

UNIVERSITY OF CALIFORNIA,
IRVINE

Cognitive Support Features for Software Development Tools

DISSERTATION

submitted in partial satisfaction of the requirements for the degree of

DOCTOR OF PHILOSOPHY

in Information and Computer Science

by

Jason Elliot Robbins

Dissertation Committee:
Professor David F. Redmiles, Chair
Professor Debra J. Richardson
Professor David S. Rosenblum

1999

The dissertation of Jason Elliot Robbins is approved
and is acceptable in quality and form
for publication on microfilm:

Committee Chair

University of California, Irvine
1999

DEDICATION

To
my loving wife Sua-Yu
and my family
for their love and support

TABLE OF CONTENTS

	Page
CHAPTER 1: Introduction	1
1.1. Background on CASE tools	1
1.2. Research Method	2
1.3. Hypothesis and Contributions	5
1.4. Organization of the Dissertation	6
CHAPTER 2: Theories of Designers' Cognitive Needs	7
2.1. Theories of Design Decision-Making	8
2.1.1. Reflection-In-Action	8
2.1.2. Opportunistic Design	10
2.1.3. Geneplore	14
2.2. Theories of Human Memory	15
2.2.1. Associative Recall	15
2.2.2. Limited Short-Term Memory	17
2.2.3. Cognitive Fixation	19
2.2.4. Limited Knowledge	21
2.2.5. Mental Biases	22
2.3. Design Visualization Theories	23
2.3.1. Comprehension and Problem Solving	23
2.3.2. Secondary Notation	26
2.3.3. Viewing as an Acquired Skill	28
2.4. User Interface Guidelines	28
2.4.1. Style Guidelines and Usability Heuristics	28
2.4.2. Fitts' Law	31
CHAPTER 3: Previous Work in Cognitive Features for Design Tools	33
3.1. Previous Work on Design Critiquing Systems	33
3.1.1. Definitions of Design Critiquing Systems	33
3.1.2. Previous Work on Critiquing Processes	35
3.1.3. Phases of the ADAIR Process	36
3.1.4. Comparison of Critiquing Systems	39
3.1.5. State of the Art of Critiquing Systems	48
CHAPTER 4: Proposed Cognitive Features	50
4.1. Knowledge Support Features	54
4.1.1. Design Critics and Criticism Control Mechanisms	54
4.1.2. Non-modal Wizards	59
4.1.3. Context Sensitive Checklists	64
4.1.4. Design History	67
4.2. Process Support Features	71

4.2.1. Dynamic “To Do” List and Clarifiers	71
4.2.2. Opportunistic Search Utility	75
4.2.3. Opportunistic Table Views	78
4.3. Visualization Support Features	80
4.3.1. Navigational Perspectives	80
4.3.2. The Broom Alignment Tool	86
4.3.3. Model-based Layout	92
4.4. Construction Support Features	97
4.4.1. Selection-Action Buttons	97
4.4.2. Create Multiple	100
4.4.3. Visual Blender	108
 CHAPTER 5: Usage Scenario	 112
5.1. Scene 1: Initial Construction, Error Detection, and Correction	113
5.2. Scene 2: Cleaning up the Design to Communicate Intent	117
5.3. Scene 3: Answering Questions that Arise During Design	120
5.4. Scene 4: Considering the Important Issues	125
5.5. Scene 5: Resolving Open Issues Before Reaching a Milestone	128
5.6. Discussion	130
 CHAPTER 6: Heuristic Evaluation of Cognitive Features	 132
6.1. Walkthrough of “To Do” List and Clarifiers	135
6.2. Walkthrough of Non-modal Wizards	137
6.3. Walkthrough of Context Sensitive Checklists	140
6.4. Walkthrough of Design History	142
6.5. Walkthrough of Opportunistic Search Utility	143
6.6. Walkthrough of Opportunistic Table Views	146
6.7. Walkthrough of Navigational Perspectives	149
6.8. Walkthrough of Broom Alignment Tool	151
6.9. Walkthrough of Model-based Layout	154
6.10. Walkthrough of Selection-Action Buttons	157
6.11. Walkthrough of Create Multiple	159
6.12. Discussion and Validation	162
 CHAPTER 7: Empirical Evaluation of Cognitive Features	 168
7.1. Pilot User Study	168
7.2. Broom User Study	171
7.3. Construction User Study	174
7.4. Classroom Usage	176
7.5. Internet Usage	179
 CHAPTER 8: A Scalable, Reusable Infrastructure	 184
8.1. Graph Editing Framework	185

