfaces: IUI 2002*. San Francisco, CA. pp. 208-209, January 13-16, 2002.

120. Microsoft, *ASP.NET Mobile Controls*, 2002. Microsoft Corp.: Redmond, WA. http://msdn.microsoft.com/mobility/othertech/asp.netmc/

121. Microsoft, *FrontPage*, 1996. Microsoft Corp. (originally by Vermeer Technologies, Inc.): Redmond, WA. http://office.microsoft.com/frontpage/

122. Microsoft, *Visio*, 1992. Microsoft Corp. (originally by Visio Corp.): Seattle, WA. http://office.microsoft.com/visio/

123. Microsoft, *Visual Basic*, 1991. Microsoft Corp.: Redmond, WA. http://msdn.microsoft.com/vbasic/

124. Microsoft, *Visual Studio*, 1997. Microsoft Corp.: Redmond, WA. http://msdn.microsoft.com/vstudio/

125. Microsoft, *Visual Studio .NET 2003*, 2003. Microsoft Corp.: Redmond, WA. http://msdn.microsoft.com/vstudio/

126. microTOOL, *objectiF*. microTOOL GmbH: Berlin, Germany. http://www.microtool.de/objectif/en/

127. ModelMaker, *ModelMaker*. ModelMaker Tools: Oosterbeek, Netherlands. http://www.modelmakertools.com/

128. Modugno, Francesmary, *Extending End-User Programming in a Visual Shell with Programming by Demonstration and Graphical Language Techniques*, Unpublished Ph.D., Computer Science Department, Carnegie Mellon University, Pittsburgh, PA, 1995.

129. Modugno, Francesmary and Brad A. Myers, Graphical Representation and Feedback in a PBD System, in *Watch What I Do: Programming by Demonstration*, Allen Cypher, Editor. MIT Press: Cambridge, MA. pp. 415-422, 1993.

130. Molina, Pedro J., Santiago Meliá, and Oscar Pastor. JUST-UI: A User Interface Specification Model. In Proceedings of *The 4th International Conference on Computer-Aided Design of User Interfaces: CADUI 2002*. Valenciennes, France. pp. 323-334, May 15-17, 2002.

131. Moran, Thomas P., Patrick Chiu, and William van Melle. Pen-Based Interaction Techniques For Organizing Material on an Electronic Whiteboard. In Proceedings of *the ACM Symposium on User Interface Software and Technology: UIST '97*. Banff, Alberta, Canada. pp. 45-54, October 14-17, 1997.

132. Mori, Giulio, Fabio Paternò, and Carmen Santoro. Tool Support for Designing Nomadic Applications. In Proceedings of *The 8th International Conference on Intelligent User Interfaces: IUI 2003*. Miami, FL. pp. 141-148, January 13-15, 2003.

133. Mozilla Foundation, *XML User Interface Language (XUL)*, 2001. Mozilla Foundation. http://www.mozilla.org/projects/xul/

134. Müller, Andreas, Peter Forbrig, and Clemens Cap. Model-Based User Interface Design Using Markup Concepts. In Proceedings of *The 8th International Workshop on the Design, Specification, and Verification of Interactive Systems: DSV-IS 2001*. Glasgow, Scotland, UK: Springer. pp. 16-27, June 13-15, 2001. http://www.dcs.gla.ac.uk/~johnson/papers/dsvis_2001/muller/

135. Myers, Brad A., Graphical User Interface Programming, in *Computer Science Handbook*, Allen B. Tucker, Editor. Chapman & Hall/CRC Press, Inc.: Boca Raton, FL. pp. 48-1 – 48-29, 2004.

136. Myers, Brad A., Rich McDaniel, Rob Miller, Brad Vander Zanden, Dario Giuse, David Kosbie, and Andrew Mickish, Our Experience with Prototype-Instance Object-Oriented Programming in Amulet and Garnet. *Interfaces*, 1998(39): pp. 4–9.

137. Myers, Brad A., Richard G. McDaniel, Robert C. Miller, Alan Ferrency, Andrew Faulring, Bruce D. Kyle, Andrew Mickish, Alex Klimovitski, and Patrick Doane, The Amulet Environment: New Models for Effective User Interface Software Development. *IEEE Transactions on Software Engineering*, 1997. **23**(6): pp. 347–365.

