System of Systems Engineering Collaborators Information Exchange (SoSECIE)

Tuesday, July 31, 2012
11:00 a.m. to Noon EDT

Adaptable System Integration on Multiple Platforms

Mr. Timothy F. Palmer
Ms. Samantha N. Swafford
Georgia Tech Research Institute

Abstract
Integrating a system on multiple platforms often creates challenges related to software design and system-level testing. Effects on system interfaces, software, test procedures, test tools and mission-specific data must be considered when adding support for a new platform to an existing system. When such integration is performed properly, it is possible for other platforms to benefit from one platform's improvements. This presentation focuses on an adaptable system integration effort on multiple platforms in a manner designed to minimize changes to the integrated system.

Coding practices provide various methods for integrating a system on multiple platforms. Generally, the more common software between platforms, the more efficient development is across multiple platforms. Using a Configuration Management (CM) tool that allows for multiple branches of software can aid in tracking different software for multiple platforms. To add a platform, a branch is made from the baseline software to be developed separately. Upon successfully integrating a new platform, the branch is merged back into the mainline to reduce the number of variations of the software. Merging branches may also add functionality to the mainline, increasing the capabilities of all of the supported platforms. Even though merging software into one main source may increase software complexity, it can decrease effort spent on tasks such as issue tracking and documentation.

Components such as Radios or Radars may not be common among two platforms; however they may provide similar types of data. If two platforms have similar components that perform the same task, then software should be designed to minimize the impact of the platform differences. When two platforms have the same component, then system software should be uniform across platforms.

A software architecture is employed to extract common data from different sources and place the data into a common data structure. This structure is then fed into the main software which allows different platforms to use the same main algorithms with minimal impact to the software. This architecture allows software to more easily adapt to multiple platforms because the format of received data is extracted from the main software.

Common requirements and designs between the system and software levels promote sharing capabilities between platforms as well as highlight the differences among platforms. System variations may be necessary due to platform constraints; however, the impact of differences is diminished using this streamlined system redesign.

With a common system design and software baseline in place, verification of the software baseline design and its branches is necessary. A cohesive system issue tracking process is vital to tracing errors to different baselines of software. An online database is in place to track issues for each software branch. An issue that exists on multiple branches requires corresponding entries in the database linking the issue to each related branch. The more common software that exists among platforms, the smaller the database is as there are fewer instances of duplicate issues.

Like software, keeping tests similar for each platform allows for features to apply to multiple platforms. Test Suites are constructed for each platform and are composed of several test procedures that are either similar across platforms or unique to a specific platform. Test procedures are composed of modular test cases, each focusing on a specific subtask such as User Interface, Error Status or Component Interoperability. When multiple platforms share a component, test procedure changes are minimal between platforms. This reduces time devoted to test development and maintenance of existing functionalities and allows for more in-depth testing of new system functionality. For test cases that are common among platforms, test step verification to requirements mapping is completed once for the platforms involved. This ensures more complete test coverage across the various platforms for the common components.

The test files created and maintained for each platform's Test Suite are thoughtfully constructed and configuration managed. Similarities in mission data propagate to simulation tool test files across platforms aiding in reducing file maintenance costs, test file development efforts and issue investigations. Division of test files by applicable test tool eases CM efforts for multiple platform files and allows for more fluid transitions as new versions of the test tools are released. For platform specific and even software version specific test files, the files are configuration managed by software branch. This proves to be invaluable as hardware and software re-designs of the system necessitate vastly different test files inputs.

Biography

Tim Palmer has been a Research Engineer for GTRI for 5 years and has served in various roles from embedded software engineer to test application developer to systems engineer. For the past 3 years he has been the Lead Software Engineer for different embedded applications within the ALQ-213 countermeasures set that controls electronic warfare systems. In 2007 he graduated from Mercer University with a MSE and BSE in Computer Engineering.

Sam Swafford has been a research engineer for GTRI for 4 years and has served as the Lead Test Engineer for ALQ-213 embedded software and support tools. For the past 3 years, she has been an Associate Branch Head in the Electronic Systems Integration Division. She brings 7 years of test experience, working for Northrop Grumman before GTRI. In 2012 she graduated with her MSE in Biomedical Engineering from Georgia Tech and she graduated with her BSE in Electrical Engineering from Georgia Tech in 2004.