Guest Editors' Introduction: Next Generation Software Reuse

Premkumar T. Devanbu, Dewayne E. Perry, and Jeffrey S. Poulin, *Senior Member*, *IEEE*

Increasing levels of software reuse constitute one of the most pervasive and profound influences in software engineering today. Technological innovations (e.g., object-oriented languages, distributed object models, domain specific languages) and process innovations (e.g., domain modeling, architectural analysis, reuse metrics) are enabling organizations to meet ever-more stringent cost, quality, and interval requirements. Furthermore, there is actually now a reuse market-place: One can literally buy class libraries, frameworks, and components out of catalogues.

The Fifth International Conference on Software Reuse that was held in Victoria, B.C., Canada, included papers, mini-workshops, panels, and tutorials dedicated to various aspects of the software reuse enterprise, including object-orientation, domain modeling, domain specific languages, software architecture, and reuse over the internet. Out of the 31 papers (selected from 96 submissions) that were presented, a select few were nominated by the program committee as representing special, innovative contributions. These were subjected a special rereview and two were selected for this special issue.

These two papers represent two of the major technological approaches to software reuse: generative/transformational reuse, and object-oriented reuse.

The first paper, by Batory, Chen, Robertson, and Wang, considers the problem of selecting the best realizations and algorithms for a general purpose data structure, a *container* [1]. There are several different implementation choices for such a data structure: linked lists, hash tables, binary trees, etc. Some of these can also be used in conjunction on the same container data structure to provide multiple access modes. Containers can also be concurrent, persistent, secure, and so on. Each of these feature combinations offers specific performance characteristics suitable for specific usage profiles. Selecting the specific combinations of implementation features, given a particular workload, involves the application of specialized design knowledge. The Batory et al. paper describes a framework that encapsulates this design knowledge into a *design wizard*. This design wizard helps a programmer select a particular combination of container implementation features tailored to a specific application and usage profile. Once a feature set has been selected, code that implements that specific feature combination is generated after applicable optimizations have been performed. As reuse libraries become larger, more feature-rich, and complex, the design choices involved in using them become more difficult; Biggerstaff has called this the "library scaling problem" [4]. The wizard approach of Batory et al. is an important step toward ameliorating this problem.

The second paper, by Maruyama and Shima, is concerned with the problem of adapting class libraries for specific requirements. The authors note that OO reuse typically involves two phases: *finding* a related class, and *modifying* it to suit the new requirements. They further observe that during the modification phase, developers make heavy use of *change precedents*: What types of changes have been made before in similar contexts. The modification phase, then, can be viewed as a process of *integrating* change precedents. They seek to provide automated support for this activity. Changes made to class libraries are stored in change histories. Using their tools involves several steps: Identify a candidate class for adaptation, then identify change precedents, and, finally, integrate change precedents into the candidate class. Their approach is based on [2] and makes use of program slicing, dependence graph, and graph matching. Maruyama and Shima's work can be viewed as a way to systematically capture and replay typical modifications and extensions that occur during the evolution of object-oriented systems.

Some of the early stirrings of software reuse can be traced back to 1958 [1], with the introduction of separate compilation of subroutines in Fortran II, and the increasing use of Fortran subroutine libraries in scientific computation over the following years. There has been dramatic progress since then in languages, component models, transformational techniques, reuse metrics, and processes. The papers in this special section, and those in the *Proceedings of the Fifth International Conference of Software Reuse* contain important, intriguing ideas that will sustain the progress of software reuse research. We invite you to join in this exciting endeavor.

Premkumar T. Devanbu
Dewayne E. Perry
Jeffrey S. Poulin

- *P.T. Devanbu is with the Computer Science Department, University of California at Davis, Davis, CA 95616. E-mail: devanbu@cs.ucdavis.edu.*
- *D.E. Perry is with the Electrical and Computer Engineering Department, University of Texas at Austin, Austin, TX 78712. E-mail: perry@ece.utexas.edu.*
- *J.S. Poulin is with Atlas Commerce. E-mail: Poulin.Jeff@atlascommerce.com.*

For information on obtaining reprints of this article, please send e-mail to: tse@computer.org, and reference IEEECS Log Number 110651.

