### NASA IV&V Program

Independent Test Capability

Concept of Operations – 6.5.1.2

Version 2.0

August 20, 2010

<table>
<thead>
<tr>
<th>Author</th>
<th>ITC Lead</th>
<th>Original Creation Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mike Wise</td>
<td>Justin Morris</td>
<td>08/20/2010</td>
</tr>
</tbody>
</table>
### VERSION HISTORY

<table>
<thead>
<tr>
<th>Version Number</th>
<th>Author</th>
<th>Revision Date</th>
<th>Reason for Revision</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>Mike Wise, Steven Seeger</td>
<td>10-31-09</td>
<td>Initial Draft</td>
</tr>
<tr>
<td>1.1</td>
<td>Justin Morris</td>
<td>03-10-10</td>
<td>Minor revisions made to address Program org changes (throughout), operational scenarios updated (Section 6), and roles within IV&amp;V Program (3.3)</td>
</tr>
<tr>
<td>2.0</td>
<td>Mike Wise</td>
<td>08-20-10</td>
<td>Formatted to ITC team document template. Updated to reflect decisions made during design.</td>
</tr>
</tbody>
</table>
## REFERENCE DOCUMENTS

<table>
<thead>
<tr>
<th>Document Number</th>
<th>Document Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.5.1.1</td>
<td>Vision and Scope Document for NASA IV&amp;V Test Capability, Version 2.0</td>
</tr>
</tbody>
</table>
# TABLE OF CONTENTS

Version History .............................................................................................................................. 2

Reference Documents .................................................................................................................. 3

Table of Contents .......................................................................................................................... 4

Table of Tables ............................................................................................................................. 6

Table of Figures ........................................................................................................................... 6

1 Introduction ........................................................................................................................ 7
  1.1 Document Overview ....................................................................................................... 7

2 Current Situation ................................................................................................................ 8
  2.1 Background .................................................................................................................... 8
  2.2 Description of Current Situation ...................................................................................... 8

3 Proposed system (Justification and Description) ............................................................. 10
  3.1 Justification ................................................................................................................... 10
  3.2 Description .................................................................................................................... 11
  3.3 Priorities ........................................................................................................................ 11

4 Proposed System (description) ........................................................................................ 12
  4.1 Background, Objectives, and Scope ............................................................................ 12
  4.2 Operational Policies and Constraints ........................................................................... 12
  4.3 Description of the Proposed System or Situation ......................................................... 12
    4.3.1 Operational Environment and Its Characteristics .................................................. 12
    4.3.2 Major System Components and Interfaces ........................................................... 14
    4.3.3 Interfaces to External Systems or Procedures ....................................................... 17
    4.3.4 Capabilities, Functions, and Features of the System, ........................................... 17
    4.3.5 TYPICAL CONTROL AND DATA FLOW .............................................................. 18
    4.3.6 Performance characteristics .................................................................................. 19
  4.4 Proposed Modes of Operation ...................................................................................... 19
  4.5 Anticipated Users and Stakeholders ............................................................................ 20
  4.6 Proposed Support Environment ................................................................................... 24

5 Operational Scenario ....................................................................................................... 25

6 Summary of Impacts ........................................................................................................ 27
6.1 Operational Impacts ................................................................. 27
6.2 Organizational Impacts ......................................................... 27
6.3 Impacts During Development ................................................ 28
TABLE OF TABLES

Table 1: Current Analyses Completed .......................................................................................... 9
Table 2: Advantages and Disadvantages of Static and Dynamic Analysis ................................. 10
Table 3: Users and Stakeholders ............................................................................................... 21
Table 4: ITC Team Roles and Responsibilities ......................................................................... 23

TABLE OF FIGURES

Figure 1: ITC Components and Interfaces .................................................................................. 14
Figure 2: Core Test Capability Architecture .............................................................................. 15
Figure 3: Typical Data Flow ........................................................................................................ 18
Figure 4: Rudimentary Testing .................................................................................................... 19
Figure 5: Organizational Responsibilities ................................................................................... 28
1 INTRODUCTION

The purpose of this document is to encapsulate how the Independent Test Capability (ITC) will be used by the NASA IV&V Program. The need for the ITC is driven by the desire to perform dynamic testing on software in order to provide more complete coverage of tests and to better analyze behavioral and timing requirements. This document follows the Vision and Scope document which attempted to answer what was needed at a high level, and now the Concept of Operations will attempt to answer how the ITC will be used.

1.1 DOCUMENT OVERVIEW

This document is divided into 3 main sections. The background section briefly describes the current situation and scope of V&V work being done. The justification and description section explain how the proposed solution will expand the capabilities of the IV&V Program and give a brief explanation of the proposed solution. The proposed solution section gives a more detailed description of the proposed system including the proposed architecture given at a high level, and examples of data flow and roles and responsibilities of the system stakeholders. The final sections give an example operational scenario and describe organizational impacts of the proposed system.
2 CURRENT SITUATION

2.1 BACKGROUND

Current IV&V Program verification activities are limited to static code analysis, manual code analysis, and test result analysis. The NASA IV&V Program does not have a Program-wide system capable of performing dynamic software analysis. Although dynamic software analysis has been done in the past, in these instances, the configuration was unique to the project for which the tests were going to be performed. The ITC will provide a more robust test capability that can be efficiently implemented by all projects in the IV&V Program.

