NASA/CR-2002-211401

Survey of NASA V&V Processes/Methods

Charles Pecheur, Stacy Nelson

Prepared for NASA Ames Research Center
under Contract NRA 8-30/TA-5/Northrop Grumman

April 2002

The NASA STI Program Office . . . in Profile

Since its founding, NASA has been dedicated to the advancement of aeronautics and space science. The NASA Scientific and Technical Information (STI) Program Office plays a key part in helping NASA maintain this important role.

The NASA STI Program Office is operated by Langley Research Center, the Lead Center for NASA's scientific and technical information. The NASA STI Program Office provides access to the NASA STI Database, the largest collection of aeronautical and space science STI in the world. The Program Office is also NASA's institutional mechanism for disseminating the results of its research and development activities. These results are published by NASA in the NASA STI Report Series, which includes the following report types:

- TECHNICAL PUBLICATION. Reports of completed research or a major significant phase of research that present the results of NASA programs and include extensive data or theoretical analysis. Includes compilations of significant scientific and technical data and information deemed to be of continuing reference value. NASA's counterpart of peer-reviewed formal professional papers but has less stringent limitations on manuscript length and extent of graphic presentations.

- TECHNICAL MEMORANDUM. Scientific and technical findings that are preliminary or of specialized interest, e.g., quick release reports, working papers, and bibliographies that contain minimal annotation. Does not contain extensive analysis.

- CONTRACTOR REPORT. Scientific and technical findings by NASA-sponsored contractors and grantees.

- CONFERENCE PUBLICATION. Collected papers from scientific and technical conferences, symposia, seminars, or other meetings sponsored or cosponsored by NASA.

- SPECIAL PUBLICATION. Scientific, technical, or historical information from NASA programs, projects, and missions, often concerned with subjects having substantial public interest.

- TECHNICAL TRANSLATION. English-language translations of foreign scientific and technical material pertinent to NASA's mission.

Specialized services that complement the STI Program Office's diverse offerings include creating custom thesauri, building customized databases, organizing and publishing research results . . . even providing videos.

For more information about the NASA STI Program Office, see the following:

- Access the NASA STI Program Home Page at *http://www.sti.nasa.gov*

- E-mail your question via the Internet to help@sti.nasa.gov

- Fax your question to the NASA Access Help Desk at (301) 621-0134

- Telephone the NASA Access Help Desk at (301) 621-0390

- Write to:
  NASA Access Help Desk
  NASA Center for AeroSpace Information
  7121 Standard Drive
  Hanover, MD 21076-1320

NASA/CR-2002-211401

Survey of NASA V&V Processes/Methods

Charles Pecheur
RIACS
Ames Research Center, Moffett Field, California

Stacy Nelson
CSC
Ames Research Center, Moffett Field, California

Prepared for NASA Ames Research Center
under Contract NRA 8-30/TA-5/Northrop Grumman

National Aeronautics and
Space Administration

Ames Research Center
Moffett Field, California 94035-1000

April 2002

Available from:

NASA Center for AeroSpace Information
7121 Standard Drive
Hanover, MD 21076-1320
(301) 621-0390

National Technical Information Service
5285 Port Royal Road
Springfield, VA 22161
(703) 487-4650

APPROVALS

Approvals:		
Task Lead:	Charles Pecheur	
Project Manager:	Marshal Merriam	Marshal L. Merriam
Area Manager:	Michael Lowry	

RECORD OF REVISIONS

REVISION	DATE	SECTIONS INVOLVED	COMMENTS
Initial Delivery	8/10/01	Sections 2, 3.2 and 3.3, 6, 7, 8, 9, 10	Draft containing results of **Survey of NASA software V&V processes and methods**. This is a draft only and not intended to be the final deliverable.
Initial Delivery	8/24/01	Sections 3.4, 4, 11	Draft containing results of **Applicability of V&V to IVHM**. This is a draft only and not intended to be the final deliverable.
Update	9/7/01	Section 3.2	Minor changes to **Survey of NASA software V&V processes and methods**. This is a draft only and not intended to be the final deliverable.
Update	9/7/01	Section 4	Refinement of **Applicability of V&V to IVHM**. This is a draft only and not intended to be the final deliverable.
Initial Delivery	9/7/01	Section 1, 5, 12, 13, 14, 15	Draft containing results of **Special IVHM V&V Requirements**. This is a draft only and not intended to be the final deliverable.
Revisions	9/28/01	Sections 1, 2, 3, 5, 16 and 17	Revisions to incorporate comments. This is a draft only and not intended to be the final deliverable.
Final	10/24/01	Sections 2, 3, 7 and 12	Revisions to incorporate comments from review. This is the final deliverable.

