

To my parents, Martha and Charles Hollingsworth,
my wife, Janet Vician Hollingsworth,
and children, Emma Jean and Max Wilson.

ACKNOWLEDGMENTS

Sincere thanks goes to Bruce Weide, my adviser. Without your guidance, patience and seemingly limitless interest in my research, I would have made too many “wrong turns” and not enough “right turns.” I also wish to thank Stuart Zweben and Neelam Soundararajan, the other members of my reading committee, for their suggested clarifications and constructive comments. Your comments and suggestions helped to make this dissertation be a more complete document.

I am also grateful to current and former members of the Reusable Software Research Group at Ohio State. Thanks goes to Bill Ogden, Stephen Edwards, Doug Harms, Wayne Heym, and Murali Sitaraman. It has been a pleasure to work with you along the way. Special thanks goes to Mike Stovsky for helping me navigate through this last requirement for a Ph.D.

Finally, I am indebted to Janet, my wife, for allowing me to take on this endeavor. Furthermore, without your encouragement and backing, I would not have been able to finish what I started. I am also grateful to Emma and Max, my children, for their patience and understanding, especially while I worked endless hours on my “black book.”

VITA

November 7, 1958 Born — Kokomo, Indiana

1982..... Bachelor of Science
Indiana University
Bloomington, Indiana

1984..... Master of Science
Purdue University
West Lafayette, Indiana

1984 - 1986 Software Design Engineer
Texas Instruments
Dallas, Texas

1986 - Present..... Research Assistant /
Teaching Assistant
The Ohio State University
Columbus, Ohio

1987 - Present..... Owner
Hollingsworth Solutions
Columbus, Ohio

1991 - Present..... Principal Research Scientist
Battelle Memorial Institute
Columbus, Ohio

PUBLICATIONS

Hollingsworth, J.E. and Weide, B.W., "Engineering 'Unbounded' Reusable Ada Generics," *Proceedings of 10th Annual National Conference on Ada Technology*, Arlington, VA, February 1992, pp. 82 - 97.

Hollingsworth, J.E., Weide, B.W., and Zweben, S.H., "Confessions of Some Used-Program Clients," *Proceedings of 4th Annual Workshop on Software Reuse*, Herndon, VA, November 1991.

Hollingsworth, J.E., Weide, B.W., and Zweben, S.H., "Abstraction Leaks in Ada," *Proceedings of 14th Minnowbrook Workshop on Software Engineering*, Blue Mountain Lake, NY, July 1991.

Hollingsworth, J.E., "Toward Formalizing Control System Simulation Software," *Proceedings of a Symposium to Honor Samuel D. Conte*, West Lafayette, IN, November 1989.

FIELDS OF STUDY

Major Field: Computer and Information Science

Minor Fields: Software Engineering, Artificial Intelligence, Semantics of Distributed Computing, Compiler Construction Techniques, and Programming Language Design

TABLE OF CONTENTS

ACKNOWLEDGMENTS.....	iii
VITA.....	iv
LIST OF FIGURES	xi
LIST OF TABLES	xiii
CHAPTER I Introduction.....	1
1.1 The Thesis	2
1.2 Definition of a Software Discipline	2
1.3 A Discipline for Constructing High-Quality Components	4
1.3.1 The Discipline's Goal and Properties.....	4
1.3.2 The Discipline's Principles.....	5
1.3.3 Requiring the Discipline to Be Practical	5
1.3.4 Related Work	9
1.4 Outline of Dissertation.....	10
CHAPTER II First Principles for Constructing Components in Ada.....	11
2.1 Component Interface Principles.....	11
2.1.1 The Unit of Modularity.....	12
2.1.2 Location of Abstract State.....	15
2.1.3 Variable Finalization	17
2.1.4 Variable Initialization	22
2.1.5 Testing Preconditions.....	25
2.1.6 Limited Private Types	28
2.1.7 Data Movement Operation.....	33
2.1.8 Operations as Procedures	35
2.1.9 Package Finalization	40
2.1.10 Package Initialization	41
2.1.11 Conceptual Type Parameters.....	45
2.1.12 Conceptual Type's Standard Operations	46

2.1.13	Fully Parameterized Components	49
2.1.14	Formal Specifications.....	54
2.1.15	Multiple Implementations	59
2.1.16	Naming and Formatting Conventions	60
2.2	Client Implementation Principles.....	61
2.2.1	Ada Constructs/Statements.....	61
2.2.2	Variable Declarations	62
2.2.3	Eliminating Unwanted Aliasing	63
2.2.4	Initialization, Finalization and Meeting Preconditions	64
2.3	Component Implementation Principles.....	65
2.3.1	Private Part's Type Declaration.....	66
2.3.2	Package Body Formal Comments.....	68
2.3.3	Eliminating Package Body Incest	72
2.3.4	Implementing Initialize_Package and Finalize_Package.....	73
2.3.5	Implementing Initialize	75
2.3.6	Implementing Finalize	75
2.3.7	Implementing Swap.....	76
2.4	Local Certification of Composability, Correctness, Reusability and Understandability	76
2.4.1	Local Certification of Composability	77
2.4.2	Local Certification of Correctness.....	79
2.4.3	Local Certification of Reusability	85
2.4.4	Local Certification of Understandability.....	86
2.5	Summary.....	87
CHAPTER III Additional Capabilities, Support for Testing and Debugging, and Partial Instantiation		89
3.1	Adding Additional Capabilities to a Component	89
3.2	Support for Testing and Debugging	96
3.2.1	Display of the Abstract Model	96
3.2.2	Display of the Representation.....	101
3.2.3	Display Model Components	104
3.2.4	Checking Component.....	111
3.3	Partial Instantiation	119