2.2 DESCRIPTION OF CURRENT SITUATION

The IV&V Program does not currently have a capability to completely confirm whether or not an implemented system behaves as expected, that can be applied to all projects in the Program. Completeness cannot be achieved because statically assessing system software does not provide a complete perspective for how a system is going to operate. Therefore, it does not provide a complete understanding of how an implemented system is behaving. Static analysis will reveal that requirements have been met, but cannot uncover problems in the interactions in which multiple requirements would be encompassed. The static analysis won't uncover run-time issues involving interactions between separately threaded processes.
### Table 1: Current Analyses Completed

|---------|----------------------|----------------------|----------------------|
| 1. IV&V receives Build N from Project  
2. Performs line-by-line walkthrough of code  
3. Compare differences with previous builds | 1. IV&V receives Build N from Project  
2. Static analyzer tools (Klocwork, Flexelint, or Code Sonar) are exercised  
3. Examine output results for vulnerabilities | 1. IV&V receives test results from Project  
2. Examine actual test results against expected test results  
3. Identify any discrepancies  
4. Identify independent test scenarios |
| Value | Provides an additional set of eyes on software components  
Verifies that single requirements have been met  
Can be applied early in the lifecycle | Provides set of rules to execute against code base to identify critical bugs such as memory leaks, buffer overflows, and NULL pointer dereferences  
Can be applied early in the lifecycle | Confirms intended system behavior  
Provides input into test scenario development |
| Cons | Burdensome task that requires many resources  
Quality of work products directly related to individual experience and skills | Significant amount of effort required to sort through analyzer output results  
False positives  
Does not provide run-time vulnerabilities  
Analyzers are only as good as the rules they are using | Solely dependent developer executed tests  
Lack of insight into test environment tools and configurations |
3 PROPOSED SYSTEM (JUSTIFICATION AND DESCRIPTION)

3.1 JUSTIFICATION

The IV&V Technical Framework calls for a assuring and ensuring system implementation. This cannot be thoroughly exercised without an environment in which the software can be run and tested against the input test scenarios through previous validation and verification work.

Verification and Validation completeness cannot be achieved solely by statically analyzing system software. The purpose for developing an independent testing capability is to provide a simulation and test environment that enables the dynamic analysis of system software through the integration of simulation tools, models, and test articles. The development of a reconfigurable independent test capability will enable a common core test environment for use Program-wide that is readily adaptable for project specific needs.

Table 2 presents the advantages and disadvantages of static and dynamic analysis. Since the advantages and disadvantages complement each other, a hybrid approach can be utilized to provide the most complete analysis. In particular, the addition of a dynamic testing capability provides a mechanism to: find run-time vulnerabilities, increase testing flexibility, uncover vulnerabilities that were flagged as false negatives during static analysis, and validate static analysis findings. Currently used automated tools such as Klocwork, CodeSonar, and Flexilent provide the IV&V Program with a static analysis capability, and the ITC will provide the dynamic analysis capability.

<table>
<thead>
<tr>
<th>Static Analysis</th>
<th>Disadvantages</th>
<th>Dynamic Analysis</th>
<th>Disadvantages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Advantages</td>
<td></td>
<td>Advantages</td>
<td></td>
</tr>
<tr>
<td>Finds weaknesses in exact location</td>
<td>Can be time consuming</td>
<td>Finds run-time vulnerabilities</td>
<td>Can provide false sense of security that everything has been addressed</td>
</tr>
<tr>
<td>Allows quicker turnaround for fixes</td>
<td>False Positives and False Negatives</td>
<td>Provide increased flexibility of what to look for</td>
<td>False Positives and False Negatives</td>
</tr>
<tr>
<td>Finds errors earlier in lifecycle</td>
<td>Requires trained personnel</td>
<td>Identifies vulnerabilities that may have been false negatives in static analyses</td>
<td>Requires trained personnel</td>
</tr>
<tr>
<td>Automated Tools</td>
<td>Do not provide runtime vulnerabilities</td>
<td>Validation of Static Analysis Findings</td>
<td>More difficult to trace vulnerability back to the exact location in the code.</td>
</tr>
<tr>
<td>Relatively fast</td>
<td>Automated Tools</td>
<td>Can provide false sense of security</td>
<td>Can scan all of code</td>
</tr>
<tr>
<td>Can scan all of code</td>
<td>Only as good as rules they are using</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The ITC will utilize input test scenarios derived from project artifacts, domain expertise, and other V&V artifacts developed by different projects within the Program to provide a value-added capability. This capability will provide a means to identify software issues that are difficult, if not impossible, to discover in currently used analysis processes. Specifically, the ITC will provide the framework for setting up and executing developer-generated tests and/or IV&V-generated tests.
3.2 DESCRIPTION

The primary capability that will be added will be the ability to dynamically test software in an environment that mimics the intended target environment. While not a new concept, efforts in the past have been project specific implementations that weren’t useful to other projects. The ITC will provide a core system and common interface to the analysts for completing the verification process on any project.