TABLE OF CONTENTS

1. DOCUMENT CONVENTIONS

The following conventions are used throughout this document:

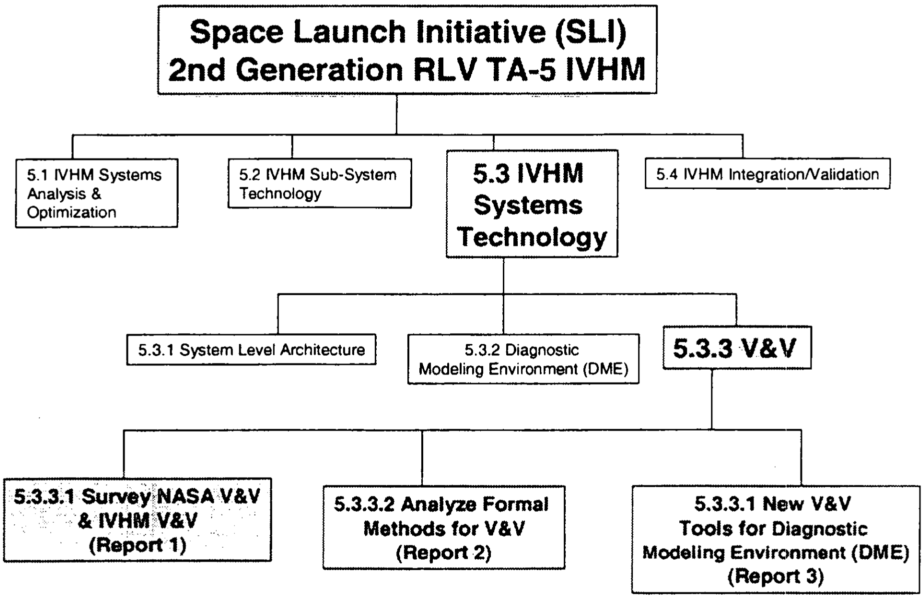
- The term "formal testing" has two meanings. Traditionally, "formal testing" has been used to describe an official test occurring at the end of each life cycle phase to demonstrate that software is ready for intended use. It includes the following:
    - Approved Test Plan and Procedure
    - Quality Assurance (QA) witnesses
    - Record of discrepancies (Problem Reports)
    - Test Report

  With the invention of more advanced software, the term "formal testing" also refers to a type of mathematical testing using Formal Methods. Formal Methods include the following types of tests:
    - Model Checking
    - Theorem Proving
    - Static Analysis
    - Runtime Monitoring

  Therefore, to avoid on confusion in this document, the traditional use of "formal testing" has been replaced with the term "official testing". The term "formal testing" used in this document means formal mathematical testing (i.e. Formal Methods).
- The term "Program" is used as a generic term to describe a mission or project conducted at NASA. For example, this document contains a survey of the Deep Space One Program, rather than the Deep Space One Mission.
- The term "Advanced Software" is used to describe model-based and/or artificial intelligence (AI) software like model based reasoning software.

2. EXECUTIVE SUMMARY

This report was prepared by the NASA Ames Research Center Automated Software Engineering (ASE) group as the deliverable for Task 5.3.3.1 "Survey NASA V&V & IVHM V&V", highlighted in green on Figure 1. It is the first of three reports for Task 5.3.3 "V&V", highlighted in blue on Figure 1.


