LIBRARY
Te Tumu Herenga
THE UNIVERSITY OF AUCKLAND

RESEARCHSPACE@AUCKLAND

http://researchspace.auckland.ac.nz

ResearchSpace@Auckland

Copyright Statement

The digital copy of this thesis is protected by the Copyright Act 1994 (New Zealand).

This thesis may be consulted by you, provided you comply with the provisions of the Act and the following conditions of use:

- Any use you make of these documents or images must be for research or private study purposes only, and you may not make them available to any other person.
- Authors control the copyright of their thesis. You will recognise the author's right to be identified as the author of this thesis, and due acknowledgement will be made to the author where appropriate.
- You will obtain the author's permission before publishing any material from their thesis.

To request permissions please use the Feedback form on our webpage.
http://researchspace.auckland.ac.nz/feedback

General copyright and disclaimer

In addition to the above conditions, authors give their consent for the digital copy of their work to be used subject to the conditions specified on the Library Thesis Consent Form.

Multiple textual and graphical views for Interactive Software Development Environments

John Collis Grundy

A thesis submitted in fulfilment of the requirements for the degree of

Doctor of Philosophy in Computer Science

University of Auckland June 1993

Abstract

Diagram construction can be used to visually analyse and design a complex software system using natural, graphical representations describing high-level structure and semantics. Textual programming can specify detailed documentation and functionality not well expressed at a visual level. Integrating multiple textual and graphical views of software development allows programmers to utilise both representations as appropriate. Consistency management between these views must be automatically maintained by the development environment.

MViews, a model for such software development environments, has been developed. MViews supports integrated textual and graphical views of software development with consistency management. MViews provides flexible program and view representation using a novel object dependency graph approach. Multiple views of a program may contain common information and are stored as graphs with textual or graphical renderings and editing. Change propagation between program components and views is supported using a novel update record mechanism. Different editing tools are integrated as views of a common program repository and new program representations and editors can be integrated without affecting existing views.

A specification language for program and view state and manipulation semantics, and a visual specification language for view appearance and editing semantics, have been developed. An object-oriented architecture based on MViews abstractions allows environment specifications to be translated into a design for implementing environments. Environment designs are implemented by specialising a framework of object-oriented language classes based on the MViews architecture. A new language is described which provides object-oriented extensions to Prolog. An integrated software development environment for this language is discussed and the specification, design and implementation of this environment using MViews are described. MViews has also been reused to produce a graphical entity-relationship/textual relational database schema modeller, a dialogue painter with a graphical editing view and textual constraints view, and various program visualisation systems.

Acknowledgments

My supervisor John Hosking has been a tremendous guiding influence during the research for and writing of this thesis. He has helped pick my spirits up when down, encouraged me when things have gone well and kept a sometimes way-ward spirit on the right path too many times to mention. I have greatly valued his friendship, constructive criticism and excellent supervision during the course of this work. I also look forward to continuing professional and personal relationships for many years to come. Thanks for everything, John.

I would also like to thank various colleagues for their helpful comments about my research. Particular thanks go to Robert Amor, Stephen Fenwick, Rick Mugridge and John Hamer from the University of Auckland and James Noble from Victoria University of Wellington.

The staff and students of the Department of Computer Science at Auckland University have made my time as a post-graduate student both enjoyable and personally and professionally enriching.

I would like to thank my friends and family for all their support over the years. I particularly would like to thank my parents, Irene and Ray, for giving me so much love and support all my life and for the educational opportunities they never had. Thanks also go to my parents-in-law, Jannie and Mike Visser, my sisters and brother Heather, Jenny and Mike, and my sister-in-law Alice, for all their love and support.

The financial assistance of the Vice-Chancellors' Committee, IBM New Zealand Ltd, and the William Georgetti Scholarship Committee are gratefully acknowledged. I would also like to thank Professor Bob Doran for helping to ensure my financial well-being with work for the Department of Computer Science over the past three years.

The biggest thank you goes to my wife, Judy, for the incredible amount of love, caring, kindness and assistance she has given me. I can not begin to express the love and appreciation I have for her for being so supportive during what has been some very good and some quite trying times. I would also like to thank my daughter, Stephanie, for blessing Judy and I with such a precious gift as herself. This thesis is for you, Judy and Stephanie. Thank you so very much.

To Judy and Stephanie:

your love and support have made this work possible.