8.1.1. Introduction	185
8.1.2. Design Overview of GEF	186
8.1.3. Implementation of Multiple Diagrammatic Views	189
8.1.4. Implementation of the Broom Alignment Tool	190
8.1.5. Implementation of Selection-Action Buttons	192
8.2. Argo Kernel	192
8.2.1. Introduction	192
8.2.2. Design Overview of the Argo Kernel	193
8.2.3. Implementation of Design Critics and Criticism Control Mechanisms	196
8.2.4. Implementation of Checklists	199
8.2.5. Implementation of Wizards	200
8.3. Views and Navigation	201
8.3.1. Introduction	201
8.3.2. Design Overview of Argo/UML Views and Navigation	202
8.3.3. Implementation of Navigational Perspectives	203
8.3.4. Implementation of the Dynamic “To Do” List and Clarifiers	204
8.3.5. Implementation of Opportunistic Table Views	205
8.3.6. Implementation of Opportunistic Search	207
8.4. Design Representation and Code Generation	209
8.4.1. Introduction	209
8.4.2. Design Overview of Design Representation and Code Generation	209
8.4.3. Implementation of the UML Meta-Model	210
8.4.4. Implementation of XMI and PGML File Formats	212
8.4.5. Implementation of Code Generation	214
 CHAPTER 9: Conclusion	 217
9.1. Reflections on the Approach	217
9.2. Review of Contributions	219
9.3. Potential Extensions	220
 REFERENCES	 222

LIST OF FIGURES

	Page	
Figure 1-1.	Feature generation approach	3
Figure 2-1.	State diagram with alignment as secondary notation	27
Figure 3-1.	Phases of the ADAIR critiquing process	36
Figure 4-1.	Argo/C2: a design tool for C2-style architectures	50
Figure 4-2.	Prefer: a requirements tool using the CoRE notation	51
Figure 4-3.	Argo/UML: an OODA tool using the UML notation	52
Figure 4-4.	Decision model editor	57
Figure 4-5.	Critic browser window	57
Figure 4-6.	Proposed graphical specification of critics and wizards	59
Figure 4-7.	Context sensitive checklist	65
Figure 4-8.	Argo/UML's feedback item dismissal dialog	70
Figure 4-9.	Argo/UML's opportunistic search utility window	76
Figure 4-10.	Tabular view of state machine transitions	79
Figure 4-11.	(a) "Package-centric" navigational perspective, (b) "State-centric" navigational perspective, (c) "Transition-centric" navigational perspective	82
Figure 4-12.	Argo/UML's navigational perspective configuration window	83
Figure 4-13.	Aligning and distributing objects with the broom	88
Figure 4-14.	(a) Standard automated layout of a state diagram, (b) Model-based layout of a state diagram	94
Figure 4-15.	Configuring model-based layout with arbitrary constrained regions	96
Figure 4-16.	Selection-action buttons on a UML class, interface, and state	98
Figure 4-17.	Mock-up of window to create multiple elements by pattern name	102
Figure 4-18.	Mock-up for creating design fragments by form filling	103
Figure 4-19.	Mock-up of the visual blender window	109
Figure 5-1.	Argo/UML initial screen	113
Figure 5-2.	After placing initial classes	115
Figure 5-3.	"To do" item description	116
Figure 5-4.	Reorganized class diagram	118

Figure 5-5.	Class diagram with annotations describing data sources	119
Figure 5-6.	One class diagram of many after the design has grown	121
Figure 5-7.	Table view of associations and their properties	122
Figure 5-8.	Aggregate classes navigational perspective	124
Figure 5-9.	Mock-up of model-based layout	125
Figure 6-1.	A survey question on clarifiers	163
Figure 7-1.	Task for pilot study	169
Figure 7-2.	Desired groupings of diagram elements	172
Figure 7-3.	Mouse dragging with the broom or standard alignment tools	173
Figure 7-4.	(a) Conventional diagramming task used in selection-action button study, (b) Unconventional diagramming task	175
Figure 7-5.	Number of new Argo/UML registered users by month in 1999	181
Figure 8-1.	UML class diagram of GEF	187
Figure 8-2.	Broom states	191
Figure 8-3.	Classes implementing the Argo kernel	194
Figure 8-4.	CPU load imposed by critics on a 233MHz computer with Windows NT	197
Figure 8-5.	UML class diagram of Argo checklists	199
Figure 8-6.	Argo/UML main window	201
Figure 8-7.	Classes implementing Argo/UML's "to do" list	205
Figure 8-8.	Classes that implement Argo/UML's table views	206
Figure 8-9.	Classes implementing Argo/UML's opportunistic search utility	208
Figure 8-10.	Some UML meta-model classes	211
Figure 8-11.	Classes implementing XML file processing	215
Figure 8-12.	UML class diagram of classes for code generation.	216

