

The Advanced Mobile Application Testing Environment

Robert V. Binder
mVerify Corporation
350 N. Orleans St, Suite 950
Chicago, IL 60654 USA

312 881 2054
Bob.Binder@mVerify.com

Senior Member, IEEE

James E. Hanlon
mVerify Corporation
350 N. Orleans St, Suite 950
Chicago, IL 60654 USA

312 881 2654
Jim.Hanlon@mVerify.com

Member, IEEE

Abstract

The Advanced Mobile Application Testing Environment (AMATE) combines model-based test generation and evaluation, controllable RF airlink variation, and a robust standards-based distributed test harness for end-to-end testing of distributed mobile applications. This report summarizes AMATE capabilities.

Problem

When a user of a cell phone or PDA taps a few keys to get current stock quotes or a weather report, a very large and complex collection of application software, operating system software, network equipment, databases, telecommunications equipment, and radio transmission equipment must cooperate to service that request.

End-to-end testing means exercising the complete round trip: from the end user, through the network, to server systems, and back to the end user. The response time and results the end user sees are affected not only by all these piece parts, but by how many other users are active and what they are doing. End user mobility complicates things: signal strength varies with speed of movement and position, airlinks can be dropped or handed-off to equipment with different protocols, and the end user's actual physical location is critical for location-based services.

Mobile application software is no less functionally complex than comparable wired applications. However, many more input combinations and a controllable mobile test harness are necessary to reveal mobile-specific failure modes and achieve adequate testing. The authors developed the AMATE system to demonstrate that these problems can be solved. The U.S. Department of Commerce, National Institute of Standards and Technology, Advanced Technology Program funded this project.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. A-MOST'05, May 15–16, 2005, St. Louis, Missouri, USA. Copyright 2005 ACM 1-59593-115-5/00/0004. \$5.00.

Approach

AMATE uses a model-based strategy to generate realistic test suites for mobile applications.

- Test engineers compose models of external actors, system requirements, and SUT's RF propagation environment.
- The simulator generates scheduled suites of test objects in accordance with a configurable operational profile. These test objects command test agents, interface drivers, and the airlink emulator to achieve input and RF conditions that correspond to each actor's simulated behavior and position within an RF propagation space.
- The test input rate may be varied to achieve any quantifiable pattern of aggregate workload.
- The postconditions specified for each model variant are automatically checked; deviations are reported.

Technical Highlights

The AMATE distributed test environment submits interleaved suites of test objects covering a broad range of actor behavior, actor location, and airlink conditions.

- Model input and editing is done with a simple spreadsheet-like GUI. Models are saved in a relational database with an object-oriented persistence framework.
- The simulator uses abstract interface definitions automatically generated from physical interfaces to generate immediately executable test objects.
- The simulator uses ray tracing over a three dimensional GIS used to generate DALE (Digital AirLink Emulator) commands.
- Distributed test execution is supported with a distributed object architecture using XML, WSDL, SOAP, BEEP, and WAP.
- The DALE uses software-defined radio to achieve real-time variation of airlink quality between each end-point interface and associated base stations.
- The test object framework blends features of the popular extreme programming Incremental Testing Framework, TTCN-3, and the UML 2.0 Testing Profile. Test objects are generated/programmed in [incr Tcl].