The main changes and challenges will come from being able to effectively run and stimulate executable software builds in an emulated environment. To be successful and to make it adaptable to all projects a core system must be developed that will remain relatively constant for the analysts. The core system will need to interface with various emulators and software systems, and will also need to accept the artifacts from the previous validation and verification processes as inputs to run tests against the software.

Additionally, the ITC should be developed with future expansion in mind. This could include distributed networked simulations internally, and running test cases from this location in a test bench that resides at another location.

3.3 PRIORITIES

The key high level capabilities and the priorities assigned to them are listed below. Features marked as essential are key features that provide the most value to the IV&V Program and are features that distinguish the ITC from other isolated solutions that have been used in the past. Desirable features are those that greatly enhance the solution, while optional features aren’t totally necessary for the ITC to be a success (but could be useful).

- The ability to run and stimulate executable software builds. Essential.
- Common user interface across projects. Essential
- Ability to save data inputs and outputs. Essential
- Ability to save and load configuration files for repeating tests. Essential
- Common library of simulation models for test environment fidelity. Desirable
- Distributed networking capability. Essential
- Ability to drive software test rigs at other locations. Optional
4 PROPOSED SYSTEM (DESCRIPTION)

4.1 BACKGROUND, OBJECTIVES, AND SCOPE

The goal of the ITC is to provide the means to do dynamic testing in the NASA IV&V Program. To more fully verify and validate the software system, it is important that the IV&V Program develop a full independent test capability in which current developer software builds can be loaded into a test environment that mimics the target platform. This will allow tests to be executed against the software as it runs in the test environment. This will provide the ability to independently verify behavior implementations, as well as the requirement implementation that is currently done. In the past, verification test rigs have been implemented, but they were specific to the project and didn’t allow for re-use across other projects. The ITC will provide a core system that is independent of specific projects, but is readily adaptable to meet the needs of any project in the Program. The ITC will be capable of accepting disparate test inputs, as well as being able to interface with multiple emulators and/or simulation packages.

4.2 OPERATIONAL POLICIES AND CONSTRAINTS

Being a new capability to the NASA IV&V Program, there aren’t any operational policies in place regarding the ITC specifically, but it should comply with Program-wide policies involving IT security and protection and handling of data. It’s difficult to quantify the challenges, but there will be some e.g. how quickly a useful (but not final) version of the ITC can be stood up, how quickly the analysts can get trained, etc.. There will need to be some constraints placed on access to the system with regards to safeguarding information, and there will also be some operational constraints regarding the number of tests that can be performed at once, but it’s too early to define the specifics. Other constraints will include licensing fees for required software packages, time and budget constraints in meeting project needs, and the ability to efficiently acquire any and all information necessary to set up the ITC configuration for a specific project.

4.3 DESCRIPTION OF THE PROPOSED SYSTEM OR SITUATION

4.3.1 OPERATIONAL ENVIRONMENT AND ITS CHARACTERISTICS

The primary objective of the ITC is to be able to perform dynamic software analysis; therefore, the primary requirement will be to provide an interface to a system under test running in a simulated environment that is indicative of the deployed system environment. In some cases, this may mean that the software is running on real hardware, but it should be sufficient in most cases to run on emulators. The ITC needs to be capable of running in real-time in order to more closely duplicate the operating environment of the software under test, but it also needs to be able to capture event driven issues. When running in real-time, the ITC needs the capability to capture performance oriented data, such as timing requirements and bandwidth usage.

The ITC should provide an intuitive interface to the analyst that will capture the inputs and outputs as the test is running, as well as a means to capture events of interest such as a breach of data thresholds. This will enable post test analysis of the data generated, and will also
provide a baseline for regression testing once issues have been identified and a corrected version of the software is provided. There will need to be significant data storage capabilities, and an efficient means to retrieve the desired data, as well as data reduction tools to get to the pertinent data or trends as efficiently as possible. The ITC should provide a means for the analyst to visualize and/or graph signals and simple interactions or relationships between them. At any point during the test, it should be possible to stop or pause the test, reset the input data, and restart the test without the need to restart the ITC.

The ITC should be configurable to work on a distributed network with an interface on the analysts PC. This will allow for the handling of multiple projects operating at once, and will eliminate the need for a test area and additional PC’s used specifically for the ITC. A distributed network configuration will also provide maximum flexibility and can facilitate growth that could include running tests on systems that reside at other locations. However, the ITC should also be flexible enough to operate in a stand-alone environment in which everything needed to perform a test resides and runs from a single PC or laptop.

Simulations, models, or configurations that need to be developed to perform the dynamic software analysis should be located in a central repository read-accessible by all analysts to facilitate reuse and eliminate duplicate development efforts. The repository should also be write-accessible for the analysts to upload test configurations, and data from tests to safeguard against data loss. Depending on the requirements of the developers and the emulators used to run the software under test, it may be necessary to have multiple servers with differing operating systems.
4.3.2 MAJOR SYSTEM COMPONENTS AND INTERFACES

The ITC will consist of several different components and interfaces (Figure 1: ITC Components and Interfaces). The individual components and the interfaces between them will be discussed in the following sections.

