

## TABLE OF CONTENTS

Chapter 1: Introduction.....	1
1.1. Major Contributions .....	1
1.2. An Overview of the Dissertation.....	2
Chapter 2: A Survey of Visual Programming.....	5
2.1. Introduction.....	5
2.2. Software Diagramming Techniques.....	11
2.3. Visual Language Formalisms .....	36
2.4. Program Visualization Systems .....	41
2.5. Programming in the Large .....	53
2.6. Visual Programming Languages.....	60
2.7. Summary.....	79
Chapter 3: A Simple Visual Language .....	80
3.1. Introduction.....	80
3.2. Ideas Behind the Language .....	80
3.3. The Prototype .....	90
3.4. Implementation Details .....	96
3.5. Notes.....	97
Chapter 4: Visual APL.....	98
4.1. Introduction.....	98
4.2. Factorial Example .....	98
4.3. The Main Menu.....	99
4.4. The Function Menu.....	101
4.5. Parameter Ordering .....	102

4.6.	Overloading.....	102
4.7.	Set and Use.....	104
4.8.	Testing for Primes.....	105
4.9.	Practical Use.....	108
4.10.	Code Generation.....	109
4.11.	Implementation Details.....	110
4.12.	Related Work.....	110
Chapter 5: Graphs as Programs .....		111
5.1.	Introduction.....	111
5.2.	A Generic Graphic Editor .....	112
5.3.	A Template For Mathematica .....	117
5.4.	Some Visual Mathematica Programs .....	123
5.5.	Executing Graph Traversal.....	129
5.6.	Observations .....	130
5.7.	Possible Extensions.....	130
5.8.	Implementation Details.....	131
Chapter 6: System Design Notation.....		132
6.1.	Introduction.....	132
6.2.	Buhr's Original Notation.....	133
6.3.	Buhr's MachineCharts: Robots and Reactors.....	134
6.4.	Mapping to Ada: Tasks and Protected Records .....	137
6.5.	Asynchronous Transfer of Control.....	139
6.6.	Requeue .....	152
6.7.	Generics.....	158
6.8.	Conclusions .....	166

Chapter 7: Limits of the Visual .....	167
7.1. Introduction.....	167
7.2. Textual Software Metrics.....	167
7.3. Graphic Complexity Measures .....	181
7.4. Analyzing Abstract Diagrams Using Metrics .....	191
7.5. Analyzing Visual Languages With Graphic Metrics.....	200
7.6. An Exploratory Direction: The Infinite Zoom of Pad.....	223
7.7. Diagram Cognition.....	228
Chapter 8: Conclusions .....	232
8.1. Recapitulation.....	232
8.2. Conclusions .....	233
Bibliography .....	236

## FIGURE LIST

Figure 2.1. Sixteenth century representation of the four basic elements. ....	8
Figure 2.2. Sutherland's diagram for calculating a square root.....	9
Figure 2.3. Flowchart example from Shooman (1983).....	15
Figure 2.4. A structure diagram from Martin (1985). ....	17
Figure 2.5. Layer diagram. ....	18
Figure 2.6. A tree structure with its corresponding Warnier-Orr Representation. ....	19
Figure 2.7. Simplified Warner-Orr Diagram.....	19
Figure 2.8. Rothon Diagram. From Brown (1983).....	20
Figure 2.9. Dimensional Flow Chart. From Witty (1977).....	21
Figure 2.10. Finite State Automata. From Johnsonbaugh (1984).....	22
Figure 2.11. State Chart from Rumbaugh (1991).....	23
Figure 2.12. Flow Chart vs. Nassi Schneiderman diagram. ....	24
Figure 2.13. Cell and arrow diagram. From Abelson(1985).....	24
Figure 2.14. Linked list insertion. From Aho (1983). ....	25
Figure 2.15. Programming language display. From Ghezzi (1982). ....	26
Figure 2.16. Tree Traversal. From Aho (1983). ....	26
Figure 2.17. Recursion frames. From Bauer (1982).....	27
Figure 2.18. Recursion with flow charts. From Bauer (1982).....	27
Figure 2.19. Stringcopy using enclosure diagrams. From Reade (1989).....	28
Figure 2.20. Entity-Relationship and OMT representation. ....	29
Figure 2.21. Homomorphism in OMT. From Rumbaugh (1991). ....	30
Figure 2.22. CASE Tool (Software Through Pictures) version of Booch notation. ....	31
Figure 2.23. Petri net. From Johhsonbaugh (1984).....	32

