

## Reading and Reference List

### 1 Background

#### 1.1 Historical Precedence

McIlroy was among the first to identify the problem of library scalability; the notion virtual machines is due to Dijkstra, and families of systems is due to Parnas.

D. McIlroy, "Mass Produced Software Components", *Software Engineering: Report on a Conference by the Nato Science Committee*, Oct 1968, P. Naur and B. Randell, eds. 138-150.

E.W. Dijkstra, "The Structure of THE Multiprogramming System", *Communications of ACM*, May 1968, 341-346.

D.L. Parnas, "Designing Software for Ease of Extension and Contraction", *IEEE Transactions on Software Engineering*, March 1979.

#### 1.2 Parameterized Programming

The concepts of horizontal parameterization (i.e., parameterizing interfaces) and vertical parameterization (i.e., layering) have been expressed elegantly by Goguen and Tracz. The certification of parameterized components has been examined in the RESOLVE project.

J.A. Goguen, "Reusing and Interconnecting Software Components", *IEEE Computer*, February 1986. Also in Prieto-Diaz and Arango text (below).

W. Tracz, "LILEANNA: A Parameterized Programming Language," *Advances in Software Reuse: Selected Papers from the Second International Workshop on Software Reusability*. Lucca, Italy. R. Prieto-Diaz and W.B. Frakes, eds. IEEE Computer Science Press, March 24-26, 1993.

M. Sitaraman and B. Weide, "Component-Based Software Using RESOLVE", *ACM Software Engineering Notes*, October 1994.

#### 1.3 Large System Development

References that survey the problems of large system development (and indirectly, problems that arise in domain modeling) are:

B. Curtis, H. Krasner, and N. Iscoe, "A Field Study of the Software Design Process for Large Systems", *Communications of the ACM*, November 1988.

G. Booch, *Object Solutions: Managing the Object-Oriented Project*, Addison-Wesley, 1995.

#### 1.4 Object-Oriented Frameworks

GenVoca components encapsulate suites of interrelated classes. So too do object-oriented frameworks; they are suites of interrelated abstract classes that have multiple concrete class implementations, which describe different implementations of what we have called "subsystems". What frameworks lack are parameterizations and method wrappers that are needed to express GenVoca compositions (See **Subjativity** and **Method Wrappers**).

R.H. Campbell and N. Islam, "A Technique for Documenting the Framework of an Object-Oriented System", *IEEE 2nd International Workshop on Object Orientation in Operating Systems (1992)*, 288-300.

R.H. Campbell, N. Islam, and P. Madany, "Choices, Frameworks and Refinement", *Computing Systems*, 5(3), 1992.

R.E. Johnson and B. Foote, "Designing Reusable Classes", *Journal of Object-Oriented Programming*, June/July 1988. Also in Prieto-Diaz and Arango text (below).

R.E. Johnson, "Documenting Frameworks using Patterns", *OOPSLA 1992*, 63-76.

R.E. Johnson, "How to Design Frameworks", Tutorial Notes, 1993.

G.C. Murphy and D. Notkin, "The Interaction Between Static Typing and Frameworks", TR 93-09-02, Dept. Computer Science and Engineering, University of Washington, Seattle, 1993.

#### 1.5 Role-Based Designs and Mixins

Role-based designs is a design technique that encapsulates features of applications through a set of classes that perform specific roles. Role-based designs are an object-oriented design technique that can be used to design GenVoca layers. Implementing role-based designs is through the use of mixins, classes whose superclass are specified via a parameter. The following papers survey recent work on mixins and on role-based designs.