Space Launch Initiative (SLI)
2nd Generation RLV TA-5 IVHM
5.1 IVHM Systems Analysis & Optimization
5.2 IVHM Sub-System Technology
5.3 IVHM Systems Technology
5.4 IVHM Integration/Validation
5.3.1 System Level Architecture
5.3.2 Diagnostic Modeling Environment (DME)
5.3.3 V&V
5.3.3.1 Survey NASA V&V & IVHM V&V (Report 1)
5.3.3.2 Analyze Formal Methods for V&V (Report 2)
5.3.3.1 New V&V Tools for Diagnostic Modeling Environment (DME) (Report 3)

Legend:

• SLI	Space Launch Initiative
• IVHM	Integrated Vehicle Health Management
• V&V	Verification and Validation
• RLV	Re-usable Launch Vehicle

*(**Note:** The current space shuttle is the 1st Generation RLV. The goal of SLI is to upgrade the space shuttle to the 2nd Generation RLV)*

Figure 1: SLI 2nd Generation RLV TA-5 IVHM Project Structure

The purpose of this report is to describe current NASA V&V techniques and to explain how these techniques are applicable to 2nd Generation RLV IVHM software. It also includes a list of NASA V&V Standards for certification of airborne software. Finally, this report contains recommendations for special V&V requirements for IVHM.

This report is divided into the following three sections:

- Survey – Current NASA V&V Processes/Methods
- Applicability of NASA V&V to 2nd Generation RLV IVHM
- Special 2nd Generation RLV IVHM V&V Requirements

Because the Survey of Current V&V Processes/Methods revealed that Advanced Software like IVHM has been used primarily on experimental missions, additional investigation was conducted. Further research revealed that Advanced Software has not been utilized more effectively because it is difficult to adequately verify and validate this software in accordance with NASA standards and guidelines for certification of airborne software. Therefore, special effort was made to obtain the appropriate NASA standards and to review the required V&V tasks and provide recommendations for meeting these requirements.

Report 2 uses Report 1 as a foundation and builds on that foundation by recommending new Formal Verification processes and methods for V&V of Advanced Software.

Additionally, in order to provide a comprehensive recommendation for V&V, this section contains information about software methodologies, project management and other topics implicitly related to V&V because a successful V&V effort is dependent upon these items.

The Executive Summary includes an overview of the main points from each section. Supporting detail, diagrams, figures and other information are included in subsequent sections. A glossary, acronym list, appendices and references are included at the end of this report.

2.1.1. SURVEY – CURRENT NASA V&V PROCESSES/METHODS

A survey of current practice in Verification & Validation (V&V) of safety-critical software across NASA was conducted to support initial planning and analysis for V&V of the 2nd Generation Re-usable Launch Vehicle IVHM.

Software V&V is defined as the process of ensuring that software being developed or changed will satisfy functional and other requirements (verification) and each step in the process of building the software yields the right products (validation).[9 & 18] In other words:

- Verification – Build the Product Right
- Validation – Build the Right Product

Three missions were selected as being representative of current software V&V practices. They are listed below and a summary of survey results follows:

- Deep Space One - Remote Agent (DS1 RA)
- X-37 IVHM Experiment
- Checkout & Launch Control System (CLCS)

2.1.1.1. Deep Space One Remote Agent Experiment [4]

The V&V process for Deep Space One Remote Agent Experiment resulted in the following:

- Successful V&V process contributed to the Remote Agent team becoming co-winners of the NASA 1999 Software of the Year Award
- Operations Scenarios were used effectively to test nominal and off-nominal events.
- Baseline testing and effective use of different fidelity testbeds resulted in project team agility and reduced testing costs
- Operational Readiness Tests resulted in identifying procedural problems during "dress rehearsal" so they could be corrected before the actual mission

***Note:** Formal Verification was also conducted on the Remote Agent Experiment (RAX). The Formal Verification included processes and tools to analyze and verify complex dynamic systems like Advanced Flight Software, using mathematically sound analysis techniques. Formal Methods applied to RAX are described in Report 2 – NASA V&V of Advanced Systems.*[1]

2.1.1.2. X-37 IVHM Experiment [11]

The X-37 plane is a reusable launch vehicle designed to operate in both the orbital and re-entry phases of flight. The X-37 IVHM Experiment will involve running integrated vehicle health management software onboard the X-37 plane. This experiment is still in the early stages. To date, the project team has developed a comprehensive V&V Plan based on NASA standards. One test has been conducted with successful results indicating that the V&V plan is being used to find anomalies, bugs, and incomplete code. This test also demonstrated that the IVHM software was able to run a nominal scenario script for

4.5 days without crashing and L2 (new version of Livingstone software) found an isolated single component failure.

