University of Technology, Sydney
Faculty of Engineering and Information Technology

Derivation of a General Purpose Architecture for Automatic User Interface Generation

Submitted by:

Richard Kennard

B. Sc (Computer Science) Hons. (1st)

2011

Supervisor: John Leaney

Submitted for the degree of

DOCTORATE OF PHILOSOPHY

Certificate of Authorship/Originality

I certify that the work in this thesis has not previously been submitted for a degree nor has it been submitted as part of requirements for a degree except as fully acknowledged within the text.

I also certify that the thesis has been written by me. Any help that I have received in my research work and the preparation of the thesis itself has been acknowledged. In addition, I certify that all information sources and literature used are indicated in the thesis.

Production Note:
Signature removed prior to publication.

Richard Kennard

Acknowledgements

Completion of this thesis would not have been possible without a number of people whom I would like to acknowledge and thank for their support, advice and encouragement.

My supervisor, John Leaney, was instrumental in opening me up to the world of academic rigour. I would like to thank him for challenging me to think about epistemologies and research methodologies, interview techniques, reflection, VVT and GQM, and a host of other disciplines. His support and enthusiasm for my work never faltered. Most of all I would like to thank him for being able to readily understand my goal, and bridge the divide between the industrial world I knew and the research community I aspired to join.

I would like to thank those people who posted forum messages, blogged, tweeted, wrote magazine articles, and published papers regarding my work. These people are not named in this thesis, but I include excerpts from their feedback and their words are publicly searchable online. I would like to thank my interviewees and adoption study participants, most of whom must remain anonymous but who know who they are, for their time and feedback. Given the Action Research methodology of this thesis, their observations were critical to my progress. I would like to thank Dan Haywood of the Naked Objects team, who was always friendly and supportive, and very professional considering the sometimes conflicting nature of our projects. I would also like to thank Dan Allen, for his help shepherding my research within Red Hat.

Finally I would like to thank and apologise to my family who lived with my long absences throughout – both physical absences during the weeks, and mental absences during the weekends when I couldn't drag my head out of the work.

