LAWRENCE
LIVERMORE
NATIONAL
LABORATORY

LLNL-TR-401150

FY07 LDRD Final Report Catalyzing the Adoption of Software Components

T. G. W. Epperly, G. K. Kumfert, T. Panas, D. J. Quinlan

February 8, 2008

Disclaimer

This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

FY07 LDRD Final Report
Catalyzing the Adoption of Software Components
LDRD Project Tracking Code: 05-ERD-012

Thomas Epperly, PI
Gary Kumfert
Thomas Panas
Daniel Quinlan
Center for Applied Scientific Computing
Lawrence Livermore National Laboratory
05-ERD-012
February 11, 2008

Software component technology has revolutionized software engineering for standard business-oriented software, enabling codes to be larger, more modular, scalable, robust, and amendable to change. Our objective was to make these benefits available to laboratory scientific codes through research and development in technology to automatically transform legacy software from its current form to a form using software component technology.

Despite budget reductions and early project termination, this project managed to produce useful results and software analysis technology that is being used by laboratory coding groups. In particular, coding groups are using our software analysis and metrics to understand the strengths and weaknesses of their current software architecture. Furthermore, some of the tools initially developed by this LDRD are being used by laboratory coding teams to find violations of coding standards, potential security flaws, and black-listed programming constructs. Finally, the software analysis and visualization technology from this project formed the basis for a new LDRD on software security. (07-ERD-057 Software Security Analysis).

Our technical approach and accomplishments are covered in the three refereed publications that follow this introduction. The first publication [Quinlan04] covers the motivation for automatically transforming software to a component-based architecture, summarizes our overall technical approach, and presents some of our early accomplishments. Our research covered a variety of clustering techniques to identify candidates for component interfaces including spring-embedders (connection based) and K-means (feature based).

The second and third publications [Panas07a,Panas07b] discuss new work in whole program visualization. These papers also indirectly show the various types of software analysis and metrics we developed for use in identifying component interfaces. By making our analysis available to developers in an easy to use graphical user interface, we provided coding groups with a valuable tool to get a high-level understanding of their software and identify areas where the design and architecture can be improved. For example, the analysis and visualization techniques developed by this project were applied to the source code for ROSE, the language analysis and transformation tool used by the project. The tool identified the unparser as a particularly complex part, and on that basis, the unparser was refactored to be easier to maintain and extend.

References

[1] Panas, T., D. Quinlan, and R. Vuduc. Analyzing and Visualizing Whole Program Architectures. ICSE Workshop on Aerospace Software Engineering (AeroSE), Minneapolis, MN, May 21-22, 2007. Also available as UCRL-PROC-231453.

[2] Panas, T., T. G. W. Epperly, D. J. Quinlan, A. Sæbjørsen, and R. Vuduc. Communicating Software Architecture using a Single-View Visualization. 12th IEEE Int. Conference on Engineering of Complex Computer Systems, Auckland, New Zealand, July 11-14, 2007. Also available as UCRL-CONF-227293.

[3] Quinlan, D., Q. Yi, G. Kumfert, T. Epperly, T. Dahlgren, M. Schordan, and B. White. Toward the Automated Generation of Components from Existing Source Code. Second Workshop on Productivity and Performance in High-End Computing, San Francisco, CA, February 13, 2005. Also available as UCRL-CONF-208403.