Contents

Multiple textual and graphical views for 1

Interactive Software Development Environments 1

Abstract i

Acknowledgments iii

Contents v

Chapter 1 17

Introduction 17

1.1. Rationale for Research 17

1.1.1. Textual vs. Graphical Program Construction and Representation 17

1.1.2. Integrated Textual and Graphical Programming 18

1.1.3. Integrated Software Development....19

1.2. Goals of Research....19

1.3. Contributions of Research 20

1.4. Thesis Organisation....21

Chapter 2 25

Related Research 25

2.1. Textual Programming 26

2.1.1. Unix-style Environments....26

2.1.2. Purpose-built, Tightly Integrated Environments 27

2.1.3. Tightly Integrated, Extensible Environments 27

2.1.4. Generated Environments 28

2.1.5. Text-based Software Development Environments....29

2.2. Graphical Programming and Visualisation 30

2.2.1. CASE Tools....30

2.2.2. Visual Programming....31

2.2.3. Program Visualisation 33

2.3. Integrated Software Development Environments 34

2.4. General Requirements....36

Chapter 3 39

Object-oriented Programming in Prolog with Snart 39

3.1. Object-oriented Programming....39

3.2. Rationale for Snart 40
3.3. A Snart Example: A Drawing Program 41
3.3.1. Snart Classes and Methods 42
3.3.2. Snart Objects 44
3.4. Classification in Snart 45
3.5. Object Tracing and Persistency 46
3.5.1. Object Tracing 47
3.5.2. Object Persistency 47
3.5.3. Implementation 48
3.6. Software Development in Snart 49
3.7. Other Object-Oriented Prologs 50
3.8. Future Research 51
3.9. Summary 52

Chapter 4 55

The Snart Programming Environment 55
4.1. Rationale for Snart Programming Environment 55
4.2. Analysis and Design of a Snart Program 57
4.2.1. Requirements for the Drawing Program 57
4.2.2. Creating a New Program 57
4.2.3. Analysis 58
4.2.4. Design 61
4.3. Implementing a Snart Program 62
4.4. Debugging a Snart Program 64
4.5. Modifying a Snart Program 64
4.5.1. Graphical Updates 65
4.5.2. Textual Updates 66
4.5.3. Update Histories 67
4.5.4. Integrated Software Development 68
4.6. Browsing a Snart Program 69
4.7. Managing a Snart Program's Complexity 71
4.7.1. Programmer-defined View and Icon Composition 71
4.7.2. Code Text Forms 72
4.7.3. Inheritance 73
4.8. Saving and Reloading a Snart Program 73
4.9. Discussion and Possible Extensions to SPE 74
4.9.1. Adequacy of Program Representation 74
4.9.2. Program Viewing and Construction Facilities 75
4.9.3. Large-scale Program Development 76
4.9.4. Programming Other Languages with SPE 77

4.10. Summary 77

Chapter 5 81

Modelling and Specifying Environments with MViews 81

5.1. Rationale For MViews 82
5.1.1. Program Representation 82
5.1.2. Multiple Textual and Graphical Views 82
5.1.3. Program and View Modification 82
5.1.4. Automatic, Efficient Consistency Management 83
5.1.5. Recording Previous Changes 83
5.1.6. Undo and redo of User Manipulations 83
5.1.7. Program and View Persistency and Multi-user Access 84
5.1.8. Tool Integration and Extensibility 84
5.2. Related Research 84
5.2.1. Smalltalk Model-View-Controller 84
5.2.2. Interviews and Unidraw 85
5.2.3. PECAN, GARDEN and FIELD 85
5.2.4. Grammar-based Environment Generators 86
5.2.5. Dannenburg's ItemList Structure 87
5.2.6. Wilk's Object Dependency Graphs 87
5.2.7. Summary 88
5.3. An Overview of MViews 89
5.3.1. Program Graphs 89
5.3.2. Views and View Components 91
5.3.3. Operations and Update Records 94
5.3.4. Program Editing using Views as Tools 97
5.3.5. Program Persistency 98
5.3.6. Summary 99
5.4. MViews Specification Language 99
5.4.1. Rationale for MVSL and MVisual 99
5.4.2. Overview of IspelM 101
5.4.3. Basic Abstractions of MViews 104
5.4.4. Defining Component Kinds with MVSL 105
5.4.5. Subset Views and Components 108
5.4.6. Operations 110
5.4.7. Update Operations 115
5.5. A Formal Specification of MVSL 118
5.5.1. Operational Semantics 118
5.5.2. Concrete vs. Abstract Syntax 118
5.5.3. Declaration Types 120

5.5.4. Building a Type Map for MVSL Program Declarations 121
5.5.5. State 123
5.5.6. Commands, Operations and Update Operations 125
5.5.7. Program Meaning 132
5.5.8. MVSL Programs 133
5.6. Specification of Visual Appearance and Semantics 135
5.6.1. Rationale for MVisual 135
5.6.2. MVisual Fundamentals 135
5.6.3. Icons and Glue 136
5.6.4. Views 139
5.6.5. Dialogues 141
5.7. Discussion and Future Research 143
5.7.1. Requirements Satisfaction 143
5.7.2. MVSL 147
5.7.3. Operational Specification of MVSL 148
5.7.4. MVisual 149
5.8. Summary 150

