This article is part of the IEEE Reliability Society 2009 Annual Technology Report.

Exploratory Testing for Mission Critical, Real-Time, and Embedded systems

Phil Laplante
Email: plaplante@psu.edu

Introduction
First described by Cem Kaner in 1988, exploratory testing is a guided, ad hoc technique that incorporates simultaneous learning, test design, and test execution [1]. Although almost exclusively applied to GUI testing in commercial applications, exploratory testing is exactly the kind of testing needed in real-time, embedded systems to augment traditional, scripted testing approaches.

Traditional software and systems testing involves techniques that rely on scripted, or context-based testing. That is, for each test, the test engineer defines a particular initial state for the system (including the environment state), a set of inputs to the system, and a set of expected outputs for the test. In exploratory testing, however, testing is conducted in an almost ad hoc manner; the tester simply uses the system in certain ways, and then records any anomalies that he encounters. Exploratory testing is "almost ad hoc" because the usage is actually guided by a behavior pattern driven by a particular posture that is adopted by the tester.

Consider the following tourism metaphor: scripted based testing involves a traveler who follows a planned itinerary. Exploratory testing involves a traveler who uses his own instincts and personal agenda to guide his explorations. Continuing with the metaphor, we may have many types of traveler personalities, and these personalities influence the journey of each traveler. Whittaker describes many such journeys in his book [3]. For example, he notes that in big cities the "locals" avoid tourist traps. The software analogy is the set of features avoided by expert users. In the "Historical District" Tour, then, testers deliberately explore those features that the experts would avoid. In the "Hotel District" tour, testers test the functionality of the code that is running behind the scenes when the software is ostensibly at rest.

Although Whittaker enumerates a number of tours that would be appropriate for security testing or embedded systems (the "Saboteur's Tour," and "Seedy District Tour;" think about the implications), there is ample room for other tours specifically related to real-time, and embedded systems. One way to develop a set of useful exploratory tests for such systems is to consider the sources of systems uncertainty [2], and then create tours that uncover these uncertainties. These explorations can be converted into use, and mis-use cases for the purpose of operationalizing the tests, and for regression testing. Consider the following examples.

Environmental explorations
Environmental explorations simulate uncertainty in the environment in which the system is to be operating. These uncertainties may emanate from operating system anomalies, as in the Mars Pathfinder Mission, or from operational domain disturbances, such as a power surge or a violent storm. So there needs to be a family of explorations that uncovers these types of problems. Let us call these kinds of explorations a "Bad Weather Tour". To facilitate such explorations, simulations need to be created that can model any kind of adverse operating environment condition that can be imagined. Past experience can serve as a guide in these situations.

Input explorations
Input uncertainties such as spurious or missed interrupts, anomalous data, and deliberately poisoned data can lead to a cascading series of failures that overload the system. Typically, an abandonment of the single fault assumption is needed to overwhelm any built in fault-tolerance mechanisms. Such tests are called "Murphy's Tour" because anything that could go wrong is made to go wrong.

Output explorations
A software control system can deliver grossly or subtly defective output to the system under control, causing the system response to be perturbed further. The aberrant feedback loop can eventually cause failure in the control system. A set of exploration tests, called “Magic Mystery Tours,” need to simulate every variant of this scenario.

State explorations
Internal faults, such as jumped program counters, due to a number of possible scenarios, can lead to uncertainty of program state that is difficult to diagnose, and nearly impossible to recover from. Because these kinds of failures lead to erratic, almost drunken behavior, we call these explorations “Fear and Loathing in Las Vegas Tours”.

Behavioral explorations
There is a broad class of timing and scheduling problems that are the hallmark of real-time systems. But these kinds of problems are very commonly tested and diagnosed using traditional scripted, context driven testing, so no new exploratory tests are offered at this time.

Language explorations
Many compilers produce code in ways that appear to be non-linear. For example, removing a single line of source code can fundamentally change the compiler output thereafter. The effects of these changes can lead to insidious timing problems, and undesirable side effects. Therefore, a set of explorations are needed to test the compiler, and other systems programs involved in the production of the executable code (debuggers, linkers, loaders, etc.). A series of exploratory tests, called “Shakeout Tours,” are needed to uncover these potential problems.

COTS explorations
These explorations seek to uncover problems in software furnished by third parties such as commercial vendors, or open source software. Traditional testing of these components needs to be done. Because “trust but verify” is the hallmark of this testing, we call these “Reagan’s Tours.”

The Future
Though never given these colorful names, this author and colleagues used these explorations extensively in testing various real-time embedded systems for avionics applications, including the Space Shuttle Inertial Measurement Unit, satellite systems, and other navigation systems. Work is ongoing to classify and socialize these exploratory tests for other kinds of embedded, real-time systems.

References
[1] J. Bach, "Exploratory Testing", in *The Testing Practitioner,* Second Edition, E. van Veenendaal Ed., Den Bosch: UTN Publishers, 2004, pp. 253-265.
[2] Phillip Laplante, “Coping with the Certainty of Uncertainty in Real-Time Systems,” *Instrumentation and Measurement*, vol. 7, no. 4, December 2004, pp. 44-50.
[3] James A. Whittaker, *Exploratory Software Testing: Tips, Tricks, Tours, and Techniques to Guide Test Design*, Addison Wesley, 2009