

Copyright is owned by the Author of the thesis. Permission is given for a copy to be downloaded by an individual for the purpose of research and private study only. The thesis may not be reproduced elsewhere without the permission of the Author.

Methodology Independent CASE Tool

A Prototype.

A thesis presented in partial fulfilment of
the requirements for the degree
of Master of Arts in Computer Science at
Massey University

Paul Clark
1994

005.11
Cla
· ~~120~~

Abstract

The object-oriented (OO) movement is at present split into many factions and as a result no standard has been defined. A direct consequence of this lack of consensus is that there are no mature CASE tools available. Current object-oriented CASE tools are methodology dependent. They are not driven by the need to enable the successful construction of OO software but rather to sell a particular methodology (at this time there are more than 30 different OO methodologies. Forcing the developer to use a particular methodology constrains his/her ability to select the most appropriate problem representation. To address these problems a research project aiming at the development of a methodology independent OO CASE tool has commenced. The thesis is the first stage in the tool development. It addresses two main problem areas of this research project:

- The development of a customisable user interface which utilises an abstract notation definition language.
- Support of the implementation phase of OO software.

A language which facilitates abstract definitions of graphical notations and the human computer interaction with them has been developed. The implemented graphical user interface uses the developed language to allow arbitrary notations and dialogues for OO models to be described and modified without recompilation of the CASE Tool.

Re-engineering facilities have been designed to allow a user to generate an OO model from existing OO source code. Automatic layout generation has been investigated and several auto-routing algorithms applied. The current tool generates Coad and Yourdon diagrams from C++ source code. The user can update and navigate the C++ source via the created OO model.

Based on the thesis results two papers were presented at the International conference on Object Oriented Information Systems in London, December 1994. A collaborative research has also commenced with the University of Technology, Sydney.

Acknowledgments

I would like to thank

Stephanie first because she wanted to be at the top of the list.

Daniela Mehandjiska-Stavreva for being my supervisor.

The following people for checking my work:

Shamus Smith

David Page

Daniela Mehandjiska-Stavreva

Sarah Stock

Wolfgang Tietz for lending me some OO literature.

Table of Contents

Abstract	iii
Acknowledgments.....	v
Table of Contents	vi
Introduction	1
1.1 What is Object Oriented ?	3
1.1.1 Benefits	5
1.1.2 Warnings	6
1.1.3 The Future	6
2. Problem Definition	9
2.1 CASE Tools.....	9
2.1.1 History.....	9
2.1.2 Deficiencies	9
2.1.3 Problem Area to be Addressed	10
2.1.4 Existing CASE Tools.....	11
2.2 Methodology Independent CASE Tool Structure.....	13
2.2.1 Tool Management System.....	13
2.2.2 Graphical User Interface	13
2.2.3 Persistent Storage	14
2.2.4 Configuration Manager.....	16
2.2.4.1 Tool Inference	16
2.4 Selection of Tools	22
3. Notation Definition Language	23
3.1 Template Language	23
3.1.1 Language Structure.....	23
3.2 Notation Definition.....	24
3.2.1 Group Templates	25
3.2.1.1 Equations.....	25
3.2.1.2 Segments	28
3.2.1.3 Future Enhancements	33
3.2.2 Template Object.....	36
3.2.2.1 Regions.....	36
3.2.2.2 Redefining IDs.....	37
3.2.2.3 Future Enhancements	37
3.2.3 Template Connections	40
3.2.3.1 Future Enhancements	44
4. Graphical User Interface	47
4.1 Interface Basics	48
4.2 Dialog Control.....	49
4.2.1 The Main Window.....	50
4.2.1.1 Menu bar	51
4.2.1.2 Model Browser	51
4.2.1.3 Function and Status bar	52
4.2.1.4 Canvas.....	53
4.2.2 Dialogs for Creation	53
4.2.3 Acquisition Dialogs	54
4.2.4 Dialogs for Region Functionality	56

4.3.4.1 Future Expansion.....	59
4.2.5 Errors.....	59
4.3 Graphic Visualisation.....	59
4.4 Object Manipulation.....	60
4.4.1 Graphic Creation.....	60
4.4.2 Graphic Movement.....	60
5. Re-Engineering	61
5.1 Re-Engineering for the CASE Tool.....	62
5.2 Language Parsing.....	63
5.2.1 The structure and composition of C++.....	63
5.2.2 Internal Representation	66
5.2.3 Class Parsing.....	68
5.2.3.1 Class Header interpretation	68
5.2.3.2 Method Interpretation.....	68
5.2.3.3 Attribute Interpretation.....	70
6. Diagrammatic Representation.....	71
6.1 Introduction	71
6.2 Diagrammatic Representation Types	72
7. Hierarchical Graphs.....	77
7.1 Multi-layer Graphs.....	81
7.2 Connections that descend more than one level.....	86
7.3 Tidying Up.....	87
8. Routing Non-Hierarchical Graph Connections.....	89
8.1 Positioning Hierarchies.....	89
8.2 Connection Generation and Routing.....	99
9. Diagramming Deficiencies.....	109
9.1 Future Enhancements.....	109
9.1.3 Saving Template.....	110
10. Conclusion	111
Appendix A	113
The Initial Object Model	115
Appendix B.....	117
Template Definition Grammar	119
Appendix C	123
Tokens	125
Appendix D	127
Steps for Padding Nodes in a Hierarchical Graph.....	129
Steps for Positioning Hierarchies in a Non-Hierarchical Graph	130
Connection Generation and Routing Steps for a Non-Hierarchical Graph.....	134
Solution for Preventing Routing Conflicts in Non-Hierarchical Graphs	135
Appendix E.....	137
C++ Class Specification Grammar	139

Appendix F	141
Lexical Analysis	143
F.1 Character Retrieval	144
F.2 Token Retrieval.....	144
F.2.1 Token Recognition.....	144
F.2.1.1 Template Definition Language	144
F.2.1.2 C++ Language	144
F.2.2 Comments	145
F.2.3 Token Look Ahead	145
F.3 Expression Retrieval	145
F.4 Error Handling.....	147

References	149
-------------------------	------------