Chapter 6 153

An Object-Oriented Architecture for MViews 153
6.1. An Object-Oriented Architecture for MViews 153
6.2. Overview of the MViews Architecture 155
6.2.1. Components as Classes 155
6.2.2. Base Components 157
6.2.3. Subset and Display Views and Components 158
6.2.4. Subset/base Relationships 158
6.2.5. Additional Abstract Classes 159
6.2.6. User Interface and Persistency 160
6.2.7. Summary 160
6.3. Components 160
6.3.1. Component State 161
6.3.2. Update Records 162
6.3.3. Persistency 162
6.4. Base Program Components 163
6.4.1. View Components 163
6.4.2. Base Components 163
6.4.3. Viewable Base Components 164
6.4.4. Textual Base Components 164
6.5. Subset and Display Components 166
6.5.1. Subset Components 166

6.5.2. Display Components 167
6.5.3. Textual Display Components 168
6.5.4. Graphical Display Components 170
6.5.5. Icons 171
6.5.6. Glue 171
6.6. Relationships 171
6.6.1. Relationships 171
6.6.2. One-to-one 172
6.6.3. One-to-many 172
6.6.4. Many-to-many 172
6.6.5. Subset/Base Relationships 173
6.7. Views 175
6.7.1. Views 175
6.7.2. Base Views 176
6.7.3. Subset Views 176
6.7.4. Display Views 177
6.7.5. Textual Display Views 177
6.7.6. Graphical Display Views 178
6.7.7. View Composition and Layout 178
6.7.8. Application Framework 179
6.8. Operations and Update Records 179
6.8.1. Operations and Update Operations 179
6.8.2. Update Record Generation and Storage 180
6.8.3. Composite Update Records 181
6.8.4. Update Record Propagation 182
6.8.5. Constraints, Semantic Attribute Recalculation and Lazy Updates184
6.9. Discussion and Future Research 185
6.9.1. Components 185
6.9.2. Base Components 185
6.9.3. Subset and Display Components 186
6.9.4. Views 186
6.9.5. Relationships 187
6.9.6. Operations and Update Records 187
6.10. Summary 187

Chapter 7 191

An Object-Oriented Implementation of MViews 191
7.1. A Snart Framework 191
7.2. MViews Framework Complexity 193
7.3. Components 194

7.3.1. Component State ... 194
7.3.2. Update Records ... 194
7.4. Base Program Components ... 197
7.4.1. View and Base Components ... 197
7.4.2. Viewable Base Components ... 198
7.4.3. Textual Base Components ... 198
7.5. Subset and Display Components ... 198
7.5.1. Subset Components ... 199
7.5.2. Display Components ... 199
7.5.3. Graphical Display Components ... 200
7.5.4. Textual Display Components ... 200
7.6. Views ... 201
7.6.1. Base View ... 202
7.6.2. Subset Views ... 202
7.6.3. Display Views ... 203
7.6.4. Textual Display Views ... 203
7.6.5. Graphical Display Views ... 205
7.7. Relationships ... 205
7.7.1. Relationship Classes ... 205
7.7.2. Subset/base relationships ... 206
7.8. Operations and Updates ... 207
7.8.1. Update Record Generation and Storage ... 207
7.8.2. Update Record Propagation ... 207
7.8.3. Constraints and Semantic Calculation via Operations ... 208
7.8.4. Lazy Application of Update Records ... 208
7.9. User Interaction ... 208
7.10. Persistent Program Storage ... 209
7.10.1. Term Data Files ... 209
7.10.2. Component Persistency Methods ... 210
7.10.3. Snart Object Persistency ... 211
7.11. Discussion and Future Research ... 211
7.11.1. Component Class Implementation ... 211
7.11.2. Operations and Update Records ... 212
7.11.3. User Interaction ... 212
7.11.4. Persistent Program Storage ... 214
7.11.5. Implementation Language ... 215
7.12. Summary ... 216

Chapter 8 219

Architecture and Implementation of IspelM and SPE 219

8.1. IspelM Architecture....219
8.1.1. Overview of the IspelM Architecture....220
8.1.2. Base Clusters....221
8.1.3. Base Classes....221
8.1.4. Class Components....223
8.1.5. Programs....226
8.1.6. Subset and Display Views and Components....226
8.1.7. Subset/base relationships....228
8.2. IspelM Implementation....229
8.2.1. Snart Implementation of IspelM....230
8.2.2. Base Classes....230
8.2.3. Class Components....233
8.2.4. Programs....233
8.2.5. Subset and Display Views and Components....234
8.2.6. Subset/base relationships....236
8.2.7. Update Records....237
8.2.8. User Interaction....239
8.2.9. Program Persistency and Execution....240
8.3. The Snart Programming Environment....240
8.3.1. Parsing and Unparsing of Snart Programs....241
8.3.2. Validation, Compilation and Saving of Snart Programs....241
8.3.3. Running and Debugging Snart Programs....242
8.4. Discussion and Future Research....243
8.4.1. IspelM Model....243
8.4.2. IspelM Implementation....244
8.4.3. The Snart Programming Environment....246
8.5. Summary....246