- G. Bracha and W. Cook, "Mixin-Based Inheritance", *ECOOP/OOPSLA 90*, 303-311.
- G. Bracha and D. Griswold, "Extending Smalltalk with Mixins", *Workshop on Extending Smalltalk* at OOPSLA 96. See <http://java.sun.com/people/gbracha/mwp.html>.
- M. Flatt, S. Krishnamurthi, M. Felleisen, "Classes and Mixins". *ACM Symposium on Principles of Programming Languages*, 1998 (PoPL 98).
- G. Kiczales, J. Lamping, A. Mendhekar, C. Maeda, C. Lopes, J. Loingtier, and J. Irwin, "Aspect-Oriented Programming", *ECOOP 97*, 220-242.
- C. Prehofer, "Feature-Oriented Programming: A Fresh Look at Objects", *ECOOP 97*, 419-443.
- M. VanHilst and D. Notkin, "Using C++ Templates to Implement Role-Based Designs", *JSSST International Symposium on Object Technologies for Advanced Software*, Springer-Verlag, 1996, 22-37.
- M. VanHilst and D. Notkin, "Using Role Components to Implement Collaboration-Based Designs". *OOPSLA 1996*.
- M. VanHilst and D. Notkin, "Decoupling Change From Design", *ACM SIGSOFT 1996*.
- M. VanHilst, "Role-Oriented Programming for Software Evolution", Ph.D. Dissertation, University of Washington, Computer Science and Engineering, 1997.

## 1.6 Domain Modeling

A key problem in modeling domains is choosing the right abstractions. The following references offer a variety of practical perspectives on this topic. (See also papers on **Frameworks**, and **Role-based Designs and Mixins**).

- D. Batory, L. Coglianese, M. Goodwin, and S. Shafer, "Creating Reference Architectures: An Example From Avionics", *ACM SIGSOFT Symposium on Software Reusability*, Seattle, 1995, 27-37.
- H. Gomaa, L. Kerschberg, V. Sugumaran, C. Bosch, and I. Tavakoli, "A Prototype Domain Modeling Environment for Reusable Software Architectures", *Third International Conference on Software Reuse*, Rio de Janeiro, November 1-4, 1994, 74-83.

- R. Prieto-Diaz and G. Arango (ed.), *Domain Analysis and Software Systems Modeling*, IEEE Computer Society Press 1991.

## 1.7 Software Reuse

Good overviews of the state-of-art results and problems in software reuse are:

- T. Biggerstaff and C. Richter, "Reusability Framework, Assessment and Directions", *IEEE Software*, March 1987.
- T. Biggerstaff, "An Assessment and Analysis of Software Reuse", in *Advances in Computers*, Volume 34, Academic Press, 1992.
- C. Krueger, "Software Reuse", *ACM Computing Surveys*, 24(2), June 1992, 131-183.
- H. Mili, F. Mili, A. Mili, "Reusing Software: Issues and Research Directions", *IEEE Transactions on Software Engineering*, June 1995, 528-562.

## 1.8 Transformation Systems

GenVoca domain models can be implemented by program transformation systems. Simonyi's paper describes an innovative approach to the integration of transformation systems and compilers to create extensible programming languages. Weigert's paper describes an impressive transformation system that is being used at Motorola to produce customized software for different radio-products. Griswold's paper deals with semantics preserving program transformations. The papers by Roberts, et al and Tokuda et al describe innovative approaches to creating tools for editing OO programs by transformations. Pu's paper is an example of a dynamic generative generator, and the other papers deal with transformations in the context of "parameterized layers" of rewrite rules:

- I. Baxter, "Design Maintenance Systems", *CACM* April 1992, 73-89.
- I. Baxter, "Transformation Systems: Theory, Implementation, and Survey", Tutorial Notes, 1996.
- L. Blaine and A. Goldberg, "DTRE - A Semi-Automatic Transformation System", in *Constructing Programs from Specifications*, Elsevier Science Publishers, 1991.
- W.G. Griswold, "Direct Update of Data Flow Representations for a Meaning-Preserving Program Restructuring Tool", *ACM SIGSOFT 1993*.