Table of Contents

1. Introduction....1

1.1. Objective....1

1.2. Significance....2

1.3. Structure of this Thesis....3

2. Literature Review....4

2.1. User Interfaces....5

2.1.1. Automatic UI Generation....6

2.1.1.1. Interactive Graphical Specification Tools....7

2.1.1.2. Model-Based Generation Tools....8

2.1.1.3. Language-Based Tools....9

2.1.2. Other UI Framework Services....12

2.1.2.1. Reusable Widgets....12

2.1.2.2. Validation....13

2.1.2.3. Data Binding....13

2.1.2.4. Navigation....14

2.1.3. Difficulties of UI Development....14

2.2. Software Mining....15

2.2.1. Static Analysis....18

2.2.1.1. Source code....18

2.2.1.2. Externalised behaviour....18

2.2.2. Dynamic Analysis....19

2.2.2.1. Reflection....20

2.2.2.2. Embedded Metadata....20

2.2.3. Historic Analysis....21

2.2.4. Potential Benefits of Software Mining....22

2.2.5. Demonstration of Software Mining....23

2.2.5.1. Properties Subsystem....23

2.2.5.2. Persistence Subsystem....24

2.2.5.3. Validation Subsystem....25

2.2.5.4. Business Process Modelling Subsystems....26

2.2.5.5. Collating Software Mining Results....26

2.2.5.6. Limitations of Software Mining........29

2.3. Proposed Research........30

3. Research Method........32

3.1. Epistemology........32

3.2. Methodology........34

3.2.1. Action Research........34

3.2.1.1. Plan........35

3.2.1.2. Act........35

3.2.1.3. Observe........36

3.2.1.3.1. Grounded Theory........36

3.2.1.4. Reflect........37

3.3. Design........38

3.4. Ethical Issues........40

4. Action Research: Alpha Cycle........41

4.1. Planning........41

4.1.1. Reflections During Planning........41

4.1.1.1. Naming the Project........41

4.1.1.2. Technology Neutral Interfaces........42

4.1.1.3. Runtime Code Generation........43

4.1.1.4. Useful Bounds of Generation........44

4.1.1.5. Layouts........45

4.2. Acting........46

4.2.1. Reflections In Action........46

4.2.1.1. Widget Builders........47

4.2.1.2. Guided Software Mining........48

4.2.1.3. CompositeInspector........49

4.2.1.4. Papers........52

4.2.1.5. Experiments........53

4.2.1.5.1. Experiment 1........55

4.2.1.5.1.1. Synopsis........56

4.2.1.5.2. Experiment 2........58

4.2.1.5.2.1. Synopsis........58

4.2.2. Action Outcomes........61

4.2.2.1. Screenshots........61

4.2.2.2. UML........63

4.2.2.2.1. Immutability....67

4.2.2.3. Promotion....70

4.3. Observing....74

4.3.1. Reflections Following Observations....74

4.3.1.1. Interviews....74

4.3.1.1.1. Duplication....75

4.3.1.1.2. Defects....77

4.3.1.1.3. Prevalence....78

4.3.1.1.4. Conclusion....79

4.3.1.2. Self-Administered Survey....80

4.3.1.3. Forums....83

4.3.1.3.1. Rebinding....83

4.3.1.3.2. Conditional Expressions....84

4.3.1.3.3. 1-to-M relationships....85

4.3.1.3.4. SWT support....85

4.3.1.4. Blogs....87

4.3.1.4.1. Explicit field ordering....87

4.3.1.5. Adoption Studies....88

4.3.1.5.1. Adoption Study 1....90

4.3.1.5.1.1. Synopsis....91

4.3.1.5.1.2. Reflection....91

4.3.1.5.2. Adoption Study 2....93

4.3.1.5.2.1. Synopsis....94

4.3.1.5.2.2. Reflection....95

4.3.1.5.3. Adoption Study 3....96

4.3.1.5.3.1. Synopsis....96

4.3.1.5.3.2. Reflection....98

5. Action Research: Beta Cycle....100

5.1. Planning....100

5.1.1. Reflections During Planning....100

5.1.1.1. Reflection on Reflection....100

5.1.1.2. Widget Builders Revisited....101

5.1.1.3. Effectiveness....104

5.2. Acting....105

5.2.1. Reflections In Action......105
5.2.1.1. Exposure......105
5.2.1.2. Papers......107
5.2.1.3. Experiments......108
5.2.1.3.1. Experiment 3......108
5.2.2. Action Outcomes......*110*
5.2.2.1. UML......110
5.3. Observing......110
5.3.1. Reflections Following Observations......*110*
5.3.1.1. Adoption Studies......110
5.3.1.1.1. Adoption Study 4......112
5.3.1.1.1.1. Synopsis......*112*
5.3.1.1.1.2. Reflection......*114*
5.3.1.1.2. Adoption Study 5......116
5.3.1.1.2.1. Synopsis......*116*
5.3.1.1.2.2. Reflection......*117*
5.3.1.1.3. Adoption Study 6......119
5.3.1.1.3.1. Synopsis......*119*
5.3.1.1.3.2. Reflection......*122*
6. Action Research: Release Candidate Cycle......123
6.1. Planning......123
6.1.1. Reflections During Planning......*123*
6.1.1.1. Widget Processors......123
6.1.1.2. Inspection Result Processors......125
6.1.1.3. Decoratable Layouts......128
6.1.1.4. Generation Pipeline......130
6.2. Acting......131
6.2.1. Reflections In Action......*132*
6.2.1.1. Exposure......132
6.2.1.2. Journal Article......132
6.2.1.3. Performance Measurements......134
6.2.1.4. DomInspector......136
6.2.2. Action Outcomes......*138*
6.2.2.1. UML......138

6.3. Observing....140

6.3.1. Reflections Following Observations....140

6.3.1.1. Blogs....140

6.3.1.2. Validation, Verification and Testing....140

7. Validation....142

7.1. Research Community Validation....142

7.1.1. Methodology....143

7.1.1.1. Inspecting existing, heterogeneous back-end architectures....145

7.1.1.2. Appreciating different practices in applying inspection results....147

7.1.1.3. Recognising multiple, and mixtures of, UI widget libraries....147

7.1.1.4. Supporting multiple, and mixtures of, UI adornments....148

7.1.1.5. Applying multiple, and mixtures of, UI layouts....150

7.1.2. Conclusion....151

7.2. Industrial Validation....152

7.2.1. Methodology....152

7.2.1.1. Goals, Questions and Metrics (GQM)....153

7.2.2. Organisation and Product Overview....155

7.2.3. Integration of Metawidget....158

7.2.4. Validation of Metawidget....163

7.2.4.1. Obviousness....163

7.2.4.2. Convenience....164

7.2.4.3. Adaptability....165

7.2.4.4. Performance....167

7.2.5. Conclusion....167

8. Conclusion....169

8.1. Strengths....169

8.1.1. Contributions to Field....169

8.1.2. Industry Adoption....170

8.2. Challenges....172

8.2.1. Lack of Standardisation....172

8.2.2. Unbalanced User Documentation....173

8.3. Future Work....173

8.3.1. Tooling....173

8.3.2. Packaging....174

8.3.3. Metadata Validation....175

8.3.4. Release Train ... 176

8.3.5. Future Research ... 176

8.4. Closing Remarks ... 177

Table of Figures

Figure 1: The significant features combine to a greater whole..3

Figure 2: Graphical User Interface..6

Figure 3: Interactive graphical specification tool...8

Figure 4: Declarative HTML model..9

Figure 5: Language-based tool..11

Figure 6: UI constructed from metadata...29

Figure 7: The UI drives the software mining...49

Figure 8: CompositeInspector collates results and appears as a single Inspector externally.......52