![Figure 1: ITC Components and Interfaces](image)

**Core Test Capability Architecture**

The Core Test Capability Architecture (CTCA) will consist of the core capabilities of the system that will remain consistent across all project implementations. The interface between the CTCA and the Software/System Under Test will be adapted for project needs, and there may be specific project needs that will drive modifications of the user interface. The CTCA will include the controls to start, stop, reset and pause the tests, the communication middleware between all of the various components, and the various models and stored configuration files used to run tests.
The central server is primarily responsible for managing network resources, users, and system control and system monitoring services. The central server makes use of the system monitoring to keep track of user applications and components. If a user application goes down, the server informs interested components that the user acquired resources should be released, and maintains a snapshot of the entire software system state.

The system control is responsible for providing components and ensuring that components are operational when needed. System control upon start up will attempt to register their identity with the central server. On successful registration, it will supply the central server with a list of all provided capabilities. The system control will monitor any acquired resources, inform owners if the resource exits unexpectedly, and notify the central server that a process is unavailable.

The system monitoring provides a mechanism to loosely monitor the execution status of ITC components and is composed of a heartbeat server and a client library. The system monitoring service be used by components as an additional way to track the status of other components on the network, and will consist of a client and server. The server will track all registered components on the network and send notifications when a component’s status changes. The client will be used by components to interact with the heartbeat server.
Software/System Under Test (SUT)

As the name implies, the SUT will be the software or system under test. This will consist of everything that is needed to get the test article running in the test environment. In most cases, this will consist of executable flight software that is loaded on a host emulator or simulation system (SoftSim, Simics, Matlab). In other cases, this might involve actual hardware or that is housed at another location. The SUT will always be project specific of course, but the interface between the CTCA and the SUT will be specific to the simulation system being used. This will result in minimal interface development as more projects using the same simulation system (i.e. Simics, or SoftSim) are integrated into the ITC.

Data Storage

The Data Storage component is straightforward. It will provide a means for the analyst to capture the relevant data for the tests that are being conducted. It will also provide a repository for other test articles and configuration files, which should maximize re-use of articles across projects. The configuration files will also enable analysts to easily recreate a test environment and re-run a test at any time in the future. The Data Storage component itself will remain static across all projects with little to no impact on the interfaces between it and the User Interface or CTCA for various projects.

ITC User Interface

The User Interface will provide an analyst with a common control and monitoring interface to be used across all IV&V projects. It will provide the basic mechanism to configure and control a simulation, as well as the interface to monitor and retrieve test data and previous configurations from the Data Storage component. There will be some project specific GUI components that will be developed, but the design and development of these should account for re-use in the future. Therefore, as the number of projects grows, so will the utility and robustness of the User Interface.

CTCA to SUT Interface

The interface between the SUT and the Core Test Capability Architecture (CTCA) provides the ability for signals to be interjected into the SUT and behaviors monitored. Regardless of how the test cases are formed, the input test data will need to be formatted in the correct form for the SUT to process the data. For fault handling analysis, it will be desirable to interject improperly formed messages. In addition to the test inputs, the CTCA will need to be able to capture the resulting outputs from the SUT. These inputs and outputs will be operated in real-time, or in event-driven mode (see Proposed Modes of Operation).

SUT to Data Storage Interface

The next interface will be between the SUT and the data logging capability to capture the input and output data. There won’t be a direct interface to the SUT and the data storage; instead the data will pass through the CTCA before being logged. The CTCA will need to mimic the environment that the SUT is intended for as closely as possible. Therefore, the input and output data for the SUT will be captured by the CTCA and then written to the data storage. This eliminates the need the “touch” the SUT anymore than is absolutely necessary to conduct the tests.
Analyst to ITC User Interface

The interface to the analyst should provide a means to stop, start, reset, and pause a test, and control the inputs to the software under test. Before the inputs get passed to the SUT, they will go into the CTCA where they will be translated from the test case or a user interface into the message format that the SUT is expecting. For structured messages, there should be an interface for the analyst to modify any of the parameters of a message before the message is sent. For real-time or scripted tests, this interface will provide a means for the analyst to input the test cases which will be processed and formed into the appropriate message structure.

Analyst to Data Storage Interface

After the tests have been run and the data logged, there will need to be an interface for the analyst to examine the test artifacts. The analyst should have various tools at their disposal for data mining and analysis activities once the data has been retrieved from the repository. There should also be a means for the analyst to save their test configuration and test cases in the data storage repository so that the exact tests can be rerun in the future to examine new issues that might arise, or to perform regression testing on new software deliveries.

4.3.3 INTERFACES TO EXTERNAL SYSTEMS OR PROCEDURES

It should be a goal in the design of the ITC to have as little impact on the work of the analysts as possible. The artifacts from the current V&V processes need to be leveraged and used as inputs to the ITC. The transformation of those artifacts into test inputs would fall on the shoulders of the analysts; ITC administrators will configure the ITC for use. The results from the analysts’ work will be fed into and translated in the ITC and sent to the software under test. The resulting outputs from the software under test will be parsed and sent back out to the user interface for the analyst, but will also be logged along with the inputs and saved into data storage for post-test analysis.