2.1.1.3. Checkout & Launch Control System (CLCS)

In 1999, it became apparent that the Atlas software delivery (part of CLCS) was late and over budget. An Independent Assessment Team reviewed CLCS processes and recommended improvements. As a result the CLCS project was re-structured with an emphasis on functional releases rather than system software deliveries.[12]

The CLCS project has comprehensive V&V plans based on NASA standards and contained in an online repository (http://clcs.ksc.nasa.gov/docs/test-specs.html). To date, only the Hypergolic Maintenance Facility has been validated under the new project structure. Specific information regarding lessons learned from V&V of the Hypergolic Maintenance Facility will be available in fourth quarter 2001.[2]

One important lesson learned to date is the necessity for evaluating IV&V budget requirements early in the project.[2] A manned mission and any mission or program costing more than $100M will require IV&V. Appendix D, "Independent Verification and Validation (IV&V) Criteria" contains recent criteria for this evaluation.[17]

2.1.2. APPLICABILITY OF NASA V&V TO IVHM

The purpose of Section 4, "Applicability of NASA V&V to 2nd Generation RLV IVHM" is to discuss which NASA V&V processes and methods are applicable and beneficial to the IVHM system planned for the 2nd Generation RLV.

General NASA V&V standards were reviewed, as well as, V&V activities described in the Surveys of the Deep Space One Remote Agent Experiment, X-37 IVHM Experiment and CLCS. Each V&V task was categorized as either a Process or Method and listed in the V&V Processes Applicability Matrix (Section 4.1.3.1) or the V&V Methods Applicability Matrix (Section 4.1.4.1). Both Applicability Matrices include a column designating whether the task is applicable to 2nd Generation RLV IVHM. Tasks designated as being applicable may be incorporated into the V&V Plan for the 2nd Generation RLV IVHM system.

Seventeen V&V Processes and forty-three V&V Methods were analyzed. Sixteen V&V Processes are applicable to IVHM and one process may be applicable depending upon the size and complexity of the project. Forty V&V Methods are applicable to IVHM and three are not.

2.1.3. SPECIAL IVHM V&V REQUIREMENTS

The purpose of Section 5, "Special IVHM V&V Requirements" is to identify unique V&V issues and requirements specific to the 2nd Generation RLV IVHM system. Special requirements were taken from lessons learned on the Deep Space Remote Agent Experiment and the CLCS reorganization and include the following recommendations:

- Educate mission operators about autonomous onboard planning technology in order to move beyond the mindset of predictability from an autonomous system and to provide a basis for acceptance of rigorous V&V as appropriate for certification so Advanced IVHM Software can fly onboard 2nd Generation RLV.
- Use a combination of the Spiral and Waterfall methodologies when presenting new technology. V&V is different for each methodology as described in Section 5.
- Enforce accountability between partners to ensure V&V is performed effectively
- Organize modeling teams with responsibility for entire sub-systems to ensure internal coherence of the resulting model and communication about models to the V&V team.
- Evaluate testing coverage of autonomous software
- Develop tools to mitigate the effect of late changes to requirements because the V&V effort for changes is currently a laborious process. The DS1 RA team was forced to forego some late changes because there was insufficient time for V&V.
- Develop ground tools early and use them during testing

- Design telemetry early and use during testing
- Develop better model validation processes and tools (some tools under development at NASA)
- Use new graphical tools being developed to provide visual inspection and modification of mission profiles, as well as constraint checking
- Develop tools and simplify the modeling languages so spacecraft experts can encode models themselves and explain the models to test engineers more effectively.
- Simplify the specification of goals (New graphical tools being developed at NASA) and automate consistency checking

3. SURVEY – CURRENT NASA V&V PROCESSES/METHODS

The purpose of the Survey of Current NASA V&V Processes/Methods is to provide actual examples of software Validation and Verification (V&V) processes used at NASA.