LIST OF TABLES

	Page
Table 2-1: Usability guidelines from Mac Look and Feel (Apple 1993)	29
Table 2-2: Usability guidelines from Java Look and Feel (Sun, 1999)	29
Table 2-3: Usability guidelines from Nielsen (1995)	30
Table 2-4: Usability guidelines from Constantine and Lockwood (1999)	30
Table 2-5: Usability guidelines from Shneiderman (1998)	30
Table 3-1: Selected sefinitions of critiquing systems	33
Table 3-2: Summary comparison of critiquing systems	39
Table 4-1: Summary of proposed cognitive features	53
Table 4-2: Examples of critics in Argo/UML	55
Table 6-1: Initial doctrine for Argo/UML	133
Table 6-2: Steps for using a clarifier and the “to do” item tab	135
Table 6-3: Step for using the “to do” list and the “to do” tab	135
Table 6-4: Steps for non-modal wizards	138
Table 6-5: Steps for using context sensitive checklists	140
Table 6-6: Steps for using design history	142
Table 6-7: Steps for using the opportunistic search utility	144
Table 6-8: Steps for using opportunistic table views	146
Table 6-9: Steps for using navigational perspectives	149
Table 6-10: Steps for using the broom alignment tool	151
Table 6-11: Steps for using grid-based layout	154
Table 6-12: Steps for using region-based layout	154
Table 6-13: Steps for using selection-action buttons	157
Table 6-14: Steps for creating multiple design elements by pattern	159
Table 6-15: Steps for creating multiple design elements by form	160
Table 6-16: Questionnaire results for clarifiers, “to do” list, wizards, and checklists	164
Table 6-17: Questionnaire results for opportunistic search and table views	165

Table 6-18:	Questionnaire results for broom and selection-action buttons	166
Table 7-1:	Known classroom usage of the Argo family	178
Table 7-2:	Some quotes from Argo/UML users	182
Table 8-1:	Description of broom states	191
Table 8-2:	Some TEE templates for generating XMI files	213

ACKNOWLEDGEMENTS

This material is based upon work supported by the National Science Foundation under Grant No. CCR-9624846 and Grant No. CCR-9701973. Effort also sponsored by the Defense Advanced Research Projects Agency, Air Force Research Laboratory, Air Force Materiel Command, USAF under agreement numbers F30602-97-2-0021 and F30602-94-C-0218, and Air Force Office of Scientific Research under grant number F49620-98-1-0061. Additional support is provided by Rockwell International. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Defense Advanced Research Projects Agency, Air Force Research Laboratory or the U.S. Government.

CURRICULUM VITAE

Jason Elliot Robbins

- 1988-91, 1994-97 Summer Student, and Flex-Force Engineer
Rockwell International Science Center, Thousand Oaks, CA
- 1992 B.S. in Computer Science, Cum Laude
University of California, Los Angeles
Major GPA: 3.9
- 1993 Senior Coder, CoBase Reseach Group
University of California, Los Angeles
Computer Science Dept.
- 1994 - 95 Teaching Assistant
ICS 52 - Systematic Software Construction
ICS 125 - Project in System Design
ICS 141 - Compilers and Interpreters
University of California, Irvine
Dept. Information and Computer Science
- 1995 M.S. in Information and Computer Science
University of California, Irvine
Major Emphasis: Software
Cumulative GPA: 4.00
- 1995 - present Graduate Student Researcher, Software Research Group
University of California, Irvine
Dept. Information and Computer Science
- 1999 Ph.D. in Information and Computer Science
University of California, Irvine
Software Research Group
Dissertation: "Cognitive Support Features for Software
Development Tools"
Advisor: Dr. David F. Redmiles

PUBLICATIONS

Journal Publications

A Component- and Message-Based Architectural Style for GUI Software. Richard N. Taylor, Nenad Medvidovic, Kenneth M. Anderson, E. James Whitehead, Jr., Jason E. Robbins, Kari A. Nies, Peyman Oreizy, and Deborah L. Dubrow. *IEEE Transactions on Software Engineering*. vol. 22. no. 6. June 1996. pp. 390-406. A significant revision and extension of the ICSE'95 paper.