4.3.4 CAPABILITIES, FUNCTIONS, AND FEATURES OF THE SYSTEM, (Including Types of Transactions Processed)

The ITC will need to be able to integrate software with different emulators or simulation systems. The CTCA will provide the backbone for the communication to these different systems, but separate interfaces will need to exist between the SUT and the CTCA for each type of emulator or simulation system used.

The ITC will need to be able to accept inputs to run tests and capture events (this data flow is captured in 4.3.5 below). As part of the V&V process, the analysts will generate input test scenarios and identify areas of test. These artifacts will be used to generate test inputs for dynamic analysis. A translator will be developed within the CTCA that will take the inputs and format them into messages that can be read by the SUT before passing them along. The user interface should also be able to handle individual signal overrides that might be independent of a test case, but will help the analyst understand system behaviors.
To run the tests, the analyst will be able to execute the software in real-time, or be able to run the system in an event driven mode, depending on the capabilities of the emulator or simulation system that the test article is running on. The analyst will also be able to view data during the test, and control the running of the simulation with run, reset, stop, and pause commands.

4.3.5 TYPICAL CONTROL AND DATA FLOW

Figure 3: Typical Data Flow

The primary data flow from the test input through output to the analyst is depicted above in Figure 3: Typical Data Flow. The analyst provides the input into the system (either manually with overrides or through scripting) through the GUI where the data is translated into a form that can be communicated across the CTCA communications middleware to an emulator bridge that provides the input to the SUT. In the data transfer stage, the data could be communicated on a single PC, or this data could be communicated across a network to another PC or server where the SUT is running. The emulator bridge provides the interface between the CTCA communications middleware and the system that the SUT is running under. This interface should not be unique across products, but will be unique to the emulator or simulation system that is housing the SUT.

The only interface to the SUT is through the emulator bridge interface and the CTCA communications middleware. The CTCA will capture the inputs and outputs that are relevant to test being executed and write this data out to the data storage component and to the user interface. The analyst will be able to monitor this data in real-time through the user interface,
and also access it through the data storage for post analysis. Any components necessary for the test will be wrapped in component wrappers and send and receive data with other components in the simulation across the CTCA communications middleware. The data storage is also used to save and restore simulations and simulation components through the GUI.

4.3.6 PERFORMANCE CHARACTERISTICS

There are no known performance characteristics except at a very high level; the ITC requires the necessary throughput to be able to run at real-time when desired.

4.4 PROPOSED MODES OF OPERATION

The primary modes of operation for the ITC will be in support of real-time and event-based testing. In order for the ITC to uncover issues with timing or bandwidth in the software under test, it must be able to handle all requisite data efficiently enough to run at real-time. This type of testing will most likely involve external simulation models to produce all of the messages that would be expected by the system under test. To avoid disrupting the flow of data, these tests will need to be scripted and run without the interjection of signals from the analyst during run time.

With event based testing, the software under test will be given inputs that will trigger a response. These inputs can be from scripted test cases, but the ITC should also provide the ability for the analyst to interject signals at any time during the test. This mode can be as simple as running the software and providing manually driven inputs to the SUT as shown in Figure 4: Rudimentary Testing.

![Figure 4: Rudimentary Testing](image)

While the primary testing can be broken down into real-time and event-based testing, the ITC will also need to run in a “playback” mode in which the data from previous tests can be run again and the behaviors examined. The ITC will also need to support a data-analysis mode in which the data from previous tests can be examined using various tools available to the analysts.
4.5 ANTICIPATED USERS AND STAKEHOLDERS

The users for the ITC will be the IV&V project analysts and software developers. The primary stakeholders will be project leads, and NASA IV&V management. The ITC will provide the capability to perform dynamic software analysis, which will affect all parties involved in verification at the NASA IV&V Program. The ITC team will be responsible for configuring the ITC for dynamic analysis and performing a subset of developer tests that will serve as an informal validation of the simulation environment.

Additional users and stakeholders include IV&V project analysts, project leads, network operations, IV&V management, software development teams, project point of contacts. Each of these roles are defined in more detail in Table 3: Users and Stakeholders. Without the ITC, the verification analysis that can be done at the NASA IV&V Program will remain limited to static analysis, except for isolated instances. While this may be comfortable for the analysts since it is familiar, it doesn’t provide the developers and project leads with feedback pertaining to behavioral and/or timing issues.