138. Mynatt, Elizabeth D. and W. Keith Edwards. An Architecture for Transforming Graphical Interfaces. In Proceedings of *ACM Symposium on User Interface*

*Software and Technology: UIST '94*. Marina del Rey, California. pp. 39-47, November 2-4, 1994.

139. NetBeans Community, *NetBeans*, 1997. NetBeans Community (originally by NetBeans, Inc.). http://www.netbeans.org/

140. Newman, Mark W. and James A. Landay. Sitemaps, Storyboards, and Specifications: A Sketch of Web Site Design Practice. In Proceedings of *DIS 2000: Designing Interactive Systems*. New York, New York. pp. 263-274, August, 2000.

141. Newman, Mark W., James Lin, Jason I. Hong, and James A. Landay, DENIM: An Informal Web Site Design Tool Inspired by Observations of Practice. *Human-Computer Interaction*, 2003. **18**(3): pp. 259–324.

142. Nichols, Jeffrey. Informing Automatic Generation of Remote Control Interfaces with Human Designs. In Proceedings of *Human Factors in Computing Systems: CHI 2002 Extended Abstracts*. Minneapolis, MN. pp. 864-865, April 20-25, 2002.

143. Nichols, Jeffrey, Brad A. Myers, Michael Higgins, Joseph Hughes, Thomas K. Harris, Roni Rosenfeld, and Mathilde Pignol, Generating Remote Control Interfaces for Complex Appliances. *UIST 2002, ACM Symposium on User Interfaces and Software Technology, CHI Letters*, 2002. **4**(2): pp. 161-170.

144. Novell, *AppWare*, 1994. Novell, Inc.: Provo, UT.

145. Nuance, *V-Builder*, 2000. Nuance Communications, Inc.: Menlo Park, CA. http://www.nuance.com/prodserv/nae.html

146. Olsen, Dan R., Scott E. Hudson, Raymond Chung-Man Tam, Genevieve Conaty, Matthew Phelps, and Jeremy M. Heiner. Speech Interaction with Graphical User Interfaces. In Proceedings of *IFIP TC.13 Conference on Human Computer Interaction: INTERACT2001*. Tokyo, Japan: IOS Press, 2001.

147. Olsen, Dan R., Sean Jefferies, Travis Nielsen, William Moyes, and Paul Fredrickson, Cross-modal Interaction using XWeb. *UIST 2000, ACM Symposium on User Interface Software and Technology, CHI Letters*, 2000: pp. 191-200.

148. OmniSphere, *OmniBuilder*. OmniSphere Information Systems Corporation: Toronto, Ontario, Canada. http://www.omnibuilder.com/

149. Ousterhout, John K., *Tcl and the Tk Toolkit*. 1st ed. Reading, MA: Addison-Wesley Professional. 480 pp., 1994. Tcl/Tk is available for download at http://www.tcl.tk/

150. Pagel, Bernd-Uwe and Mario Winter. Towards Pattern-Based Tools. In Proceedings of *European Conference on Pattern Languages of Programs: EuroPLoP '96*. Kloster Irsee, Germany, July 11-13, 1996. http://www.informatik.fernuni-hagen.de/import/pi3/publikationen/abstracts/EuroPLoP96.pdf

151. Palanque, Philippe, Rémi Bastide, and Louis Dourte. Contextual Help for Free with Formal Dialogue Design. In Proceedings of *5th International Conference on Human-Computer Interaction: HCI International '93*. Orlando, FL: Elsevier, August 8-13, 1993.

152. Paternò, Fabio. ConcurTaskTrees and UML: How to Marry Them? In Proceedings of *Workshop Towards a UML Profile for Interactive Systems Development (TUPIS 2000) at the Third International Conference on the Unified Modeling Language: UML 2000*. York, United Kingdom, October 2–3, 2000. http://giove.cnuce.cnr.it/Guitare/Document/ConcurTaskTrees_and_UML-new.htm

153. Paternò, Fabio, *Model-Based Design and Evaluation of Interactive Applications*. Applied Computing, ed. Ray Paul, Peter Thomas, and Jasna Kuljis. London: Springer-Verlag. 192 pp., 2000.

