

The Influence of Software Reuse on Programming Language Design

DISSERTATION

Presented in Partial Fulfillment of the Requirements for
the Degree Doctor of Philosophy in the Graduate
School of the Ohio State University

By

Douglas Eugene Harms, B.S., M.S.

* * * * *

The Ohio State University

1990

Dissertation Committee:

Dr. Bruce W. Weide

Dr. Neelam Soundararajan

Dr. Timothy J. Long

Approved by

Adviser

Department of
Computer and Information Science

To my parents, Clarence and Mary Ann Harms,
my wife, Mary Beth,
and daughters, Rebecca Elizabeth, Gretchen Marie,
and Alisha Anne

ACKNOWLEDGEMENTS

I express sincere appreciation to my adviser, Bruce Weide, for his patience and guidance during this research, and for his confidence. I also thank the other members of my committee, Timothy Long and Neelam Soundararajan, for finding time in their busy schedules to make constructive suggestions for both the research and dissertation organization.

I am grateful to the members of the Reusable Software Research Group at Ohio State, especially Bill Ogden, Stu Zweben, Mike Stovsky, Murali, Joan Krone, and Joe Hollingsworth. This research has been positively influenced and directed by the ideas and suggestions expressed during our many discussions.

I am indebted to the Muskingum College community for supporting me in both financial and moral senses during the years it has taken me to achieve this goal. I especially thank Dan Van Tassel, Sam Speck, Jim Smith, Ralph Hollingsworth, Ray Rataiczak, and Art DeJong.

I am also thankful for the encouragement and support provided by the members of the United Methodist Church in New Concord and St. Andrew Presbyterian Church in Columbus.

Finally, and most importantly, I am grateful to my wife, Mary Beth, and children, Rebecca, Gretchen, and Alisha, who made this adventure possible with their understanding and constant encouragement.

VITA

June 10, 1957.....	Born — Hillsboro, Kansas
1979.....	Bachelor of Science Muskingum College New Concord, Ohio
1979-1981.....	System Programmer NCR Corporation Cambridge, Ohio
1981-1983.....	Lecturer Muskingum College New Concord, Ohio
1983.....	Master of Science The Ohio State University Columbus, Ohio
1983-present	Assistant Professor Muskingum College New Concord, Ohio

PUBLICATIONS

Types, Copying, and Swapping: Their Influences on the Design of Reusable Software Components, with B.W. Weide, Department of Computer and Information Science, The Ohio State University, Columbus, OH, March 1989, OSU-CISRC-3/89-TR13.

Efficient Initialization and Finalization of Data Structures: Why and How, with B.W. Weide, Department of Computer and Information Science, The Ohio State University, Columbus, OH, March 1989, OSU-CISRC-3/89-TR11.

“Deadlock-Avoidance Mechanisms in Distributed Systems”, with A. Datta, S. Ghosh, and A. Elmagarmid, *Computer Systems Science and Engineering*, Vol. 3, No. 2 (April 1988), pp. 67-82.

Swapping — A Desirable Alternative to Copying, with B.W. Weide, Department of Computer and Information Science, The Ohio State University, Columbus, OH, January 1988, OSU-CISRC-1/88-TR2.

“Deadlock Avoidance in Real-Time Resource Sharing Distributed Systems: An Approach Using Petri Nets”, with A. Datta and S. Ghosh, *Proceedings of the Real-Time Systems Symposium*, December 1984, pp. 49-61.

FIELDS OF STUDY

Major Field: Computer and Information Science — Software Engineering
Minor Fields: Programming Language Design, Compiler Construction Techniques,
Computer Architecture, Formal Program Specification and Verification, and
Theoretical Computer Science

TABLE OF CONTENTS

ACKNOWLEDGEMENTS	iii
VITA	iv
LIST OF FIGURES.....	x
CHAPTER I Introduction	1
CHAPTER II Definitions and Framework	5
2.1 Definitions	5
2.2 Reusability Issues.....	7
2.2.1 What Is A “Reusable Software Component?”.....	7
2.2.2 Advantages of Software Reuse	8
2.2.3 Non-Technical Impediments to Software Reuse.....	9
2.2.4 Technical Impediments to Software Reuse	10
2.3 Characteristics of Reusable Parts.....	12
2.3.1 Formal Specification	12
2.3.2 Separation of Specification from Implementation.....	15
2.3.3 Generics	16
2.3.4 Multiple Implementations.....	16
2.3.5 Efficient Implementations Possible.....	18
2.4 Reusable Part Design Issues.....	20
2.4.1 Encapsulation	20
2.4.1.1 Abstract Data Objects.....	21
2.4.1.2 Abstract Data Types	24
2.4.1.3 Primary and Secondary Operations.....	26
2.4.2 Specification.....	26
2.4.2.1 Mathematical Theories.....	27
2.4.2.2 Algebraic Specification.....	29
2.4.2.3 Model-Based Specification	31
2.5 Programming Language Design Issues.....	32

