

Advances in Software Specification and Verification Introduction to Minitrack

Ann E. Kelley Sobel

Computer Science & Systems Analysis Department
Miami University
Oxford, OH 45056
sobelae@muohio.edu

Richard C. Linger

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213
rlinger@sei.cmu.edu

Complex network systems lie at the heart of critical national infrastructures, including transportation, finance, communication, energy, and healthcare. As societal dependence on these systems grows, the effects of failures become increasingly severe. Recognition of the risks and consequences of failures is driving increasing use of rigorous specification and verification for such systems. The **Advances in Software Specification and Verification** minitrack focuses on research, engineering methods, and case studies that help enable widespread application of rigorous specification and verification methods for complex, large-scale systems. At least three reviewers refereed each of the submitted papers. The six accepted papers are summarized below.

Schedule and market pressures often mitigate against broad use of formal verification methods in commercial software development. However, targeted use of these methods requires traceability from requirements to code to select promising system parts for analysis. In the paper *Experiments in the Use of XML to Enhance Traceability Across Software Engineering Life-Cycle Products*, authors Jim Alves-Foss, Daniel Conte de Leon, and Paul Oman describe methods to link requirements, designs, and code for traceability, change management, and evaluation. They describe implementation of traceability between UML design specifications and implementations in source code using XML-derivative representations.

It is well known that requirements definition and analysis are difficult and error-prone processes, as is the translation of requirements into designs and implementations. In the paper *Monitoring Software Requirements using Instrumented Code*, author William Robinson presents an automatable framework for monitoring requirements of software as it executes, using a combination of assertions and model checking to inform the monitor. The focus of the work is on suspect requirements, and transparency of the software and its environment to the monitor.

Verification through model checking has received substantial acceptance in some application areas, however

use of temporal logic formulae for specifying safety-critical time-bounded constraints remains a difficult problem. In the paper *Specification of Real-Time Properties for UML Models*, authors Stephan Flake and Wolfgang Mueller present extensions to the Object Constraint Language which enables specification of state-related time-bounded constraints in a UML framework.

While substantial improvements has been made, further reduction in errors in complex system development remains a challenging problem. In the paper *Unconventional Software Development*, authors Christopher Landauer and Kirstie Bellman discuss how the Problem Posing Programming Paradigm provides an explicit account of design decisions and the connection between requirements, designs, and code, and how it can be used with simple verification techniques.

A key problem in model checking is to deal with the size of the state spaces to be explored and how the state spaces should be represented. In the paper *Splitting Trees and Partition Refinement in Real-time Model Checking*, authors R. F. Lutje Spelberg and W. J. Toetenel describe a novel approach to representing symbolic state spaces based on splitting trees for partition refinement in model checking real-time systems.

Large-scale network systems exhibit dynamic functionality and connectivity, and complex asynchronous behavior that can exceed engineering capabilities for intellectual control. In the paper *The Flow-Service-Quality Model: Unified Engineering for Large-Scale, Adaptive systems*, authors Alan Hevner, Richard Linger, Ann Sobel, and Gwendolyn Walton characterize user-task Flow Structures and their architecture service traces as dependable representations for specification and design in such systems. Flow Structures preserve compositional properties of abstraction, refinement, and verification, and can be annotated with quality attributes. The Flow-Service-Quality architecture framework provides a flow-based system design and implementation model.