For completeness, Table 4: ITC Team Roles and Responsibilities includes a summary of the ITC team internal organization. Complete details on the team organization and processes can be found in the ITC Software Development Plan.
## Table 3: Users and Stakeholders

<table>
<thead>
<tr>
<th>User / Stakeholder</th>
<th>User Type(s)</th>
<th>User Description and Role</th>
</tr>
</thead>
<tbody>
<tr>
<td>IV&amp;V Project Manager</td>
<td>Customer</td>
<td>The PM is a civil service employee that is responsible for the overall leadership and direction of the IV&amp;V efforts. The PM is responsible for establishing the goals and objectives of the IV&amp;V efforts, performing the RBA efforts, performing project management/financial management, project tracking and oversight and risk management of the IV&amp;V efforts. The PM is to ensure that the ITC efforts are included in the IV&amp;V Project Execution Plan (IPEP) and communicated to the Project POC. The Project Manager is also responsible for identifying appropriate project points of contact for the ITC team and working with the ITC to determine the appropriate artifacts and resources needed. The PM will need to have appropriate funding for the necessary testing tools and resources and will need to work with the ITC team to identify an appropriate schedule for the work to be done.</td>
</tr>
<tr>
<td>ITC Team</td>
<td>Developer; User</td>
<td>(Developer) The ITC team is responsible for the development and maintenance of the simulation testing environment (ITC framework). The ITC team consists of a Lead, Software Developers, Simulation Leads, Testers, and Quality Assurance personnel. (User) During initial efforts on the GPM project, it is assumed that the ITC team will aid in performing the initial testing activities in order to provide a full end-to-end solution to the IV&amp;V program to support dynamic analysis processes. (IV&amp;V Project POC) One ITC member will attended IV&amp;V project meetings, be added to project distributions, and serve as the single POC between the ITC team and the IV&amp;V project.</td>
</tr>
<tr>
<td>IV&amp;V Project Analyst</td>
<td>Primary ITC User</td>
<td>(User) The IV&amp;V project analyst is the primary user of the ITC framework. Project analysts are responsible for developing natural language test scenarios and test procedures as well as test scripts. Project analysts will perform the testing to include regression testing, stress testing, and performance-based testing. Project analysts are responsible for executing the tests, analyzing the test results, and writing issues and observations against their findings. (ITC POC) One IV&amp;V project member will serve as the single POC between the ITC team and the IV&amp;V project.</td>
</tr>
<tr>
<td>NASA Project Point of Contact (POC)</td>
<td>Customer</td>
<td>The NASA project POC serves as the designated Project interface for the IV&amp;V efforts. Responsible for providing artifacts or access to artifacts and associated schedules in support of the IV&amp;V efforts (electronic access is preferred). Responsible for review of IV&amp;V analysis results and ensuring appropriate levels of coordination in terms of resolution of any shortcomings with the system/software as revealed from these analysis results. The NASA project POC also helps ensure effective and efficient communications between the development project and ITC team members and timely delivery of all needed artifacts.</td>
</tr>
<tr>
<td>User / Stakeholder</td>
<td>User Type(s)</td>
<td>User Description and Role</td>
</tr>
<tr>
<td>-------------------------------------------------------</td>
<td>----------------------------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>NASA Project Software Developers and Testers</td>
<td>Customer; Potential ITC Developer</td>
<td>(Customer) NASA project software developers and testers are the end customers of the issues, observations, and reports produced by the IV&amp;V analysts. The ITC has also found that having access to a software developer and/or tester for a given project increases effectiveness while decreasing time spent on some critical tasks. Communicating with software developers and testers provides valuable insight to ITC team members on the technologies and techniques already being implemented on a given project. (Potential ITC Developer) Although most likely infrequent, NASA project software developers and testers could serve as valuable members of the ITC team if agreements are made early in the project's lifecycle to collaborate on the development of simulation and test environments. For the GPM project, this role is being addressed by members of the GPM Operational Simulator (Go-Sim) development team.</td>
</tr>
<tr>
<td>Chief Engineer / R&amp;D Lead</td>
<td>Customer</td>
<td>The Chief Engineer/Research Lead determines the ITC team’s budget for the current fiscal year. This budget is applied to development and maintenance of the core ITC framework, identifying, testing, and infusing new dynamic analysis tools into the Program, and training/travel for the ITC team members.</td>
</tr>
<tr>
<td>Software Assurance Tools</td>
<td>Potential Customer</td>
<td>Based on the added value capabilities provided by the addition of the ITC, the ITC would transition from the IV&amp;V Program’s R&amp;D branch to Software Assurance Tools branch, meaning that the deliveries and proven ITC tools and training would be used by IV&amp;V analysts in their day-to-day activities.</td>
</tr>
<tr>
<td>Network Operations</td>
<td>Stakeholder</td>
<td>Network Operations is responsible for ensuring appropriate securities for ITC data are in place. ITC team will collaborate with network operations to: (1) identify, acquire, configure, and manage ITC test assets; (2) deploy solutions to providing adequate access and permissions to the ITC team and other users.</td>
</tr>
<tr>
<td>Other IV&amp;V Management</td>
<td>Stakeholder</td>
<td>IV&amp;V Management receives most status from the ITC team and its capabilities via the Chief Engineer. Presentations, return on investment discussions, etc. can be provided to IV&amp;V Management as necessary by the ITC team as requested.</td>
</tr>
</tbody>
</table>
## Table 4: ITC Team Roles and Responsibilities