2.5.1	Pointers	33
2.5.2	Data Movement Primitives.....	35
2.5.3	Types and Variables.....	37
2.5.3.1	Meaning of Types and Variables.....	37
2.5.3.2	Role of Types	42
2.5.3.3	Type Equivalence	43
2.5.3.4	Type Coercion.....	44
2.5.3.5	Dynamic vs. Static Typing.....	45
2.5.3.6	Type Initialization and Finalization	46
2.5.3.7	Built-In Types.....	49
2.6	Programming Language Survey.....	51
2.6.1	Ada and Anna	51
2.6.2	Modula-2	54
2.6.3	Euclid	56
2.6.4	Gypsy.....	58
2.6.5	Alphard	59
2.6.6	C++	62
2.6.7	Eiffel	66
2.6.8	CLU and Larch/CLU.....	70
2.6.9	Z.....	72
2.7	Summary.....	73
CHAPTER III RESOLVE.....		75
3.1	Conceptualizations and Type Parameters.....	76
3.1.1	LIFO Stacks and Conceptualization Stack_Template.....	76
3.1.2	Generic Conceptualizations and Type Parameters	79
3.1.3	Mathematical Theory Modules	80
3.1.4	Specification of Types	81
3.1.5	RESOLVE Operations	81
3.1.6	Parameter Modes	83
3.1.7	Operation Specification	84
3.1.8	Another Example: Conceptualization One_Way_List_Template	86
3.1.9	Summary	90
3.2	Simple Realizations	90
3.2.1	Realization Stack_Real_1 of Stack_Template	90

3.2.2	Conceptualization Auxiliary Section.....	93
3.2.3	Realization Auxiliary Section.....	93
3.2.4	Interface Section	95
3.2.4.1	Type Representations.....	95
3.2.4.2	Operation Implementations	97
3.2.5	Summary	97
3.3	Data Movement and Control Structures.....	98
3.3.1	Swapping — RESOLVE’s Data Movement Primitive	98
3.3.1.1	The Swap Statement.....	99
3.3.1.2	Swapping Is Efficient	99
3.3.1.3	Operation Invocation and Parameter Passing.....	101
3.3.1.4	The Function Assignment Statement	104
3.3.1.5	Copying a Variable	104
3.3.2	Ifs, Whiles, and Control Invocations	105
3.3.3	Return Statements	107
3.3.4	Summary	108
3.4	Types and Type Equivalence	108
3.4.1	Domains	109
3.4.2	Math Types.....	111
3.4.3	Program Types and Markers	114
3.4.4	Type Equivalence.....	117
3.4.5	Influence on Compiler Implementation	118
3.4.6	Summary	119
3.5	Conceptual Facility Parameters.....	120
3.5.1	Conceptualization Bounded_Stack_Template	121
3.5.2	Conceptualization Copy_Stack_Template	125
3.5.3	Conceptualization Array_Template	128
3.5.4	Summary	131
3.6	Conceptual Constants and Variables.....	131
3.6.1	Conceptualization Bounded_Integer_Template	131
3.6.2	Conceptualization Single_Link_Ref_Template.....	134
3.6.3	Conceptualization ADO_Stack_Template.....	139
3.6.4	Summary	141
3.7	Realization Parameters, Constants, and Variables	141