154. Paternò, Fabio and Menica Mezzanotte. Formal Verification of Undesired Behaviours in the CERD Case Study. In Proceedings of *Engineering for Human-Computer Interaction: EHCI '95*. Jackson Hole, WY: Chapman & Hall. pp. 213-226, August 14-18, 1995.

155. Ponnekanti, Shankar R., Brian Lee, Armando Fox, Pat Hanrahan, and Terry Winograd. ICrafter: A Service Framework for Ubiquitous Computing Environments. In Proceedings of *The 3rd International Conference on Ubiquitous Computing: Ubicomp 2001*. Atlanta, GA: Springer. pp. 56-75, 2001.

156. Providence Software Solutions, *eXtensible Virtual Toolkit (XVT)*, 1988. Providence Software Solutions, Inc. (originally by XVT Software, Inc.): Cary, NC. http://www.xvt.com/

157. Puerta, Angel. The Mecano Project: Comprehensive and Integrated Support for Model-Based Interface Development. In Proceedings of *1996 International*

*Workshop of Computer-Aided Design of User Interfaces: CADUI '96*. Namur, Belgium: Namur University Press. pp. 19-36, June 5-7, 1996.

158. Puerta, Angel, Michael Micheletti, and Alan Mak. The UI Pilot: A Model-Based Tool to Guide Early Interface Design. In Proceedings of *The 10th International Conference on Intelligent User Interfaces: IUI 2005*. San Diego, CA. pp. 215–222, Jan. 9–12, 2005.

159. Puerta, Angel R. and Jacob Eisenstein. XIML: A Common Representation for Interaction Data. In Proceedings of *2002 International Conference on Intelligent User Interfaces: IUI 2002*. San Francisco, CA. pp. 216-217, January 13-16, 2002.

160. Rumbaugh, James, Michael Blaha, William Premerlani, Frederick Eddy, and William Lorenson, *Object-Oriented Modeling and Design*. Englewood Cliffs, N.J.: Prentice Hall. 500 pp., 1991.

161. Saund, Eric, David J. Fleet, Daniel Larner, and James Mahoney, Perceptually Supported Image Editing of Text and Graphics. *UIST 2003, ACM Symposium on User Interface Software and Technology, CHI Letters*, 2003: pp. 183-192.

162. Saund, Eric and Thomas P. Moran. A Perceptually-Supported Sketch Editor. In Proceedings of *the ACM Symposium on User Interface Software and Technology: UIST '94*. Marina del Rey, CA. pp. 175-184, November 2–4, 1994.

163. Schilit, Bill N., Gene Golovchinksy, and Morgan N. Price. Beyond Paper: Supporting Active Reading with Free Form Digital Ink Annotations. In Proceedings of *Human Factors in Computing Systems: CHI '98*. Los Angeles, CA. pp. 249-256, April 18-23, 1998.

164. Schlungbaum, Egbert and Thomas Elwert. Automatic User Interface Generation from Declarative Models. In Proceedings of *1996 International Workshop of Computer-Aided Design of User Interfaces: CADUI '96*. Namur, Belgium: Namur University Press. pp. 3-18, June 5-7, 1996.

165. Schreiber, Siegfried. Specification and Generation of User Interfaces with the BOSS-System. In Proceedings of *East-West International Conference on Human-Computer Interaction: EWHCI'94*. St. Petersburg, Russia: Springer-Verlag. pp. 107-120, August 2-6, 1994.

166. Sears, Andrew. AIDE: A Step Toward Metric-Based Interface Development Tools. In Proceedings of *ACM Symposium on User Interface Software and Technology: UIST '95*. Pittsburgh, PA. pp. 101-110, November 15-17, 1995.

167. Sefika, Mohlalefi, Aamod Saney, and Roy H. Campbell. Monitoring Compliance of a Software System With Its High-Level Design Models. In Proceedings of *18th International Conference on Software Engineering: ICSE-18 '96*. Berlin, Germany. pp. 387-396, March 25-29, 1996.