- J. Neighbors, "Draco: A Method for Engineering Reusable Software Components", in T.J. Biggerstaff and A. Perlis, eds., *Software Reusability*, Addison-Wesley/ACM Press, 1989.
- H. Partsch and R. Steinbruggen, "Program Transformation Systems", *Computing Surveys*, March 1983, 199-236.
- C. Pu, H. Massalin, and J. Ioannidis, "The Synthesis Kernel", *Computing Systems*, 1(1):11-32, Winter 1988.
- D. Roberts, J. Brant, and R. Johnson, "A Refactoring Tool for Smalltalk", University of Illinois at Urbana-Champaign, Dept. Computer Sciences, 1997.
- C. Simonyi, "The Death of Computer Languages, the Birth of Intentional Programming", Microsoft Corporation, Sept 1995.
- D.R. Smith, "KIDS: A Semiautomatic Program Development System", *IEEE Transactions on Software Engineering*, Sept. 1990, 1024-1043.
- L. Tokuda and D. Batory, "Automated Software Evolution via Design Pattern Transformations", *Proc. 3rd International Symposium on Applied Corporate Computing*, Monterrey, Mexico, Oct. 1995.
- T.J. Weigert, J.M. Boyle, T.J. Harmer, "Knowledge-Based Derivation of Programs from Specifications", *Artificial Intelligence in Automation*, World Scientific Press, 1996.

### 1.9 Software Architectures

Software architectures deal with issues of "programming-in-the-large" and building software systems from components. A variety of popular perspectives are:

- E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison-Wesley, 1994.
- D. Garlan and M. Shaw, "An Introduction to Software Architecture", in *Advances in Software Engineering and Knowledge Engineering*, Volume I, World Scientific Publishing Company, 1993.
- D. Garlan, et al, "Architectural Mismatch or Why It's Hard to Build Systems out of Existing Parts", *ICSE 1995*.
- M.M. Gorlick and R.R. Razouk, "Using Weaves for Software Construction and Analysis", *Proc. ICSE 1991*, 23-34.

- D.E. Perry and A.L. Wolf, "Foundations for the Study of Software Architecture", *ACM SIGSOFT Software Engineering Notes*, October 1992, 40-52.
- J. Udell, "Componentware", *BYTE*, May 1994.
- E. White and J. Purtilo, "Integrating the Heterogeneous Control Properties of Software Modules", *ACM SIGSOFT 1992*.

### 1.10 Subjectivity

When modeling families of related applications, objects do not have single interfaces. Rather, they are described by families of related interfaces. The interface that is appropriate for an application is application specific (i.e., subjective). The following references give an overview of current thinking on the topic. Also see **Role-Based Designs and Mixins**.

- D. Batory and B.J. Geraci, "Validating Component Compositions and Subjectivity in GenVoca Generators", *IEEE Transactions on Software Engineering*, February 1997, 67-82.
- W. Harrison and H. Ossher, "Subject-Oriented Programming (A Critique of Pure Objects)", *OOPSLA 1993*, 411-428.
- W. Harrison, H. Ossher, R.B. Smith, and D. Ungar, "Subjectivity in Object-Oriented Systems: Workshop Summary", *Addendum to OOPSLA 1994*.
- H. Ossher and W. Harrison, "Combination of Inheritance Hierarchies", *Proc. OOPSLA 1992*, 25-40.
- H. Ossher, et al., "Subject-Oriented Composition Rules", *OOPSLA 1995*, 235-250.
- Y. Smaragdakis and D. Batory, "Implementing Reusable OO Components", *ICSR 1998*.
- Y. Smaragdakis and D. Batory, "Implementing Layered Designs with Mixin Layers", *ECOOP 1998*.
- M. Van Hilst and D. Notkin, "Using Role Components to Implement Collaboration-Based Designs", *OOPSLA 1996*, 359-369.
- M. Van Hilst and D. Notkin, "Using C++ Templates to Implement Role-Based Designs", *Proc. JSSST Int. Symposium on Object Technologies for Advanced Software*, Springer-Verlag 1996, 22-37.
- M. Van Hilst and D. Notkin, "Decoupling Change from Design", *SIGSOFT 1996*.

