

The IV&V Program is hosting its Annual Workshop on Validation and Verification September 15-17, 2010. All engineers and scientists are invited to participate in discussions with the goal of generating solutions to discovered challenges. If you are interested in leading a particular discussion or presenting a paper/discussion slides, contact Marcus.S.Fisher@nasa.gov by August 9th to be considered for this year's agenda.

Basic set of working discussions to cover

• **Validating Software Requirements**

- Techniques that have been evaluated and/or advanced that would allow us to determine whether or not a set of software requirements are the right requirements for the software.
- Approaches that are based on using references as the basis for determining whether a set of requirements are the right ones, approaches that rely on comparisons among the software requirements themselves, etc ...
- Approaches that have been evaluated and/or advanced that will allow engineers to quickly find ambiguous or inconsistent requirements
- Approaches that are based on engineers providing proof of concept (assurance cases) or evidence records that justify that the specified software requirements are the right ones
- Approaches that incorporate formal techniques to evaluate sets of software requirements

• **Validating Developer's Testing**

- Techniques to determine whether software testing is adequately evaluating the software requirements and/or the desired system behaviors
- Techniques that enable engineers to extract information and/or behaviors from test scripts used to evaluate software requirements

• **Verifying Software Architectures**

- Techniques that enable engineers to evaluate the global organization of the system software as a composition of components, relationships among software components, global control structures and communication strategies, and strategies to handle failures and respond intelligently to them with the goal of determining whether or not a specific software architecture will satisfy the set of software requirements
- Techniques that enable engineers to estimate performance of an architecture in order to identify potential risks

- **Verifying Software Designs**

- Techniques that enable engineers to evaluate the messaging/protocols, state transitions, control flow, internal data flows, and structure (data, object, procedure) in order to show whether a software design can satisfy the software requirements and/or desired system behaviors
- Modeling and simulation techniques to evaluate software designs against a set of software requirements and/or set of desired system properties

- **Verifying Implementation**

- Techniques to enable efficient correlations to be established between the source code and the requirements, as well as to the desired system behaviors
- Techniques that enable engineers to find common programming problems and determining whether they will cause problems during software execution
- Techniques to evaluate the results of testing in order to determine if the software requirements are being met by the chosen implementation
- Techniques to exercise the code in an intelligent manner to evaluate requirements and/or desired system behaviors
- Intelligent testing and monitoring techniques to enable efficient testing of software
- Techniques to enable the formal analysis of source code or the execution of code and exhaustively determine whether all of the requirements are being met
- Modeling and simulation techniques for evaluating system software

- **Managing IV&V**

- Techniques to help identify the desired software behaviors
- Techniques that enable the criticality or risk of a behavior or software component to be better understood
- Scheduling techniques and latest tools to make the management of a schedule more efficient

Opportunistic Discussions:

- Working discussions and tracks regarding the common issues seen across the Agency, as well as in other organizations, and potential solutions/strategies/analytic techniques to stop these issues from occurring in the future
- Working discussions regarding Project-based IV&V vs IV&V based on common capabilities being built by/for the Agency (e.g. should IV&V have been done on CFE/CFS when it was built as a capability for GSFC or should IV&V wait for this functionality to be incorporated into the development projects?) - lessons learned and challenges of either approach
- Working discussions on performing IV&V using immature or less-descriptive-than-needed artifacts (e.g. verifying software designs using developer's software designs captured in power point charts)

Domain Specific topics slated for discussion at the workshop

- Intelligent software systems and techniques to provide software assurance
- Intelligent fault handling and intelligent environment control to sustain life
- SWARM Technologies and adaptive systems in general: what are they?, what are their significant differences from typical software systems?, what are our challenges in providing assurance?
- Spacecraft attitude control and software analysis strategies
- Embedded software and analysis on code written for specific microcontrollers

1. Presentations and working discussions on operating system and platform abstraction techniques (board support packages, etc) and how, in many cases, more than 90% of the software can be portable C while the specific code for each microcontroller and each board (one microcontroller can be on many different kinds of boards) is often less than 10% of the overall effort

2. Configurable build environments and the need for properly ensuring separation of layers in the build system.

3. Training and knowledge sharing sessions regarding embedded software and the necessity for code analysts to understand the embedded code nuances when statically analyzing it and when preparing unit tests/integration tests