Tutorial ICSE 2000

Product-Line Architectures, Aspects, and Reuse

Don Batory
Department of Computer Sciences
University of Texas
Austin, Texas 78712
batory@cs.utexas.edu

A *Product-Line Architecture (PLA)* is a blue-print for developing large families of related software applications quickly and cheaply from reusable components. Implementations of PLAs are *generators* (also known as component configuration tools). Generators are among the pinnacle of results on software reuse: they are implementations of reference architectures that define how applications of a domain can be assembled by composing prewritten components.

In this tutorial, we present the GenVoca software design methodology for creating product-lines and building architecturally-extensible software — i.e., software that is extensible via component additions and removals. GenVoca takes the idea of components that export and import standardized interfaces to its logical conclusion; it is also a scalable outgrowth of an old and practitioner-ignored methodology called *step-wise refinement*, which advocates that efficient programs can be created by revealing implementation details in a progressive manner. Traditional work on step-wise refinement focussed on microscopic program refinements (e.g., `x+0` $\Rightarrow$ `x`), for which one had to apply hundreds or thousands of refinements to yield admittedly small programs. While the approach is fundamental and industrial infrastructures are on the horizon, GenVoca extends step-wise refinement largely by scaling refinements to a component or layer (i.e., multi-class modularization) granularity, so that applications of great complexity can be expressed as a composition of a few large-scale refinements.

GenVoca PLA designs have been created for diverse domains: 2-way radios, extensible compilers, communication protocols, command-and-control fire support, avionics, and matrix computation libraries [2][3][6][7]. GenVoca designs are used in industry; its central concepts integrate a wide variety of contemporary and classical research topics, including: aspect-oriented programming, parameterized programming, OO frameworks, collaboration-based design [1], Perry's lite semantics [8][5], generative programming [7], design maintenance [9], and layered software.

References

[1] Y. Smaragdakis and D. Batory, “Implementing Layered Designs with Mixin Layers”. *12th European Conference on Object-Oriented Programming*, (ECOOP '98), July 1998.

[2] D. Batory and S. O’Malley, “The Design and Implementation of Hierarchical Software Systems with Reusable Components”, *ACM Transactions on Software Engineering and Methodology*, October 1992.

[3] D. Batory, V. Singhal, M. Sirkin, and J. Thomas, “Scalable Software Libraries”, *ACM SIGSOFT*, December 1993.

[4] D. Batory, L. Coglianese, M. Goodwin, and S. Shafer, “Creating Reference Architectures: An Example from Avionics”, *1995 Symposium on Software Reuse* (Seattle, Washington).

[5] D. Batory and B. Geraci, “Validating Compositions and Subjectivity and GenVoca Generators”, *IEEE Transactions on Software Engineering*, February 1997.

[6] D. Batory, B. Lofaso, and Y. Smaragdakis, “JTS: A Tool Suite for Building GenVoca Generators”, *5th International Conference on Software Reuse*, June 1998.

[7] K. Czarnecki and U.W. Eisenecker, “Components and Generative Programming”, *ACM SIGSOFT*, 1999.

[8] D. Perry, “The Logic of Propagation in The Inscape Environment”, *ACM SIGSOFT* 1989.

[9] I. Baxter, “Design Maintenance Systems”, *CACM*, April 1992.