Extending Design Environments to Software Architecture Design. Jason E. Robbins, David M. Hilbert, David F. Redmiles. *International Journal of Automated Software Engineering*. Special issue: The Best of KBSE'96. vol. 5. 1998. pp. 261-290. A significant revision and extension of the KBSE'96 paper.

Software Architecture Critics in the Argo Design Environment. Jason E. Robbins, David F. Redmiles. *Knowledge-Based Systems*. Special issue: The Best of IUI'98. In press. A significant revision and extension of the IUI'98 paper.

Refereed Conference Publications

Reusable Objects. David Morley, Stephen Chiu, Jason Robbins, Tim Maddux, Geoffrey Voelker. *Technology of Object-Oriented Languages and Systems (TOOLS'92)*. Paris, France. March, 1992.

A Component and Message-Based Architectural Style for GUI Software. Richard N. Taylor, Nenad Medvidovic, Ken M. Anderson, E. James Whitehead, Jr., and Jason E. Robbins. *International Conference on Software Engineering 1995 (ICSE'95)*. Seattle WA. April 23-30, 1995. pp. 295-304. This paper was named one of the best papers of the conference.

Software Architecture Design from the Perspective of Human Cognitive Needs. Jason E. Robbins and David F. Redmiles. *Proceedings of the California Software Symposium 1996*. Los Angeles, California. April 17, 1997. pp. 16-27.

Visual Language Features Supporting Human-Human and Human-Computer Communication. Jason E. Robbins, David J. Morley, David F. Redmiles, Vadim Filatov, Dima Kononov. *IEEE Symposium on Visual Languages 1996 (VL'96)*. Boulder, CO. Sept. 3-6, 1996. pp. 247-254.

Extending Design Environments to Software Architecture Design. Jason E. Robbins, David M. Hilbert, and David F. Redmiles. *Knowledge-Based Software Engineering 1996 (KBSE'96)*. Syracuse, NY. Sept. 25-28, 1996. pp. 63-72. This paper was selected as best of conference.

Using Object-Oriented Typing to Support Architectural Design in the C2 Style. Nenad Medvidovic, Peyman Oreizy, Jason E. Robbins, and Richard N. Taylor.

Proceedings of SIGSOFT'96: The Fourth Symposium on the Foundations of Software Engineering (FSE-4). San Francisco, CA. October 16-18, 1996. pp. 24-32.

Argo: A Design Environment for Evolving Software Architectures. Jason E. Robbins, David M. Hilbert, David F. Redmiles. *Proceedings of the 19th International Conference on Software Engineering (ICSE'97)*. Boston, MA. May 17-23, 1997. pp. 600-601.

Software Architecture Critics in Argo. Jason E. Robbins, David M. Hilbert, David F. Redmiles. *Proceedings of the 1998 International Conference on Intelligent User Interfaces (IUI'98)*. San Francisco, CA. Jan. 6-9, 1998. pp. 141-144. This paper was named one of best papers of the conference.

EDEM: Intelligent Agents for Collecting Usage Data and Increasing User Involvement in Development. David M Hilbert, Jason E. Robbins, and David F. Redmiles. *Proceedings of the 1998 International Conference on Intelligent User Interfaces (IUI'98)*. San Francisco, CA. Jan. 6-9, 1998. pp. 73-76.

Modeling C2 in the Unified Modeling Language. Jason E. Robbins, David F. Redmiles, David S. Rosenblum. *Proceedings of the California Software Symposium 1997*. Irvine, CA. Nov. 7, 1997. pp. 11-18.

Integrating Architecture Description Languages with a Standard Design Method. Jason E. Robbins, Nenad Medvidovic, David F. Redmiles, David S. Rosenblum. *Proceedings of the 1998 International Conference on Software Engineering (ICSE'98)*. Kyoto, Japan. April 19-25, 1998. pp. 209-218.

Workshop Publications

Software Architecture: Foundations of a Software Component Marketplace. E. James Whitehead, Jr., Jason E. Robbins, Nenad Medvidovic, and Richard N. Taylor. *ICSE 1995 Workshop on Software Architecture*. Seattle, WA.

Using Critics to Analyze Evolving Architectures. Jason E. Robbins, David M. Hilbert, and David F. Redmiles. *Second International Software Architecture Workshop (ISAW-2)*. Held in conjunction with FSE'96. San Francisco, CA. October 16-18, 1996. pp. 90-93.

Technical Reports

A Software Architecture Design Environment for Chiron-2 Style Architectures. Jason E. Robbins, E. James Whitehead, Jr., Nenad Medvidovic, and Richard N. Taylor. January 1995. Tech Report Arcadia-UCI-95-01.