Software V&V is defined as the process of ensuring that software being developed or changed will satisfy functional and other requirements (validation) and each step in the process of building the software yields the right products (verification).[9 & 18] In other words:

- Validation – Build the Right Product
- Verification – Build the Product Right

Three recent or ongoing NASA programs were selected as being representative of Software V&V practices at NASA. They are listed below and described in subsequent sections of this report:

- Deep Space One - Remote Agent
- X-37 IVHM Experiment
- Checkout & Launch Control System (CLCS)

Note: *Appendix A includes a list of other missions incorporating Livingstone software. Appendix B includes a list of other projects reviewed but not selected for this report. It also includes a list of Advanced Software projects that have been proven effective but have not been certified due to lack of adequate V&V methods.*

A formal survey was developed consisting of the questions listed in Figure 2 below. These questions were derived from the following documents, as well as previous V&V and traditional software testing experience:

- Software Test Plan (STP) – MIL STD 498 DI-IPSC-81438 [22]
- *Software Considerations in Airborne Systems and Equipment Certification,* Document No. RTCA (Requirements and Technical Concepts for Aviation) /DO-178B [20]

3.1.1. Figure 2 – Survey Questions

Number	Question
1	Describe the program (brief overview of goals and hardware)
2	Program maturity (For example: software flew, earthbound experiment etc)
3	Describe program software
4	Which organization performed the V&V? Was this organization independent?
5	Provide an overview of V&V (optional, depending upon Program size and complexity)
6	What Verification Methods were used?
7	What Validation Methods were used? • Describe Operations Scenarios, if applicable • Describe Test Environment including the following, if applicable: • Software • Hardware • Describe Testing Tools • Describe the V&V Team • Describe Testing Methods and Procedures
8	Describe Methods for Analyzing Test Results
9	Describe Review Methods
10	Describe the success of V&V Process (What worked/What needed improvement)

Answers to survey questions were obtained from reading project documents and NASA V&V guidelines and interviewing key project team members. References to project documents are listed in the Reference Section in numerical order as cited in the report. Special thanks to the individuals listed in Figure 3 for their time in answering questions and providing project documents:

3.1.2. Figure 3 – Individuals Participating in Survey

Project/Mission	Individual	Location
DS1-Remote Agent	Bill Millar	NASA ARC
	Nicola Muscettola	NASA ARC
X-37	Scott Christa	NASA ARC
	Scott Poll	NASA ARC
	Jeff Samuels	NASA ARC
	Mark Schwabacher	NASA ARC
CLCS	Ric Hurt	NASA KSC
	Steven Raque	NASA IV&V
	Frank Rockwell	NASA IV&V
	Glenn Semmel	NASA KSC
	Scott Wilson	NASA KSC

3.2. Deep Space One Remote Agent

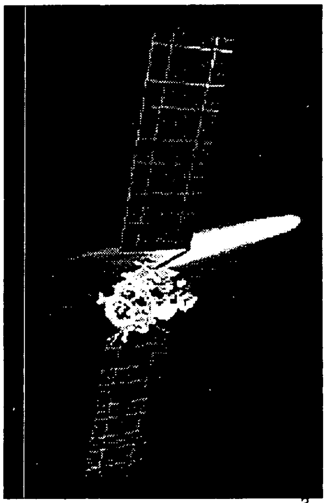
Information, diagrams and pictures presented below are from references 1and 3 - 6 in the references section.

3.2.1. Program Description

Deep Space One (DS1) launched from Cape Canaveral on October 24, 1998. The objective of the DS1 mission was to test 12 advanced technologies in deep space so these technologies could be used to reduce the cost and risk of future missions.[1] One of the 12 technologies on DS1 was called Remote Agent (RA). RA is an artificial intelligence (AI) software product designed to operate a spacecraft with minimal human assistance.