3.4	Summary	124
CHAPTER IV	Bootstrapping From Raw Ada	127
4.1	Ada's Built-in Scalar Types	128
4.2	Encapsulating Ada's Record and Array Type Constructors	131
4.2.1	Ada Records	132
4.2.2	Ada Arrays	137
4.3	Encapsulating Ada's Access Type Constructor: Acyclic Linked Structures	141
4.3.1	Intuition Behind <code>One_Way_Nilpotent_Template</code> and its Specification	141
4.3.2	<code>Queue_Template</code> Implemented Using <code>One_Way_Nilpotent_Template</code>	146
4.3.3	Alternative Implementations for <code>One_Way_Nilpotent_Template</code>	153
4.4	Encapsulating Ada's Access Type Constructor: Cyclic Linked Structures	155
4.4.1	Intuition Behind <code>Permutation_Template</code> and its Specification	155
4.4.2	<code>Two_Way_List_Template</code> Implemented Using <code>Permutation_Template</code>	161
4.4.3	Implementing <code>Permutation_Template</code>	168
4.5	Abstract Module State in <code>One_Way_Nilpotent_Template</code> and <code>Permutation_Template</code>	170
4.6	Summary	171
CHAPTER V	Conclusion	173
5.1	Summary and Conclusions	173
5.1.1	Definition of a Software Discipline	173
5.1.2	Formulation of a Specific Software Discipline for Constructing Ada Components	174
5.2	Contributions	176
5.3	Future Work	176
APPENDIX A	Component Specifications	179
A.1	<code>One_Way_List_Template</code>	179
A.2	<code>Two_Way_List_Template</code>	183
A.3	<code>N_Way_Nilpotent_Template</code>	187

A.4	Tuple2_Model_Template.....	190
A.5	Set_Model_Template.....	193
APPENDIX B	Component Implementations.....	197
B.1	Built_In_Types.....	197
B.2	Static_Array_Template.....	203
B.3	One_Way_Nilpotent_Template.....	205
B.4	Permutation_Template.....	209
APPENDIX C	RESOLVE/Ada Naming and Formatting Conventions.....	217
APPENDIX D	Compendium of Principles.....	219
BIBLIOGRAPHY	225

LIST OF FIGURES

Figure 1	—	A Software System as a Collection of Components.....	2
Figure 2	—	Reverse Queue	6
Figure 3	—	Area of Reasoning for Certifying Reverse Queue	7
Figure 4	—	Area of Reasoning Because of a Leaky Abstraction.....	8
Figure 5	—	Linked List Representation for Queue.....	17
Figure 6	—	Initial Configuration for a Queue Variable.....	22
Figure 7	—	Items Needed When Reasoning About a Client.....	29
Figure 8	—	Array Representation of a Bounded Queue.....	29
Figure 9	—	Singly Linked List Representation of a Bounded Queue.....	30
Figure 10	—	Non-local Reasoning Required by Bounded_Queue	32
Figure 11	—	Multiple Implementations of Queue_Template	59
Figure 12	—	Multiple Implementations of Queue_Template in Ada.....	60
Figure 13	—	Client Implementation.....	61
Figure 14	—	Component Implementation	66
Figure 15	—	Items Examined to Locally Certify Composability	77
Figure 16	—	Area of Reasoning for Local Certification of Correctness	80
Figure 17	—	A Client of Components Having Abstract Module-Level State ..	83
Figure 18	—	Items Required to Certify Correctness When Abstract Module- Level State is Present.....	84
Figure 19	—	Items Examined to Locally Certify Reusability and Understandability.....	86
Figure 20	—	Bootstrapping From Raw Ada	128
Figure 21	—	Simulated Pointers Using Parallel Arrays	142
Figure 22	—	Free List of Individual Pieces of Storage	153
Figure 23	—	Free List of Linked Structures	154
Figure 24	—	Simulated Pointers for Doubly Linked Structure	155
Figure 25	—	Two Positions in Their Own Cycle.....	159

Figure 26	—	Two Positions Linked by One Cycle.....	159
Figure 27	—	Cycle Containing More Than Two Positions.....	160
Figure 28	—	One Position Removed From a Cycle Using Transpose_After .	160
Figure 29	—	One Position Removed From a Cycle Using Transpose_Before	160
Figure 30	—	Splitting a Cycle into Two.....	161
Figure 31	—	Joining Two Cycles	161
Figure 32	—	Cyclic Linked Structure With Three Positions	168
Figure 33	—	Circular Doubly Linked List Implementation.....	169
Figure 34	—	Representation Using a Shadow List.....	170

LIST OF TABLES

Table 1	—	Component Interface Principles (Section 2.1) Followed by Published Component Libraries and/or Guidelines.....	175
---------	---	--	-----