### 1.11 Component Design

The following papers give an overview of techniques on how components can be implemented. Please refer to the **Subjectivity** lecture for an overview. (See also **Method Wrappers** and **Role-based Designs and Mixins**).

- J.S. Heidemann and G.J. Popek, "File-System Development with Stackable Layers", *ACM Transactions on Computer Systems*, 12(1), 58-89.
- N. Hutchinson and L. Peterson, "The *x*-kernel: an Architecture for Implementing Network Protocols", *IEEE Trans. Software Engineering*, January 1991.
- M. Van Hilst and D. Notkin, in **Subjectivity** above.
- Y. Smaragdakis and D. Batory, in **Subjectivity** above.

### 1.12 Method Wrappers

The encapsulation of method wrappings within components is an important part of the GenVoca model. The following references provide a state-of-the-art look at current ideas in method wrappings:

- D. Batory, "Subjectivity and GenVoca Generators", *Fourth International Conference on Software Reuse, Orlando, Florida*, April 1996.
- D. Batory and Y. Smaragdakis, "Another Look at Architectural Styles and ADAGE", Loral FSD Owego T.R. ADAGE-UT-95-02.
- S. Danforth and I. Forman, "Reflections on Metaclass Programming in SOM", *OOPSLA 1994*, 440-452.
- I.R. Forman, S. Danforth, and H. Madduri, "Composition of Before/After Metaclasses in SOM", *OOPSLA 1994*, 427-439.
- P. Graham, *ANSI Common Lisp*, Prentice Hall, 1995.
- J.S. Heidemann and G.J. Popek, "File-System Development with Stackable Layers", *ACM Transactions on Computer Systems*, 12(1), 58-89.
- G. Kiczales, J. des Rivieres, and D.G. Bobrow, *The Art of the Metaobject Protocol*, MIT Press, 1991.

### 1.13 Validating Component Compositions

Shallow consistency checking is an important technique used in software architectures and software system generators in order to validate compositions of compo-

nents. It offers a practical approach to consistency checking without requiring full-fledged verification.

- D. Batory and B.J. Geraci, "Validating Component Compositions and Subjectivity in GenVoca Generators", *IEEE Transactions on Software Engineering*, February 1997, 67-82.
- M.D. Katz and D.J. Volper, "Constraint Propagation in Software Libraries of Transformation Systems", *Int. Journal Software Engineering and Knowledge Engineering*, Vol. 2 #3 (1992), 355-374.
- M. Moriconi and X. Qian, "Correctness and Composition of Software Architectures", *ACM SIGSOFT 1994*.
- D. E. Perry, "The Inscape Environment", *Proc. ICSE 1989*, 2-12.
- D.E. Perry, "Software Interconnection Models", *ICSE 1987*.
- D.E. Perry, "The Logic of Propagation in The Inscape Environment", *ACM SIGSOFT 1989*.

### 1.14 Scalability

Papers that put forth similar arguments for library scalability are:

- D. Batory, V. Singhal, M. Sirkin, and J. Thomas, "Scalable Software Libraries", *ACM SIGSOFT 1993*.
- T. Biggerstaff, "The Library Scaling Problem and the Limits of Concrete Component Reuse", *Third International Conference on Software Reuse, Rio de Janeiro*, November 1-4, 1994, 102-110.
- I.R. Forman, S. Danforth, and H. Madduri, "Composition of Before/After Metaclasses in SOM", *OOPSLA 1994*, 427-439.
- D. McIlroy, "Mass Produced Software Components", *Software Engineering: Report on a Conference by the Nato Science Committee*, Oct 1968, P. Naur and B. Randell, eds. 138-150.
- H. Ossher and W. Harrison. "Combination of Inheritance Hierarchies", *Proc. OOPSLA 1992*, 25-40.