168. Sinha, Anoop K., Scott R. Klemmer, and James A. Landay, Embarking on Spoken-Language NL Interface Design. *International Journal of Speech Technology*, 2002. **5**(2): pp. 159-169.

169. Sinnig, Daniel, Homa Javahery, Peter Forbrig, and Ahmed Seffah. The Complicity of Model-Based Approaches and Patterns for UI Engineering. In Proceedings of *Perspectives in Business Informatics Research: BIR 2003*. Berlin, Germany: Shaker Verlag. pp. 120-131, September 18-20, 2003.

170. Smart, Julian, Robert Roebling, Stefan Csomor, Markus Holzem, Vadim Zeitlin, Guilhem Lavaux, and Vaclav Slavik, *wxWidgets*, 1992. wxWidgets Team. http://www.wxwidgets.org/

171. Smith, Ian, *Support for Multi-Viewed Interfaces*, Unpublished Ph.D. Dissertation, College of Computing, Georgia Institute of Technology, Atlanta, GA, 1998.

172. Smith, Randall B. and David Ungar. Programming as an Experience: The Inspiration for Self. In Proceedings of *The 9th European Conference on Object Oriented Programming: ECOOP '95*. Aarhus, Denmark: Springer. pp. 303-330, August 7-11, 1995. http://research.sun.com/self/papers/programming-as-experience.html

173. Souchon, Nathalie and Jean Vanderdonckt. A Review of XML-Compliant User Interface Description Languages. In Proceedings of *The 10th International Workshop on the Design, Specification and Verification of Interactive Systems: DSV-IS 2003*. Funchal, Madeira Island, Portugal: Springer. pp. 377-391, June 11-13, 2003.

174. Sun Microsystems, *Abstract Window Toolkit*, 1995. Sun Microsystems, Inc.: Santa Clara, CA. http://java.sun.com/products/jdk/awt/

175. Sun Microsystems, *Java Foundation Classes (JFC/Swing)*, 1998. Sun Microsystems, Inc.: Santa Clara, CA. http://java.sun.com/products/jfc/

176. Sutton, Stephen and Ronald Cole. The CSLU Toolkit: Rapid Prototyping of Spoken Language Systems. In Proceedings of *The 10th Annual ACM Symposium*

*on User Interface Software and Technology: UIST '97*. Banff, Alberta, Canada. pp. 85-86, 1997.

177. Sutton, Stephen, David G. Novick, Ronald Cole, Pieter Vermeulen, Jacques de Villiers, Johan Schalkwyk, and Mark Fanty. Building 10,000 Spoken Dialogue Systems. In Proceedings of *The 4th International Conference on Spoken Language Processing: ICSLP 96*. Philadelphia, PA. pp. 709-712, October 3-6, 1996.

178. Szekely, Pedro. Retrospective and Challenges for Model-Based Interface Development. In Proceedings of *Design, Specification and Verification of Interactive Systems: DSV-IS'96*. Namur, Belgium. pp. 1-27, June 5-7, 1996.

179. Szekely, Pedro, Ping Luo, and Robert Neches. Beyond Interface Builders: Model-Based Interface Tools. In Proceedings of *Human Factors in Computing Systems: INTERCHI '93*. Amsterdam, The Netherlands: ACM Press. pp. 383-390, April 24-29, 1993.

180. Szekely, Pedro, Piyawadee "Noi" Sukaviriya, Pablo Castells, Jeyakumar Muthukumarasamy, and Ewald Salcher. Declarative Interface Models for User Interface Construction Tools: the Mastermind Approach. In Proceedings of *Engineering for Human-Computer Interaction: EHCI '95*. Jackson Hole, WY: Chapman & Hall. pp. 120-150, August 14-18, 1995.