Figure 2.24. Petri Net node types. From Bauer(1982).....	32
Figure 2.25 $x^2 - 2x + 3$ from Shu (1988).....	33
Figure 2.26 Roots of a quadratic equation. From Sharp (1985). ....	33
Figure 2.27. Illustration of token firing, from Bauer(1982).....	33
Figure 2.28. Simple data flow diagram. From Martin (1983).....	34
Figure 2.29. Complex data flow diagram. From Gane (1979).....	35
Figure 2.30. Nested dataflow diagrams. From Fisher (1991).....	35
Figure 2.31. Sieve example. From Abelson (1985).....	36
Figure 2.32. Machine notation from Hopcroft (1979). ....	36
Figure 2.33. Flow chart and language equivalence. From Pratt (1971a).....	38
Figure 2.34. Circuit diagram from a grammar. In Gonzalez (1978). ....	40
Figure 2.35. Chromosome from a grammar. In Gonzalez (1978).....	40
Figure 2.36. Complete 4 node graph. In Gonzalez (1978).....	41
Figure 2.37. Typeset source code. From Baecker (1986).....	42
Figure 2.38. TFPDRAW. In Matsumura (1986).....	43
Figure 2.39. Incense. From Meyers (1983).....	45
Figure 2.40. Castle. From Boecker (1986). ....	46
Figure 2.41. Fooscape. From Boecker (1986).....	47
Figure 2.42. Contour diagrams for a code fragment. From Organick (1974). ....	48
Figure 2.43. Pecan. From Reiss (1985).....	50
Figure 2.44. Balsa. From Brown (1987). ....	52
Figure 2.45. Balsa animation example of a Vornoi diagram algorithm.....	53
Figure 2.46. Software design visualization. From Brown(1985).....	55
Figure 2.47. Pegasys network diagram. From Moriconi (1985) .....	58
Figure 2.48. Mapping graphs with meta-edges. From Goguen (1985).....	59

Figure 2.49. Puzzle piece convention. From Goguen (1985) .....	60
Figure 2.50. PICT factorial example. From Glinert (1985).....	61
Figure 2.51. PICT conditionals and loops. From Glinert (1985).....	62
Figure 2.52. Blox. From Glinert (1986).....	63
Figure 2.53. PROGRAPH square program. From Matwin (1985).....	64
Figure 2.54. PROGRAPH conditionals. From Matwin (1985).....	65
Figure 2.55. PROGRAPH reverse. From Matwin (1985).....	66
Figure 2.56. PROGRAPH factorial. From Matwin (1985).....	67
Figure 2.57. PROGRAPH merge. From Matwin (1985). .....	69
Figure 2.58. PROGRAPH matrix multiply. From Matwin (1985).....	71
Figure 2.59. PROGRAPH find subtree. From Matwin (1985).....	73
Figure 2.60. Show and Tell. From Gillet (1986). .....	74
Figure 2.61. Programming in Pictures. From Raeder (1984). .....	75
Figure 2.62. Visual FP. Pagan (1977).....	76
Figure 2.63. An enclosure convention for lisp. From Lakin (1986).....	77
Figure 2.64. Garden. From Reiss (1987).....	78
Figure 3.1. Data and function concept. ....	80
Figure 3.2. Simple pipes. ....	82
Figure 3.3. Arguments as constant functions. ....	82
Figure 3.4. Type system.....	84
Figure 3.5. Functions as arguments to functions. ....	85
Figure 3.6. Factorial representation. ....	86
Figure 3.7. Awk representation. ....	87
Figure 3.8. Recursion.....	88
Figure 3.9. Fibonacci.....	89

Figure 3.10. User interface layout.....	91
Figure 3.11. User interface zoom for H-graphs.....	92
Figure 3.12. Graphic menu.....	92
Figure 3.13. Use of menus.....	94
Figure 3.14. Hierarchy of representation.....	97
Figure 4.1. Factorial calculation.....	98
Figure 4.2. Main menu.....	99
Figure 4.3. Row and column panel.....	100
Figure 4.4. Blank input array.....	100
Figure 4.5. Function menu.....	101
Figure 4.6. Transpose.....	101
Figure 4.7. Parameter ordering example (2 - 7).....	102
Figure 4.8. Parameter ordering example (7 - 2).....	102
Figure 4.9. Scalar * scalar.....	103
Figure 4.10. Scalar * vector.....	103
Figure 4.11. The reduction of a two-dimensional array.....	104
Figure 4.12. A variance calculation showing the setting and later use of a variable.....	105
Figure 4.13. Primality test on the number 7.....	107
Figure 4.14. Primality test on the number 8.....	108
Figure 5.1. An example from Templa (Hekmatpour (1990)).....	116
Figure 5.2. The graphic editor.....	118
Figure 5.3. The edges available within the graph program.....	119
Figure 5.4. The relations screen.....	119
Figure 5.5. For the graph data family.....	120
Figure 5.6. The links in the graph data family.....	121