### 1.15 Generators

GenVoca is not the only way in which software generators are implemented. Other approaches include:

- B. Abbott, T. Bapty, T., C. Biegl, G. Karsai, J. Sztipanovits: "Model-Based Approach for Software Synthesis," *IEEE Software*, pp. 42-53, May, 1993.
- H. Gomaa, L. Kerschberg, et. al., paper in **Domain Modeling** (above).
- M.L. Griss and K.D. Wentzel, "Hybrid Domain-Specific Kits for a Flexible Software Factory", *Proc. ACM SAC'94*, March 1994, 47-52.
- G. Kiczales, J. Lamping, A. Mendhekar, C. Maeda, C. Lopes, J. Loingtier, and J. Irwin, "Aspect-Oriented Programming", *ECOOP 97*, 220-242.
- R. Kieburtz, L. McKinney, J. Bell, J. Hook, A. Kotov, J. Lewis, D. Oliva, T. Sheard, I. Smith and L. Walton, "A Software Engineering Experiment in Software Component Generation", *ICSE1996*.
- J.C.S. do Prado Leite, M. Sant'Anna, and F.G. de Freitas, "Draco-Puc: A Technology Assembly for Domain Oriented Software Development", *Third International Conference on Software Reuse, Rio de Janeiro*, November 1-4, 1994, 94-101.
- K.J. Lieberherr, *Adaptive Object-Oriented Software: The Demeter Method with Propagation Patterns*, PWS Publishing Company, Boston, 1996.
- J. Neighbors, "The Draco Approach to Constructing Software from Reusable Components", *IEEE Trans. Software Engineering*, Sept. 1984, 564-574.
- J.Q. Ning, K. Miriyala, and W. Kozaczynski, "An Architecture-Driven, Business-Specific, and Component-Based Approach to Software Engineering", *Third International Conference on Software Reuse, Rio de Janeiro*, November 1-4, 1994, 84-93.
- G. Novak, "Creation of Views for Reuse of Software with Different Data Representations", *IEEE Transactions on Software Engineering*, December 1995, 993-1005.

## 2 GenVoca

The first paper covers many of the basic concepts of GenVoca. The second paper focuses on composition validation and subjectivity. The third paper presents a broad overview of results and experiences using GenVoca generators.

- D. Batory and S. O'Malley, "The Design and Implementation of Hierarchical Software Systems with Reusable Components", *ACM TOSEM*, October 1992.

- D. Batory, "Component Validation and Subjectivity in GenVoca Generators", *to appear IEEE Trans. Software Engineering*, 1997.
- D. Batory, "Intelligent Components and Software Generators", Invited presentation to the Software Quality Institute Symposium on Software Reliability, Austin, Texas, April 1997. Technical Report 97-06, Department of Computer Sciences, University of Texas at Austin, February 1997

Also see <http://www.cs.utexas.edu/users/schwartz/index.html>

### 2.1 ADAGE (Avionics)

- D. Batory and Y. Smaragdakis, "Another Look at Architectural Styles and ADAGE", Loral FSD Owego T.R. ADAGE-UT-95-02.
- D. Batory, "Subjectivity and Software System Generators", *Fourth International Conference on Software Reuse*, Orlando, Florida, April 1996.
- D. Batory, L. Coglianese, M. Goodwin, and S. Shafer, "Creating Reference Architectures: An Example From Avionics", *ACM SIGSOFT Symposium on Software Reusability*, Seattle, 1995, 27-37.
- L. Coglianese and R. Szymanski, "DSSA-ADAGE: An Environment for Architecture-based Avionics Development", *Proceedings of AGARD 1993*.
- D. McAllester, "Variational Attribute Grammars for Computer Aided Design." ADAGE-MIT-94-01.
- W. Tracz and L. Coglianese, "An Adaptable Software Architecture For Integrated Avionics", *Proceedings of NAECON'93*, May 1993.