Using Critics to Support Software Architects. Jason E. Robbins, David M. Hilbert, David F. Redmiles. April 1997. Technical Report UCI-ICS-97-18.

Supporting Ongoing User Involvement in Development via Expectation Driven Event Monitoring. David M. Hilbert, Jason E. Robbins, David F. Redmiles. April 1997. Technical Report UCI-ICS-97-19.

Integrating Architecture Description Languages with a Standard Design Method.

Jason E. Robbins, Nenad Medvidovic, David F. Redmiles, David S. Rosenblum. August 1997. Technical Report UCI-ICS-97-35.

FORMAL PRESENTATIONS

Software Architecture Design from the Perspective of Human Cognitive Needs. California Software Symposium 1996. Los Angeles, CA. April 17, 1997.

Visual Language Features Supporting Human-Human and Human-Computer Communication. IEEE Symposium on Visual Languages 1996 (VL'96). Boulder, CO. Sept. 3-6, 1996.

Extending Design Environments to Software Architecture Design. Knowledge-Based Software Engineering 1996 (KBSE'96). Syracuse, NY. Sept. 25-28, 1996.

Argo: A Tool for Evolving Software Architectures. Nineteenth International Conference on Software Engineering (ICSE'97). Boston, MA. May 17-23, 1997.

Software Architecture Critics in Argo. 1998 International Conference on Intelligent User Interfaces (IUI'98). San Francisco, CA. Jan. 6-9, 1998.

Extending Design Environments to Software Architecture Design. Presented at Jet propulsion Laboratories. Pasadena, CA. June 1997.

Modeling C2 in the Unified Modeling Language. California Software Symposium 1997. Irvine, CA. Nov. 7, 1997.

Using Critics to Analyze Evolving Architectures. Second International Software Architecture Workshop (ISAW-2). Held in conjunction with FSE'96. San Francisco, CA. October 16-18, 1996.

Integrating Architecture Description Languages with a Standard Design Method. EDCS Cross Cluster Workshop. Austin, TX. Nov. 10-12. 1997.

RESEARCH PROPOSALS

Applying Design Critics to Software Requirements Engineering. (PI: D. F. Redmiles).

Submitted to the Microelectronics Innovation and Computer Research Opportunities (MICRO) program. Funded with a one-year budget of \$33,152 (includes \$20,000 in matching funds from Rockwell). August, 1997.

Open Technology for Software Evolution: Hyperware, Architecture, and Process. (PI's: R. N.

Taylor and D. F. Redmiles). Submitted to the Defense Advanced Research Projects Agency (DARPA). Funded with a three-year budget of \$2,606,666. October, 1996.

PROFESSIONAL ASSOCIATIONS

Association for Computing Machinery (ACM)
IEEE Computer Society
ACM Special Interest Group on Software Engineering (SIGSOFT)
ACM Special Interest Group on Computer Human Interaction (SIGCHI)

HONORS, AWARDS, FELLOWSHIPS

Thomas J. Watson Memorial Scholar
National Merit Society Finalist
Twice elected President of UCLA's Computer Science Undergraduate
Association (UCLA's student chapter of the Association for Computing Machinery)
Tau Beta Pi (the national engineering honor society)
UCLA School of Engineering Dean's Honor List
UCLA School of Engineering Dean's Circle
MICRO Fellowship

ABSTRACT OF THE DISSERTATION

Cognitive Support Features for Software Development Tools

By

Jason Elliot Robbins

Doctor of Philosophy in Information and Computer Science

University of California, Irvine, 1999

Professor David F. Redmiles, Chair

Software design is a cognitively challenging task. Most software design tools provide support for editing, viewing, storing, sharing, and transforming designs, but lack support for the essential and difficult cognitive tasks facing designers. These cognitive tasks include decision making, decision ordering, and task-specific design understanding. To date, software design tools have not included features that specifically address key cognitive needs of designers, in part, because there has been no practical method for developing and evaluating these features.

This dissertation contributes a practical description of several cognitive theories relevant to software design, a method for devising cognitive support features based on these theories, a basket of cognitive support features that are demonstrated in the context of a usable software design tool called Argo/UML, and a reusable infrastructure for building similar features into other design tools. Argo/UML is an object-oriented design tool that includes several novel features that address the identified cognitive needs of software designers. Each feature is explained with respect to the cognitive theories that inspired it and the set of features is evaluated with a combination of heuristic and empirical techniques.