Figure 5.7. The relations for graph program. ....	121
Figure 5.8. The icons for the graph data family places and links. ....	122
Figure 5.9. The icons for graph program, ....	122
Figure 5.10. A graph using routines from Skiena (1990) ....	123
Figure 5.11. The graph is exploded from the data node. ....	124
Figure 5.12. A node explosion. ....	125
Figure 5.13. A for loop in Mathematica style. ....	126
Figure 5.14. Mathematica output. ....	127
Figure 5.15. A visual Mathematica program. ....	127
Figure 5.16. Mathematica window overlapped with graphic editor. ....	128
Figure 6.1. Buhr package and task notation. ....	133
Figure 6.2. Sockets in a task. ....	134
Figure 6.3. MachineChart distinctions. ....	135
Figure 6.4. A visit from an engine ....	135
Figure 6.5. A control flag as a reactor. ....	135
Figure 6.6. Expected visit timing diagram for test and set. ....	136
Figure 6.7. A counting semaphor. ....	138
Figure 6.8. Installing and removing a robot. ....	141
Figure 6.9. Using abort. ....	142
Figure 6.10. Using exceptions. ....	143
Figure 6.11. Asynchronous transfer using exceptions. ....	143
Figure 6.12 Proposed convention. ....	144
Figure 6.13. The sequence. ....	146
Figure 6.14. Shell example. ....	147
Figure 6.15. Shell using robots. ....	148



Figure 6.16. Delay visualization. ....	149
Figure 6.17. Delay shorthand. ....	149
Figure 6.18. Cascading transfer of control. ....	151
Figure 6.19. A invokes B and deadlocks. ....	153
Figure 6.20. Requeueing from A to B. ....	154
Figure 6.21. Requeue as a load balancing mechanism. ....	156
Figure 6.22. Stack instantiation from a template. ....	158
Figure 6.23. Generics using existing conventions. ....	160
Figure 6.24. An alternative way of representing generics. ....	161
Figure 6.25. A new convention for generics. ....	162
Figure 6.26. Icons for generics. ....	163
Figure 6.27. Group signature representation. ....	164
Figure 6.28. Using an OMT-like convention. ....	164
Figure 6.29. An example with the new convention. ....	165
Figure 7.1. Flow diagram for the above code. ....	170
Figure 7.2. McCabe complexity by counting regions. ....	171
Figure 7.3. A 1 inch by 6 inch strip of a map of France showing regions. ....	184
Figure 7.4. A textual and graphic representation of containment. ....	191
Figure 7.5. Adjoinment. ....	191
Figure 7.6. Enclosed node representation of a tree. ....	192
Figure 7.7. Open node representation of a tree. ....	192
Figure 7.8. Unbalanced tree representation. ....	193
Figure 7.9. Binary Expression Trees. ....	194
Figure 7.10. Smallest resolvable graphic tree representation. ....	195
Figure 7.11. From a field of 64 possible nodes, 16 can be realistically used. ....	195

Figure 7.12. Smallest resolvable unlabeled graph representation. ....	196
Figure 7.13. Binary Expression Tree. ....	196
Figure 7.14. Simple graph representation. ....	197
Figure 7.15. Nassi-Schneiderman and flow chart representation. ....	200
Figure 7.16. Recursive stringcopy representation from Reade(1989). ....	201
Figure 7.17. Lakin's deeply nested enclosures. ....	202
Figure 7.18. Entity-relationship modeling and Object-Oriented Modeling ....	204
Figure 7.19. Enclosure diagram-equivalent for OMT notation. ....	205
Figure 7.20. Large data flow diagram from Gane (1979). ....	206
Figure 7.21. Program and equivalent flow chart from Pratt (1971a). ....	207
Figure 7.22. String grammar equivalent for a complete 4 node graph. ....	208
Figure 7.23. Two examples of Balsa program visualization. From Brown(1987). ....	209
Figure 7.24. Matwin's PROGRAPH list reversal. ....	210
Figure 7.25. Programming in Pictures. Raeder (1984). ....	211
Figure 7.26. Searching for a string in all files. ....	212
Figure 7.27. Visual APL version of a prime number checker. ....	214
Figure 7.28. A weighted graph input interface. ....	215
Figure 7.29. A program to generate a random graph. ....	216
Figure 7.30. Buhr notation for a select statement. From Buhr (1984). ....	217
Figure 7.31. Buhr diagram for an office environment. From Buhr(1984). ....	218
Figure 7.32. Buhr MachineChart representation of a reactor. ....	220
Figure 7.33. Overview of Dining Philosophers from Buhr (1990). ....	221
Figure 7.34. Detail of Dining Philosophers with SETL-like textual code. ....	222
Figure 7.35. A company balance sheet in Pad. From Perlin (1993). ....	224
Figure 7.36. Overview and detail of a document from Perlin(1993). ....	225