Also see <http://www.sei.cmu.edu/arpa/dssa/dssa-adage/dssa.html>

### 2.2 Avoca/x-kernel (Communication Networks)

- N. Hutchinson, L. Peterson, S. O'Malley, and M. Abbott, "RPC in the x-Kernel: Evaluating New Design Technique", *Symposium on Operating System Principles*, (December 1989), 91-101.
- N. Hutchinson and L. Peterson, "The x-kernel: an Architecture for Implementing Network Protocols", *IEEE Trans. Software Engineering*, January 1991.
- S. O'Malley and L. Peterson, "A Dynamic Network Architecture", *ACM Transactions on Computer Systems*, 10, 2, May 1992.

S. O'Malley, H. Orman, E. Menze III, and L. Peterson, "A Fast General Implementation of Mach IPC in a Network", *Proceedings of the USENIX Mach III Symposium*, April 1993.

Also see: <http://www.cs.arizona.edu/xkernel/www/index.html>

### 2.3 Ficus (File Systems)

R.G. Guy, J. Heidemann, W. Mak, T. Page, G. Popek, and D. Rothmeier, "Implementation of the Ficus Replicated File System", *USENIX Conference*, June 1990.

R.G. Guy, G.J. Popek and T.W. Page, "Consistency Algorithms for Optimistic Replication", *Proceedings of the First International Conference on Network Protocols*, IEEE Press, October 1993.

J.S. Heidemann and G.J. Popek, "File-System Development with Stackable Layers", *ACM Transactions on Computer Systems*, 12(1), 58-89.

P. Reiher, J. Heidemann, D. Ratner, and G. Popek, "Resolving File Conflicts in the Ficus File System", *Proceedings of the 1994 Summer USENIX Conference*.

Also see <http://www.isi.edu/~johnh/WORK/index.html>

### 2.4 Genesis (Database Management Systems)

D.S. Batory, J.R. Barnett, J.F. Garza, K.P. Smith, K. Tsukuda, B.C. Twichell, T.E. Wise, "GENESIS: An Extensible Database Management System", *IEEE Transactions on Software Engineering*, Vol. 14 #11 (November 1988), 1711-1730.

D.S. Batory, "Concepts for a Database System Synthesizer", *ACM Principles Of Database Systems Conference* 1988, 184-192. Also in Prieto-Diaz and Arango text (above).

D. Batory and J. Barnett, "DaTE: The Genesis DBMS Software Layout Editor", in *Conceptual Modeling, Databases, and CASE: An Integrated View of Information Systems Development*, P. Loucopoulos and R. Zicari, editors, Wiley, 1992.

D. Batory and D. Vasavada, "Software Components for Object-Oriented Database Systems", *International Journal of Software Engineering and Knowledge Engineering*, 2 #3, 1993, 165-192.

Also see <http://www.cs.utexas.edu/users/schwartz/index.html>

### 2.5 P2 and P3 (Data Structures)

M. Sirkin, D. Batory, and V. Singhal, "Software Components in a Data Structure Precompiler", *Proc. ICSE* 1993.

D. Batory, V. Singhal, M. Sirkin, and J. Thomas, "Scalable Software Libraries", *ACM SIGSOFT* 1993.

D. Batory, J. Thomas, and M. Sirkin, "Reengineering a Complex Application Using a Scalable Data Structure Compiler", *ACM SIGSOFT* 1994.

D. Batory and J. Thomas, "P2: A Lightweight DBMS Generator", to appear *Journal of Intelligent Information Systems*, 1997.

D. Batory, G. Chen, E. Robertson, and T. Wang, "Web-Advertised Generators and Design Wizards", *International Conference on Software Reuse* 1998.

G. Jimenez-Perez and D. Batory, "Memory Simulators and Software Generators", to appear in *1997 Symposium on Software Reuse*.

Also see <http://www.cs.utexas.edu/users/schwartz/index.html>

### 2.6 Jakarta Tool Set

D. Batory, B. Lofaso, and Y. Smaragdakis, "JTS: Tools for Implementing Domain-Specific Languages", *5th International Conference on Software Reuse*, Victoria, Canada, June 1998.

Also see <http://www.cs.utexas.edu/users/schwartz/index.html>