3.7.1	Realization Copy_Stack_Real_1 of Copy_Stack_Template	142
3.7.2	Realization List_Real_1 of One_Way_List_Template	144
3.7.3	Other Realization Sections	151
3.7.4	Summary	152
3.8	Implementation Issues	152
3.8.1	Primitive Realizations for Conceptualizations	152
3.8.2	Lazy Initialization of Variables	153
3.8.3	Efficient Implementation of Generic Modules	154
3.9	Summary	154
CHAPTER IV	Interaction of Programming Language and Environment Design	157
4.1	Programming Language Influence on Environmental Design	157
4.2	Environmental Influence on Programming Language Design	159
4.3	Summary	163
CHAPTER V	Conclusion	164
5.1	Summary and Conclusions	164
5.1.1	Software Reusability	164
5.1.2	RESOLVE Programming Language	166
5.1.3	Interaction of Programming Language and Environment Design	169
5.2	Future Work	170
5.2.1	Enhancements	170
5.2.2	Accessors	172
5.2.3	Synoptic Comments	172
5.3	Contributions	173
APPENDIX A	An Editing Environment for RESOLVE	175
A.1	<VAR NAME> Placeholders and Variables	177
A.2	Inserting Statements and Control Invocations	179
A.3	Variable Declaration	183
A.4	Inserting Function Invocations	186
A.5	If and Return Statements	186
A.6	Editing Assertions	187
A.7	Creating Conceptualizations, Types, and Operations	188
A.8	Creating Realizations	191
A.9	Selection and Deletion	192
BIBLIOGRAPHY	195

LIST OF FIGURES

Figure 1	Informal Description of a Template Providing Type Stack.....	19
Figure 2	Specification for a Module Providing the Generic Type Stack	77
Figure 3	Specification of Function Top	82
Figure 4	Specification for a Module Providing the Generic Type List	87
Figure 5	Realization Stack_Real_1 of Stack_Template Using One_Way_List_Template	91
Figure 6	Abstract Effect of Swap Statement “x :=: y”	100
Figure 7	Implementation of Swap Statement “x :=: y”	101
Figure 8	Definition and Invocation of Sample Procedure.....	102
Figure 9	Effect of Sample Procedure Invocation.....	103
Figure 10	Specification of Function Replica for Type T.....	105
Figure 11	Relationship Between types, Markers, and Domains.....	110
Figure 12	Example Type Declarations	112
Figure 13	Type Mappings for Example Type Declarations.....	113
Figure 14	Type Mappings for Realization Stack_Real_1	116
Figure 15	Specification for a Module Providing Generic Type Bounded_Stack	122
Figure 16	Bounded_Stack_Template Client.....	124
Figure 17	Program Type Mappings for Bounded_Stack_Template Client	125
Figure 18	Specification for a Module Providing Procedure Copy_Stack.....	126
Figure 19	Copy_Stack_Template Client.....	127
Figure 20	Specification for a Module Providing Generic Type Array	128
Figure 21	Specification for a Module Providing Type Int.....	132
Figure 22	Specification for a Module Providing Generic Type Reference	135
Figure 23	Specification for a Module Providing an “Object-Oriented” Stack	140
Figure 24	Realization Copy_Stack_Real_1 of Copy_Stack_Template.....	142
Figure 25	Realization List_Real_1 of One_Way_List_Template	144
Figure 26	Effect of Alternate Syntax.....	161

Figure 27	RESOLVE Editor Screen Snapshot.....	176
Figure 28	RESOLVE Editor Menus.....	177
Figure 29	Replacing An Untyped <VAR NAME> Placeholder	178
Figure 30	Replacing a Typed <VAR NAME> Placeholder	178
Figure 31	Changing a Variable	179
Figure 32	Insertion Arrow Positions.....	180
Figure 33	Inserting a While Statement Into Copy_Stack.....	181
Figure 34	Inserting an Invocation of Procedure Pop.....	182
Figure 35	Inserting an Invocation of Control Is_Empty.....	182
Figure 36	Type Declaration Menu.....	184
Figure 37	Dialog Box to Name a Variable.....	184
Figure 38	Menu for In-Place Variable Declaration	185
Figure 39	Dialog Box for In-Place Variable Declaration.....	185
Figure 40	Replacing a Typed <FUNCTION CALL> Placeholder	186
Figure 41	Inserting an If...Then...Else Statement	187
Figure 42	Inserting a Return Yes Statement.....	187
Figure 43	Editing an Assertion	188
Figure 44	Newly Created Conceptualization.....	189
Figure 45	Replacing a <CONCEPTUAL NAME> Placeholder in a Facility Declaration	189
Figure 46	Inserting a Function Declaration.....	190
Figure 47	Inserting a Formal Parameter Declaration	190
Figure 48	Dialog Box for Naming a Formal Parameter.....	191
Figure 49	Newly Created Realization.....	191
Figure 50	Interface Section of Realization Stack_Real_1.....	192
Figure 51	Selecting a Statement	193
Figure 52	Selecting Multiple Statements By Dragging.....	193