Figure 7.37. Extracts from the previous figure..... 225

Figure 7.38. Directly nested H-graphs..... 227

Figure 7.39. Flowgraph representation using directly nested H-graphs..... 228

Figure 7.40. Test graph..... 229

# CHAPTER 1

## INTRODUCTION

While much of computer science textbooks and classroom lectures are filled with diagrams, and much of our design activity as programmers takes place on whiteboards, we write our programs as text. With the recent proliferation of graphic user interfaces, technology exists that would allow us to draw rather than write programs. This dissertation addresses the question of whether we should be diagramming rather than coding, as well as the broader question of how the visual can be used in the process of programming.

### 1.1. MAJOR CONTRIBUTIONS

This thesis contributes to the field in four areas. The first area is in the analysis of diagrams. The use of diagrams in the field of computer science is thoroughly surveyed for the first time, and some underlying principles identified. The visual conventions of Adjoinment, Linking, and Enclosure are defined and illustrated.

The second area is in the creation of Visual Programming Languages. Three languages are developed - a simple programming language that encompasses shell commands, *awk* and SASL, a visual version of APL, and a visual front end for Mathematica. The visual version of APL is notable in that it presents both a program and instances of data undergoing transformation as part of one unified diagram.

The third area is the creation of system design conventions. Building on the work of R. J. A. Buhr, new visual systems designing conventions are created to handle the intricacies of facilities in the Ada 9X language. Asynchronous transfers of control, requeueing, and generic formal parameters are addressed. The asynchronous transfer of control convention is suitable for CASE representations of the language construct, and can be easily animated.

The fourth area is in the analysis of the effectiveness of graphic representations of computer programs. Some existing software metrics are modified for use in analyzing diagrams, and two new metrics are proposed: graphic token count and diagram class complexity. A graphic design measure, data density, is transformed into a computer science measure, token density. Using these metrics, graphic representations are compared to each other and to textual representations. From this, a strong set of conclusions are drawn about the relative strengths of graphic and textual representation.

## **1.2. AN OVERVIEW OF THE DISSERTATION**

In Chapter 2, the current state of the field is surveyed. The use of diagrams in computer science is looked at first. In particular, the representation of recursion is highlighted. Visual language formalisms are discussed, including Pratt's H-graphs. Program visualization systems, programming in the large, and visual programming languages are all surveyed.

In Chapter 3, a simple visual language is developed, based on a simple two-dimensional design making use of simplicity of functional languages. A visual typing system is also suggested. The

first version of the language runs on top of the Unix shell, performing file manipulations. This language is then modified to run over AWK, and then extended to generate SASL.

A visual version of APL is created in chapter 4. By extending the simple language of chapter 2 to include intermediate output after the invocation of each function, it becomes possible to create easily understandable APL code through combining boxes and running sample data through the visual program.

In Chapter 5, a visual version of a subset of Mathematica is created, based on a generic template editor. This chapter discusses a general approach to the creation of visual languages using essentially a language design tool. Using this tool it becomes possible to easily create a front end for Mathematica by drawing a graph of the program. It is also possible to interactively create data structures that can serve as input to the Mathematica program.

System design notation is discussed in chapter 6. In particular, the two conventions created by Buhr, Buhr diagrams and MachineCharts, are reviewed, and extensions proposed to MachineCharts. Conventions for handling asynchronous control, requeueing, and generic parameters are created.

In chapter 7, textual metrics are examined and visual counterparts proposed. A new measure, Diagram Class Complexity, is created to measure the complexity of families of diagrams. Another metric, Graphic Token Complexity, is created and used together with Textual Token Complexity to analyze the representation of basic diagram types such as trees and graphs. Then the metrics are applied to many of the systems surveyed in chapter 2 and developed in subsequent chapters. From the results of this chapter, diagram cognition is discussed.

In chapter 8, a short recapitulation is performed, followed by strong conclusions about the role of the visual in programming.