REFERENCES

[1] J. Backus, "The History of FORTRAN I, II, III," *IEEE Annals of the History of Computing*, vol. 20, no. 4, 1998.
[2] S. Horwitz, J. Prins, and T. Reps, "Integrating Noninterfering Versions of Programs," *ACM Trans. Programming Languages and Systems*, vol 11, no. 3, July 1989.
[3] D.R. Musser and A.A. Stepanov, "Algorithm Oriented Generic Libraries," *Software Practice and Experience*, vol. 24, no. 7, 1994.
[4] T. Biggerstaff, "The Library Scaling Problem and the Limits of Concrete Component Reuse," *Proc. Int'l Conf. Software Reuse*, pp. 102-110, Nov. 1994.

Premkumar T. Devanbu received his BTech in electrical engineering (light current) from the Indian Institute of Technology, Madras, India, in 1977, and his MS and PhD in computer science from Rutgers University, Piscataway, New Jersey, in 1979 and 1994. He has developed both systems and telecommunications software, first at the Perkin-Elmer Corporation and then at AT&T Bell Laboratories from 1979 to 1985. From 1985, he was a researcher at AT&T Bell Labs in Murray Hill, New Jersey, until the trivestiture in 1996. From 1996 to 1997, he was a principal technical staff member at AT&T Research. In January 1998, he joined the faculty of the Department of Computer Science at the University of California at Davis.

Prof. Devanbu's research interests include retargetable, flexible software tools, software reuse, and approaches to building trust in software engineering tools and processes that are sensitive to intellectual property protection. More details are available at http://www.cs.ucdavis.edu/~devanbu.

Dewayne E. Perry is currently the Motorola Regents Chair of Software Engineering at The University of Texas at Austin. The first half of his computing career was spent as a professional programmer, with the latter part combining both research (as a visiting faculty member in computer science at Carnegie-Mellon University) and consulting in software architecture and design. The last 16 years were spent doing software engineering research at Bell Laboratories in Murray Hill, New Jersey. His appointment at UT Austin began in January 2000. His research interests include (in the context of software system evolution) empirical studies, formal models of the software processes, process and product support environments, software architecture, and the practical use of formal specifications and techniques. He is particularly interested in the role architecture plays in the coordination of multisite software development, as well as its role in capitalizing on company software assets in the context of product lines. His educational interests at UT include building a great software engineering program, both at the graduate and undergraduate levels, creating a software engineering research center, and focusing on the empirical aspects of software engineering to create a mature and rigourous empirical software engineering discipline. He is a co-editor-in-chief of Wiley's *Software Process: Improvement & Practice*; a former associate editor of the *IEEE Transactions on Software Engineering*; a member of ACM SIGSOFT and the IEEE Computer Society; and has served as organizing chair, program chair, and program committee member on various software engineering conferences.

Jeffrey S. Poulin earned his Bachelor's degree from the United States Military Academy at West Point and his Master's and PhD degrees from Rensselaer Polytechnic Institute in Troy, New York. He recently joined Atlas Commerce, a provider of business-to-business e-commerce solutions as the chief technology officer. He previously worked as a senior software engineer and systems architect with Lockheed Martin Federal Systems (formally Loral Federal Systems and IBM Federal Systems Company) in Owego, New York. As a member of the Postal Software Development/Products Group, he has led the technical activities on major large-scale programs for the United States Department of Defense, Postal Service, and commercial customers. As a senior member of the IEEE, he actively participates in conferences and standards activities of the IEEE, the IEEE Computer Society, and the ACM. Dr. Poulin has more than 60 publications on software measurement, software reuse, and domain-specific software architectures, including a book titled *Measuring Software Reuse* (Addison-Wesley). Dr. Poulin is a Hertz Foundation Fellow.