Adoption of Test Driven Development and Continuous Integration for the Development of the Trick Simulation Toolkit

John M. Penn
Trick Development Lead/ Senior Engineer
L-3 STRATIS Division
NASA JSC Engineering Directorate
Software, Robotics, and Simulation Division
Simulation and Graphics Branch

This paper describes the adoption of a Test Driven Development approach and a Continuous Integration System in the development of the Trick Simulation Toolkit, a generic simulation development environment for creating high fidelity training and engineering simulations at the NASA/Johnson Space Center and many other NASA facilities.

It describes what was learned and the significant benefits seen, such as fast, thorough, and clear test feedback every time code is checked-in to the code repository. It also describes a system that encourages development of code that is much more flexible, maintainable, and reliable.

The Trick Simulation Toolkit development environment provides a common architecture for user-defined simulations. Trick builds executable simulations using user-supplied simulation-definition files (S_define) and user supplied "model code". For each Trick-based simulation, Trick automatically provides job scheduling, checkpoint / restore, data-recording, interactive variable manipulation (variable server), and an input-processor. Also included are tools for plotting recorded data and various other supporting tools and libraries.

Trick is written in C/C++ and Java and supports both Linux and MacOSX.

Prior to adopting this new development approach, Trick testing consisted primarily of running a few large simulations, with the hope that their complexity and scale would exercise most of Trick's code and expose any recently introduced bugs. Unsurprising, this approach yielded inconsistent results. It was obvious that a more systematic, thorough approach was required.

After seeing examples of some Java-based projects that used the JUnit test framework, similar test frameworks for C and C++ were sought. Several were found, all clearly inspired by JUnit. Googletest, a freely available Open source testing framework, was selected as the most appropriate and capable.

The new approach was implemented while rewriting the Trick memory management component, to eliminate a fundamental design flaw. The benefits became obvious almost immediately, not just in the correctness of the individual functions and classes but also in the correctness and flexibility being added to the overall design. Creating code to be testable, and testing as it was created resulted not only in better working code, but also in better-organized, flexible, and readable (i.e., articulate) code. This was, in essence the Test-driven development (TDD) methodology created by Kent Beck. Seeing the benefits of Test Driven Development, other Trick components were refactored to make them more testable and tests were designed and implemented for them.

Jenkins (originally named Hudson) brought significant automation to our development process. Jenkins is a freely available (MIT License), easy to use, Java application that initiates and monitors repeated job executions. Its primary use is as a continuous integration system, for automated building and testing of software projects. Jenkins schedules a build of the Trick distribution, on each of the different machine architecture/operating system combinations that is supported, every 15 minutes (if our Subversion source code repository has been updated). After a successful build, it runs the entire unit test suite and then a collection of Trick based test simulations. Jenkins also provides both a summary and a detailed status result.

Within 15 minutes of any source code check in, the state of the entire code base is determined. If a problem is detected, Jenkins provides the details that allow the problem to be quickly identified.

The combination of Test Driven Development and a Continuous Software Integration System has given us much more confidence in our code. Not only has it made Trick more reliable, flexible, maintainable, it’s made Trick development more efficient and more pleasant.