<table>
<thead>
<tr>
<th>Role</th>
<th>Responsibility</th>
<th>External Interfaces</th>
</tr>
</thead>
<tbody>
<tr>
<td>ITC Lead</td>
<td>Leads the overall program and technical direction of the ITC team. Responsible for planning iteration tasking, project tasking, executing to project budget, facilitating interactions with IV&amp;V projects, and identifying future tools and opportunities for collaboration with various projects.</td>
<td>Chief Engineer/Research Lead, Project points of contact, Network Operations, IV&amp;V Management, IV&amp;V Project Analysts</td>
</tr>
<tr>
<td>ITC Lead Software Engineer</td>
<td>Leads development of the core ITC framework by choosing, developing, monitoring, and revising team practices and technologies to support team’s objectives.</td>
<td>Project points of contact and software developers/testers</td>
</tr>
<tr>
<td>Developer(s)</td>
<td>Responsible for development of the ITC framework and additional supporting tools and technologies</td>
<td>Technical tool vendor support, software engineers and developers/testers as necessary (if collaborations established)</td>
</tr>
<tr>
<td>Project Specific Expert(s)</td>
<td>Responsible for identifying tools and technologies to stimulate systems under test and providing training and materials to IV&amp;V project analysts</td>
<td>Software Engineer and project software developers/testers</td>
</tr>
<tr>
<td>ITC Simulation Lead(s)</td>
<td>Responsible for identification and expertise in simulation tools and technologies and ensuring mechanisms for interfacing into ITC framework</td>
<td>Technical tool vendor support</td>
</tr>
<tr>
<td>ITC Tester(s)</td>
<td>Tests the ITC tools and technologies for complete functionality for each build. Executes regression tests, user scenarios, etc.</td>
<td></td>
</tr>
<tr>
<td>Quality Control and Configuration Manager</td>
<td>Ensures the quality of software development and items are controlled appropriately</td>
<td></td>
</tr>
</tbody>
</table>
4.6 PROPOSED SUPPORT ENVIRONMENT

The new capabilities provided to the IV&V Program by the ITC will require investment in hardware in the form of a server, large data storage devices, and potentially some system hardware or the equivalent in a de-ruggedized form. In addition to the new hardware, software Application Program Interfaces will need to be developed to create the interfaces discussed previously. Software packages will be needed to run the emulations, to support the development of executable test cases, and software to aid in data reduction and analysis.

An effort should be made to keep the interface to the analyst as simple as possible. The analysts currently develop and use a various forms of models, state charts, and test cases to support their current tasks. Additional work levied upon the analysts by this capability should be monitored and minimized.

Initial work will involve executing flight software using individual workstations, stimulating the flight software inputs, and analyzing the output data. Secondary work will involve development of a graphical user interface to provide analysts with a standard work environment. Translational tools will need to be developed to help analysts feed test cases into the ITC. Finally, the ITC should have distributed networking capabilities so that the graphical user interface and simulation controls can be utilized from analysts’ personal computers, and simulations, test cases, and test results can be executed and stored on a centralized server.
5 OPERATIONAL SCENARIO

The system is operated in several different phases, and has some non-operational prerequisites. Before the system can be used and testing can begin, the following requirements must be fulfilled:

1. The required behavior of software under test must be understood. This includes software requirements and the environment under which it runs. This also includes the hardware specifications, the identification of the necessary emulator and documentation and a working executable build of the software and any of the software’s dependencies.

2. The ITC must be configured to simulate the environment under which the software runs. It will be imperative for the ITC team to have all of the information and tools necessary to run the software under test. This includes any necessary documentation (i.e. Interface Control Documents, emulator documentation, etc.) as well as the understanding and emulation of any additional hardware or software dependencies (housekeeping hardware and software) to be able to fully exercise the software and capture inputs and outputs. There may also be external simulations that need to be developed to fully test the software as well, such as propagation, atmospheric, or ephemeris models.

3. Test scenarios must be developed. This could be manually driven test inputs as in Figure 4: Rudimentary Testing, or scripted tests driven from the validation and verification artifacts as in Figure 3: Typical Data Flow.

Once all prerequisites are met, the system can begin to perform tests.

1. A project must be created. Projects can be shared among many users. A project is a collection of artifacts that include any of the following: test scenarios, test procedures, test scripts, test models, hardware emulators, environment simulators, code, instrument simulators, binaries, etc. Executable tests (scripts, models, etc) will request use of certain artifacts in the project.

2. The user will select a set of executable test scenarios to perform.

3. Assuming that all artifacts the scenario calls for are present, the test will execute.

4. The user will be provided real-time data on the executing simulation to support verification activities.

5. The test output will be saved in the logging system and reports can be generated from the log. The ITC will have some method of extendable method of generating reports.

6. Analysts can use external tools to parse and analyze the actual output versus the expected output.

An example of how the system works within the IV&V Program can be illustrated in this example involving a new project that has come for test:

1. A NASA Project is selected for work by the IV&V Program and the IV&V Project Execution Plan (IPEP) includes verification and validation of implementation.

2. The IV&V Project Manager will establish a point of contact for the ITC team to discuss the requirements and appropriate path for dynamic analysis and potential integration into the ITC framework.

   a. Items of initial discussion include:

      i. Availability of Project Artifacts and Repositories to the ITC
      ii. A software development team and testing point of contact for the ITC
      iii. Current development project testing plans and procedures
iv. Identification of ITC shadowing and participation in test team tabletop discussions, internal peer reviews, and other reviews, development project testing rigs and labs
v. Can IV&V leverage project testing labs
vi. Integration opportunities into ITC

b. Outcomes
i. ITC provides recommendation of work to support dynamic analyses for given project. Communicates recommendation to both project and IV&V Program.

