Exploiting Global Causality in Testing of Distributed and Component-Based Applications

Jun Li, Keith Moore
Hewlett-Packard Laboratories
1501 Page Mill Road
Palo Alto, CA 94304
{junli, kem}@hpl.hp.com

Abstract

A new approach to testing component-based applications is presented, which exploits the practice in component-based systems of generating stub/skeleton modules and using these stubs/skeletons to construct a global perspective of end-to-end causality of inter-component communication. This global causality is captured regardless of reentrancy, callbacks, thread and process boundaries, and unsynchronized clocks.

The captured logs created from the interception points are used to construct a system-wide component interaction model that can expose the inter-component dependencies usually hidden in static analysis of application code. These discovered dependencies are used to create a test boundary for applying a component test harness for that component and the set of dependent components. Similarly, the discovered dependencies can be applied to pruning the available test cases to identify those cases that are best suited to exposing defects when one or more components are changed. A particular advantage of the approach has been the ability to isolate the sequence of events that led up to a crash or a deadlock condition and view the entire system behavior (not just a particular thread's perspective or a linear log of intercepted messages).

1. Summary

Component-based systems (such as CORBA, J2EE, and COM/.NET) have improved the construction of large-scale distributed systems; however, testing these systems remains challenging. Our observation is that large industrial component-based systems are dynamically bound, multi-threaded applications with complex component interactions. The dynamic nature of these systems defeats static analysis approaches that count on a one-to-one mapping between interface and implementation, and the concurrent component interactions defeat simple message interception.

Our approach to component testing leverages previous work on a distributed applications monitoring framework called coSpy [1]. The IDL compiler automatically inserts probes into generated stub and skeleton modules. These probes propagate global causality at runtime without modification to application components. The data collected from these probes reveals the system-wide function caller/callee relationship at the component interface level. We have deployed the monitoring framework on an industrial application (2 million lines of code) that involves 200,000 inter-component interactions.

As a further refinement, we allow the annotation of the component interface definitions with test specification such that the IDL compiler can inject test-related actions into the instrumentation probes. These actions are used to log test-related attributes (such as test suite number or test case number), and also apply test-related control during test execution. Analysis of the logs from these augmented probes automatically links observed test execution behavior with the test specifications.

This testing framework is then used to perform:

- **Collaborator components and their stubbing boundary determination**: to determine which components form tight collaborations by monitoring the running system and use this information to construct a unit component test harness for the current release;
- **Regression test case selection:** to resolve a subset of test cases for retesting when a component changes;
- **Crash/deadlock site pinpointing:** to identify the sequence of events and the set of components/component interactions that lead up to a crash/deadlock during integration or system test.

[1] Jun Li, "Monitoring and Characterization of Component Based Systems with Global Causality Capture," Proceedings of the 23rd International Conference on Distributed Computing Systems, pp. 422-31, May 2003.