Testing and Certification of Trustworthy Systems
Introduction to Minitrack

Alan R. Hevner
Information Systems & Decision Sciences Dept.
Univ. of South Florida
Tampa, FL 33620
ahevner@coba.usf.edu

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

Gwendolyn Walton
Electrical & Computer Engineering Dept.
Univ. of Central Florida
Orlando, FL 32816
gwalton@mail.ucf.edu

The task of testing and certifying computing system trustworthiness presents significant research challenges. Modern society is increasingly dependent on large-scale systems in its critical infrastructures, including defense, transportation, communication, energy, finance, and healthcare. As a result, the consequences of failures are becoming increasingly severe. These systems exhibit complex networks, highly distributed computing, and challenging requirements for quality and evolvability. Demonstrating the trustworthiness of such systems is a difficult problem indeed. The Testing and Certification of Trustworthy Systems Minitrack focuses on research that will drive widespread use of rigorous testing and certification technologies. Viable topic areas for the minitrack include:

- New techniques for trustworthiness certification
- Testing and certification metrics and measures
- Testing attributes such as security and survivability
- Development and validation of test oracles
- Engineering practices and tools for certification
- Testing in system maintenance and evolution
- Specification methods to support system certification
- Role of correctness verification in system certification
- Industrial case studies in testing and certification

At least three reviewers refereed each of the submitted papers. The Minitrack papers are summarized below.

Demonstrating trustworthiness of automatic code generators can be very difficult. *Certification Support for Automatically Generated Programs,* by Johann Shumann, Bernd Fisher, Mike Whalen, and Jon Whittle, describes an alternate approach to certify the generated code rather than its code generator. Many aspects of trustworthiness can be expressed as properties and checked using Hoare-style verification. For domain-specific specifications, both the code and verification conditions can be automatically generated.

Trustworthiness can be difficult to determine in distributed systems subject to continual evolution. In the paper *Verifying Trustworthiness Requirements in Distributed Systems with Formal Log-file Analysis,* authors Andreas Ulrich, Hesham Hallal, Alex Petrenko, and Sergiy Boroday discuss a method involving instrumentation of systems to generate traces of runtime events that can be converted into system specifications. The specifications can be checked against independently defined trustworthiness requirements by a model checker.

Software testing requires sampling an essentially infinite population of possible executions. Sampling based on anticipated usage permits valid inferences about the quality of the software. To apply this method, usage models must be developed and validated. The models can be used to generate test cases representing expected usage and to reason about software quality given test performance. In *JUMBL: A Tool for Model-based Statistical Testing,* author Stacy Prowell introduces tools for definition and analysis of usage models, generation and execution of test cases, and analysis of test results.

Automatically-generated test cases can be free of cognitive biases often exhibited by human testers. In *Breeding Software Test Cases with Genetic Algorithms,* authors D. Berndt, J. Fisher, L. Johnson, J. Pinglikar, and A. Watkins discuss using genetic algorithms to breed new test cases. An evolving fitness function based on previous test outcomes stored in a data warehouse is employed to evaluate the fitness of generated test cases. Novel search and visualization techniques are introduced.

In *Trusted Software's Holy Grail,* author Jeffrey Voas discusses a grand challenge facing the software quality research community, namely, the ability in the earliest stages of development to accurately define and measure required levels of non-functional attributes, including reliability, availability, fault tolerance, testability, maintainability, performance, safety, and security. The paper focuses on the difficult tasks of defining metrics and measures, as well as methods to compose and evaluate attribute values. The author explores technical and economic tradeoffs in quality certification.

Proceedings of the 36th Hawaii International Conference on System Sciences (HICSS'03)
0-7695-1874-5/03 $17.00 © 2002 IEEE