Chapter 9 249

Further Applications of MViews 249
9.1. Entity-Relationship Modelling....250
9.1.1. Entity-Relationship and Relational Database Modelling....250
9.1.2. MViewsER....250
9.1.3. Specification....252
9.1.4. Design....253
9.1.5. Implementation....254
9.2. Dialogue Painting....255
9.2.1. User Interface Specification....255
9.2.2. MViewsDP....256
9.2.3. Specification....257

9.2.4. Design 259
9.2.5. Implementation 259
9.3. Program Visualisation 261
9.3.1. Tally Graph of Method Calls 262
9.3.2. Sorting Algorithm Animation 263
9.3.3. Visual Debugging 264
9.4. Other Applications of MViews 266
9.4.1. Facets and Object Persistency for ICAtect 266
9.4.2. Support for More OOA/OOD Notations 267
9.4.3. Dataflow Analysis Diagrams and Method Implementations 269
9.4.4. Tool-based Abstraction 272
9.5. Discussion and Future Research 273
9.5.1. MViewsER 273
9.5.2. MViewsDP 274
9.5.3. Program Visualisation 275
9.5.4. MViews 276
9.6. Summary 278

Chapter 10 281

Conclusions and Future Research 281
10.1. Research Contributions and Conclusions 281
10.1.1. Program and View Representation in MViews 281
10.1.2. Update Records for Change Propagation 282
10.1.3. View Editing and Tool Interfacing 283
10.1.4. MVSL and MVisual 284
10.1.5. MViews Architecture and Framework 285
10.1.6. SPE and IspelM 286
10.1.7. Reuse of MViews 286
10.1.8. Snart 287
10.2. Future Research 287
10.2.1. Abstract, Flexible Component Persistency 288
10.2.2. Tool Integration Issues 289
10.2.3. Version Control and Configuration Management Tools 290
10.2.4. Multi-user, Distributed Software Development 291
10.2.5. Lazy Update Processing and Attribute Recalculation 292
10.2.6. Partial Generation from Abstract Specification 294
10.2.7. IspelM and SPE Enhancements 294
10.2.8. Imperative Classification and Framework Specialisation 295
10.3. Summary 296

Glossary 298

Appendix A 305

LPA MacProlog Facilities 305

A.1. .... LPA

Graphics 305

A.2. .... LPA

Menus and Dialogs 306

A.3. .... LPA

File and Resource Management 308

Appendix B 309

The Snart Language 309

B.1. Syntax 309

B.1.1. Class Definitions 309

B.1.2. Method Predicates 310

B.1.3. Prolog Predicates 311

B.2. Objects 311

B.3. Environment 313

B.3.1. Search Menu 313

B.3.2. Eval Menu 315

B.3.3. Defining and Compiling Classes and Method Predicates 315

B.3.4. Debugging Snart Programs 316

B.3.5. Saving Snart Programs 316

B.4. Compiler 316

B.4.1. Class and Method Predicate Storage 317

B.4.2. The Compilation Process 318

B.4.3. Compiled Class Format 319

B.5. Run-time System 320

B.5.1. Object Creation, Attributes and Destruction 320

B.5.2. Method Calls 321

B.5.3. Other Object Predicates and Methods 321

B.5.4. Object Spying 322

B.5.5. Object Persistency 323

B.6. Performance 323

B.6.1. Speed 323

B.6.2. Memory Requirements 325

B.6.3. Programming in Snart vs. conventional LPA Prolog 325

B.7. Comparison to Other Object-Oriented Prologs 326

B.8. Quintus Snart 327

B.8.1. Compiler 327
B.8.2. Run-time System 327
B.8.3. Environment 328
B.9. Future Extensions 328
B.9.1. Explicit Redefinition and Information Hiding 328
B.9.2. Data-driven Support 328
B.9.3. Optimisations for Performance Enhancement 329
B.9.4. Typed Variables 330
B.9.5. Lazy, Functional Feature Evaluation 330

Appendix C 333

A Gofer Implementation of MVSL 333

Appendix D 353

An MVSL Specification of IspelM 353

Appendix E 369

An MVisual Specification for IspelM 369
E.1. IspelM Component Appearance 369
E.2. IspelM Component Interaction and Updates 370
E.3. MViews and IspelM Dialogs 374

Appendix F 379

A BNF Grammar for MVSL 379

References
ccclxxxiii