181. Tellme, *Tellme Studio*, 2000. Tellme Networks, Inc.: Mountain View, CA. http://studio.tellme.com/

182. Tidwell, Jenifer, *Common Ground: A Pattern Language for Human-Computer Interface Design*, 1999. http://www.mit.edu/~jtidwell/common_ground.html

183. Tidwell, Jenifer, *Designing Interfaces: Patterns for Effective Interaction Design*. Sebastopol, CA: O'Reilly Media. 384 pp., 2005. http://time-tripper.com/uipatterns/

184. Trætteberg, Hallvard, Model based design patterns. April 1-6, 2000: Position paper for CHI 2000 Workshop: Pattern Languages for Interaction Design: Building Momentum. http://www.idi.ntnu.no/~hal/publications/design-patterns/CHI00-position.pdf

185. Trolltech, *Qt*, 1991. Trolltech AS: Oslo, Norway. http://www.trolltech.com/products/qt/

186. Unisys, *Natural Language Speech Assistant*, 1999. Unisys Corporation.

187. van de Kant, Maarten, Stephanie Wilson, Mathilde Bekker, Hilary Johnson, and Peter Johnson. PatchWork: A Software Tool for Early Design. In Proceedings of *CHI 98 Conference Summary on Human Factors in Computing Systems*. Los Angeles, CA. pp. 221-222, April 18-23, 1998.

188. van Duyne, Douglas K., James A. Landay, and Jason I. Hong, *The Design of Sites*. Boston: Addison-Wesley, 2002.

189. van Welie, Martijn and Hallvard Trætteberg. Interaction Patterns in User Interfaces. In Proceedings of *The 7th Pattern Languages of Programs Conference: PLoP 2000*. Monticello, Illinois, August 13-16, 2000. http://jerry.cs.uiuc.edu/~plop/plop2k/proceedings/Welie/Welie.pdf

190. Vanderdonckt, Jean, Quentin Limbourg, and Murielle Florins. Deriving the Navigational Structure of a User Interface. In Proceedings of *Ninth IFIP TC13 International Conference on Human-Computer Interaction: INTERACT 2003*. Zurich, Switzerland: IOS Press. pp. 455-462, September 1-5, 2003.

191. Voxeo, *Voxeo Evolution*, 2002. Voxeo Corp.: Orlando, FL. http://community.voxeo.com/

192. W3C HTML Working Group, *XHTML™ 1.0: The Extensible HyperText Markup Language (Second Edition)*: World Wide Web Consortium, 2002. http://www.w3.org/TR/xhtml1/

193. W3C Voice Browser Working Group, *Voice Extensible Markup Language (VoiceXML) Version 2.0*: World Wide Web Consortium, 2004. http://www.w3.org/TR/voicexml20/

194. W3C XForms Working Group, *XForms 1.0*: World Wide Web Consortium, 2003. http://www.w3.org/TR/xforms/

195. Wagner, Annette, Prototyping: A Day in the Life of an Interface Designer, in *The Art of Human-Computer Interface Design*, Brenda Laurel, Editor. Addison-Wesley: Reading, MA. pp. 79-84, 1990.

196. Weiser, Mark D., The Computer for the 21st Century. *Scientific American*, 1991. **265**(3): pp. 94–104.

197. Wiecha, Charles, William Bennett, Stephen Boies, John Gould, and Sharon Greene, ITS: A Tool for Rapidly Developing Interactive Applications. *ACM Transactions on Information Systems*, 1990. **8**(3): pp. 204-236.

198. Wiecha, Charles, Stephen Boies, Margaret Gaitatzes, Stephen Levy, Julie Macnaught, Paul Matchen, Scott Mcfaddin, David Mundel, and Rich Thompson, Position paper for CHI 2001 Workshop: Transforming the UI for Anyone. Anywhere. April 1-5, 2001: Seattle, WA.

199. Wilson, Stephanie and Peter Johnson. Bridging the Generation Gap: From Work Tasks to User Interface Designs. In Proceedings of *1996 International Workshop of Computer-Aided Design of User Interfaces: CADUI '96*. Namur, Belgium: Namur University Press. pp. 77-94, June 5-7, 1996.

200. Wong, Yin Yin. Rough and Ready Prototypes: Lessons From Graphic Design. In Proceedings of *Human Factors in Computing Systems: CHI '92*. Monterey, CA. pp. 83-84, May 3–7, 1992.

201. Zimmermann, Gottfried, Gregg Vanderheiden, and Al Gilman. Prototype Implementations for a Universal Remote Console Specification. In Proceedings of *Human Factors in Computing Systems: CHI 2002 Extended Abstracts*. Minneapolis, MN. pp. 510-511, April 20-25, 2002.