Figure 9: Question screen before answer type selected...60

Figure 10: Question screen after answer type selected...60

Figure 11: Metawidget does not try to 'own' the UI..62

Figure 12: Five Metawidgets are used in the UI...63

Figure 13: UML class diagram of Alpha Action Research Cycle..66

Figure 14: metawidget.org Web site..71

Figure 15: Elevator pitch cartoon...72

Figure 16: Live demo running inside the Web browser...73

Figure 17: Neural network trapped in a local minima...101

Figure 18: CompositeWidgetBuilder can compose multiple Widget Builders together............103

Figure 19: Portions of code saved by retrofitting..109

Figure 20: UML class diagram of Beta Action Research Cycle..111

Figure 21: Widget Processors...124

Figure 22: Inspection Result Processors parallel Widget Processors.......................................128

Figure 23: Layout decorated with horizontal rules inside tabs...129

Figure 24: Layout decorated with tabs inside horizontal rules...129

Figure 25: Metawidget pipeline..130

Figure 26: Unnecessary serialization and deserialization..137

Figure 27: Optimised DOM passing...137

Figure 28: UML class diagram of Release Candidate Action Research Cycle..........................139

Figure 29: Naked Objects' Object Oriented User Interface..144

Figure 30: Naked Objects hexagonal architecture...145

Figure 31: Naked objects sequence diagram as implemented by Isis Wicket viewer................149

Figure 32: Health Portal administration..156

Figure 33: Health Portal scheduler...157

Figure 34: Simplified UML diagram of Health Portal...157

Figure 35: Metawidget is used while lodging individual claims..162

Figure 36: Metawidget is used while lodging multiple claims...162

Figure 37: Metawidget is used while printing invoices...163

Figure 38: Health Portal uses a custom inspector..165

Figure 39: Health Portal uses a custom widget processor..166

Glossary of Terms

API	Application Programming Interface
AST	Abstract Syntax Tree
BPM	Business Process Modelling
CLR	Common Language Runtime
CRUD	Create, Retrieve, Update and Delete
DRY	Don't Repeat Yourself
DSL	Domain Specific Language
ERP	Enterprise Resource Planning
GP	General Practitioner
GQM	Goals, Questions and Metrics
GUI	Graphical User Interface
HFD	Human Factors Designers
JAXB	Java API for XML Binding
JPA	Java Persistence Architecture
JSF	Java Server Faces
JSP	Java Server Pages
JVM	Java Virtual Machine
MVC	Model View Controller
NHS	National Health System
OID	Object Identifier
OOUI	Object Oriented User Interface
ORM	Object Relational Database Mapper
PDG	Program Dependency Graph
SSOT	Single Source of Truth
UI	User Interface
VVT	Validity, Verification and Testing
WYSIWYG	What You See Is What You Get

Abstract

Many software projects spend a significant proportion of their time developing the User Interface (UI), therefore any degree of automation in this area has clear benefits. Research projects to date generally take one of three approaches: interactive graphical specification tools, model-based generation tools, or language-based tools. The first two have proven popular in industry but are labour intensive and error-prone. The third is more automated but has practical problems which have led to a lack of industry adoption.

This thesis set out to understand and address these limitations. It studied the issues of UI generation in practice using Action Research cycles guided by interviews, adoption studies, case studies and close collaboration with industry practitioners. It further applied the emerging field of software mining to address some of these issues. Software mining is used to collate multiple inspections of an application's artefacts into a detailed model, which can then be used to drive UI generation. Finally, this thesis explicitly defined bounds to the generation, such that it can usefully automate some parts of the UI development process without restricting the practitioner's freedom in other parts. It proposed UI generation as a way to augment manual UI construction rather than replace it.

To verify the research, this thesis built an Open Source project using successive generations of Iterative Development, and released and promoted it to organisations and practitioners. It tracked and validated the project's reception and adoption within the community, with an ultimate goal of mainstream industry acceptance. This goal was achieved on a number of levels, including when the project was recognised by Red Hat, an industry leader in enterprise middleware. Red Hat acknowledged the applicability and potential of the research within industry and integrated it into their next generation products.