3. IV&V Project Manager decides whether or not to support dynamic analysis tasking based on ITC recommendation
4. ITC Team will provide a list of tasks to be performed the given project based on internal discussions and developer input. Minimal tasks will include development of test scenarios by the IV&V project team to exercise the system under test.
5. The ITC team will identify tools and technologies for integration and implement all the interfaces required to the software and hardware.
6. The ITC team will generate a test build configuration for the given project and release it to all authorized users.
7. The IV&V project team will execute their tests using the ITC framework, analyzing real-time results and logging data. This data will be retained on the ITC central server.
8. The IV&V project team will generate the following from their work.
   a. TIMS and any reports generated
   b. Independent Test Plan
   c. Independent Test Report
   d. Functional Areas Tested (Scenarios, Procedures, Scripts Generated)
   e. ITC Framework feature requests and change requests
   f. ITC bugs and fixes
   g. Lessons Learned
6 SUMMARY OF IMPACTS

6.1 OPERATIONAL IMPACTS

As new methods of testing will be provided to the analysts, new data can be collected. Not only will the analysts have to make sure the components of the capability they need for their testing are provided, but they will also have to make sure their test cases are in such a format that it can be understood by the system. Project leads will be able to gather data from the system in the form of reports or simple queries so they can get an idea of what tests were run and what needs to be looked at further.

New testing capability means that different types of test cases will have to be created. The capability has the ability to perform performance based testing, and this area is one that the IV&V Program has dealt with on a limited number of projects. The analyst will need to have an understanding of the target hardware so they will know what to expect for the test. The analyst may be required to help the ITC administrators determine what is necessary to run the project under test in a simulated environment so it can be configured for the analyst. In addition, the analyst will have to use the results of these new types of tests in their deliverables.

6.2 ORGANIZATIONAL IMPACTS

The new capability increases the exposure of the IV&V Program to the development process. The Program will be put in a position where it can perform actual product testing for a developer during the prototype phase. With this new responsibility, and the increased effectiveness of the work done, some new responsibilities will be created and training required. However, the impacts to the effectiveness of the work performed and the benefits to the missions supported will be tremendous. The verification of software behavior, requirements and interface implementation will be thoroughly tested, providing increased confidence in IV&V results and better support to the missions. Many issues can only be fully tested with dynamic software analysis and the ITC will provide the Program with an efficient means to provide this service to the projects earlier in the software development cycle. The new testing capability will significantly increase the thoroughness of the verification process and significantly decrease the likelihood of errors in mission software.

The development and use of the ITC will result in a significant shift in the relationship between the IV&V Program and the projects it supports. In order for the ITC team to develop the necessary simulations there will need to be artifacts provided from the supported project that haven’t been provided in the past. Additionally, the ITC team will not only be developing simulation environments for use in the Program, but will also be teaming with other facilities to assist with simulation developments at their locations as well. The V&V work being done will still be independent, but will also require more collaboration between teams. An example of the collaboration necessary is shown in Figure 5: Organizational Responsibilities.
Analysts and administrators will have to be trained. The administrator group will support the analysts who use the capability. They will also work with vendors to obtain licenses and tools to increase the coverage of the capability’s simulation and emulation capability.

![Organizational Responsibilities Diagram]

**Figure 5: Organizational Responsibilities**

### 6.3 IMPACTS DURING DEVELOPMENT

During development, a project will be chosen to use as the initial target for the capability. This gives the development team an avenue to show incremental progress. It also allows a select group of analysts to test the capability and provide feedback to the development team. This feedback would be especially valuable in helping to work out ease of use, new features, and data retention/reporting methods. The development team would act as interim administrators to fill the role of system configuration.
### ACRONYMS AND ABBREVIATIONS

<table>
<thead>
<tr>
<th>6.3.1.1 Acronym</th>
<th>6.3.1.2 Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CTCA</td>
<td>Core Test Capability Architecture</td>
</tr>
<tr>
<td>GUI</td>
<td>Graphical User Interface</td>
</tr>
<tr>
<td>IT</td>
<td>Information Technology</td>
</tr>
<tr>
<td>ITC</td>
<td>Independent Test Capability</td>
</tr>
<tr>
<td>IV&amp;V</td>
<td>Independent Validation and Verification</td>
</tr>
<tr>
<td>NASA</td>
<td>National Aeronautics and Space Administration</td>
</tr>
<tr>
<td>PC</td>
<td>Personal Computer</td>
</tr>
<tr>
<td>R&amp;D</td>
<td>Research and Development</td>
</tr>
<tr>
<td>SCM</td>
<td>Software Configuration Management</td>
</tr>
<tr>
<td>SUT</td>
<td>Software/System Under Test</td>
</tr>
<tr>
<td>V&amp;V</td>
<td>Validation and Verification</td>
</tr>
<tr>
<td>WBS</td>
<td>Work Breakdown Structure</td>
</tr>
</tbody>
</table>