Teaching Java Framework Design Using Classic Problems

Tutorial Presentation

H. Conrad Cunningham
Department of Computer and Information Science
University of Mississippi
University, MS 38677
(662) 915-5358
cunningham@cs.olemiss.edu

Yi Liu
Department of Electrical Engineering and Computer Science
South Dakota State University
Brookings, SD 57007
yi.liu@sdstate.edu

Cuihua Zhang
Department of Computer Information Systems
Northwest Vista College
San Antonio, TX 78251
czhang@accd.edu

In a 1976 article [4] Parnas observes, "Variations in application demands, variations in hardware configurations, and the ever-present opportunity to improve a program means that software will *inevitably* exist in many versions." He argues that the design of a program should be approached as the design of a family of related programs. He defines a program family as a set of programs "whose common properties are so extensive that it is advantageous to study the common properties of the programs before analyzing individual members." If programmers can identify and exploit these "common aspects and predicted variabilities" [6], the resulting software can be constructed to reuse code for the common parts and to enable convenient adaptation of the variable parts for specific applications. A quarter-century after his original article, Parnas notes that there is "growing academic interest and some evidence of real industrial success in applying this idea," yet "the majority of industrial programmers seem to ignore it in their rush to produce code" [5]. If software families are to become pervasive, students need to learn to design and construct them effectively.

Software families are difficult to teach in a college course because their design may require extensive knowledge of an application's domain and the use of special-purpose languages and tools [6]. However, the form of software family called a *software framework* is more accessible because the framework techniques build upon standard object-oriented concepts that students learn in undergraduate courses. A framework is essentially the reusable skeleton of a family implemented entirely in an object-oriented programming language. The common aspects are expressed by a set of abstract and concrete "classes that cooperate closely with each other and together embody a reusable solution" [1] to problems in the application domain. The framework can be customized to a specific member of the family by "plugging in" appropriate subclasses at the supported points of variability.

This tutorial is based, in part, on article [2]. The tutorial introduces the technical concepts and techniques for the design and use of software frameworks in Java, giving attention to the teaching of framework concepts in the classroom. It uses function generalization [2,3] and other systematic approaches to framework design. It illustrates these techniques using two case studies. The first is the family of programs that use the well-known divide and conquer algorithmic strategy. The second is the family of

programs that carry out traversals of binary trees. Because students study these classic data structures and algorithms in a typical computing science curriculum, the example frameworks can be used in the classroom without requiring much time for domain analysis.

REFERENCES

[1] Budd, T., *An Introduction Object-Oriented Programming*, Third edition, Boston, MA: Addison-Wesley, 2002.

[2] Cunningham, H. C., Liu, Y., Zhang, C., Using classic problems to teach Java framework design, S*cience of Computer Programming*, Special Issue on Principles and Practice of Programming in Java (PPPJ 2004), 59 (1-2), 147-169, 2006.

[3] Cunningham, H. C., Tadepalli, P., Using function generalization to design a cosequential processing framework, *Proceedings of the 39th Hawaii International Conference on System Sciences (HICSS),* Kauai, Hawaii, IEEE, 2006.

[4] Parnas, D. L., On the design and development of program families, *IEEE Transactions on Software Engineering,* SE-2, (1), 1-9, 1976.

[5] Parnas, D. L., Software design, In: Hoffman, D. M., Weiss, D. M., editors. *Software Fundamentals: Collected Papers by David L. Parnas*, Boston, MA: Addison-Wesley, 2001.

[6] Weiss, D. M., Lai, C. T. R., *Software Product-Line Engineering: A Family-Based Software Development Process*, Reading, MA: Addison-Wesley, 1999.

PRESENTERS

H. Conrad Cunningham is Chair and Associate Professor of Computer and Information Science at the University of Mississippi (Ole Miss). His professional interests include software architecture, programming methodology, and concurrent and distributed computing. He has a BS degree in mathematics from Arkansas State University and MS and DSc degrees in computer science from Washington University in St. Louis. Cunningham has taught courses on software engineering, software architecture, software components, object-oriented programming, and other topics related to framework design.

Yi Liu is an Assistant Professor of Software Engineering at South Dakota State University. Her professional interests are in software engineering, component-oriented programming languages, and artificial intelligence. She has a Master's degree in computer science from Nanjing University in P.R. China and a PhD degree in computer science from the University of Mississippi. Liu's teaching has included courses on software requirements and specification, software architecture, and C++ programming.

Cuihua Zhang is faculty member in Computer Information Systems at Northwest Vista College. Her professional interests include software engineering, networking, security, and computing education. She has both a Bachelor's degree and a Master's degree in English Language and Literature from Northeast Normal University in P.R. China and has an MS degree in Computer and Information Science and a PhD in Curriculum and Instruction from the University of Mississippi. Zhang's teaching has included courses on Java programming, database management, and computing security.