3.2.2. Program Maturity

RA software flew on DS1 and was flight validated between May 17 and May 21, 1999[3]



Artist Rendering of DS1[3]

3.2.3. Program Software

RA is unique and differs from traditional spacecraft commanding because ground operators can communicate with it using goals like "during the next week take pictures of the following asteroids and thrust 90% of the time". It is a model-based system composed of the three AI technologies listed below:

- Planner-Scheduler
- Smart Executive (EXEC)
- Livingstone or MIR (Mode Identification and Reconfiguration) – a model-based fault diagnosis and recovery system[4]

The Planner-Scheduler generates plans that RA uses to control the spacecraft. Given the initial spacecraft state and a set of goals, Planner-Scheduler generates a set of synchronized high-level tasks to achieve goals.

The Smart Executive or EXEC is responsible for the following:

- o Requesting and executing plans from the planner
- o Requesting/executing failure recoveries from MIR
- o Executing goals and commands from human operators
- o Managing system resources
- o Configuring system devices
- o System-level fault protection
- o Achieving and maintaining safe-modes as necessary[4]

The EXEC is goal oriented, for example: EXEC's goal: "keep device A on from time X to time Y". If EXEC detects that device A was off during the prescribed time, it issues the commands to turn it back on.[1]

MIR is responsible for Mode Identification (diagnosis) and Mode Reconfiguration (recovery) explained below:

- o Mode Identification observes the EXEC issuing commands, receives state information from the spacecraft and uses model-based inference to deduce the state of the spacecraft and provide feedback to EXEC

- Mode Reconfiguration serves as a recovery expert. It takes EXEC constraints and uses declarative models to recommend a single recovery action to EXEC[5]

The Remote Agent Architecture is shown in Figure 4.

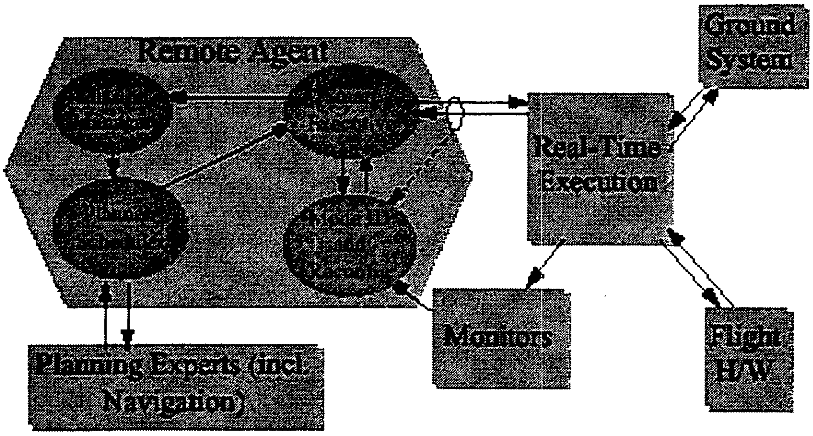


Figure 4 – Remote Agent Architecture
Diagram from *Validating the DS1 Remote Agent Experiment*[5]

3.2.4. Organizations Performing V&V

NASA AMES Research Center and the Jet Propulsion Laboratory

3.2.5. Verification Methods [4]

RA was verified to prove it could autonomously command a system as complex as a spacecraft for an extended period of time. Specific verification objectives are listed below:

- **Planner-Scheduler**
  - Generate plans onboard the spacecraft
  - Reject low-priority unachievable goals
  - Re-plan following a simulated failure
  - Enable modification of mission goals from ground
- **EXEC**
  - Provide a low-level commanding interface
  - Initiate onboard planning
  - Execute plans generated onboard and from ground
  - Recognize and respond to plan failures
  - Maintain required properties in the face of failures
- **Livingstone (MIR)**
  - Confirm executive command execution
  - Demonstrate model-based failure detection, isolation and recovery
  - Demonstrate ability to update Livingstone state via ground commands
- **Other Verification objectives**
  - Address impact of introduction of RA into a "traditional" spacecraft software architecture
  - Demonstrate that RA could be commanded at various autonomy levels
  - Integrate RA (LISP) with DS1 Flight Software
  - Adapt RA models and scenarios to reflect operational constraints imposed by the flight team even late in development process

In order to achieve the verification objectives, the DS1 team used the following verification methods:

- Informal Reviews as needed:
  - The RAX team was organized horizontally so team members specialized in one of the Planner-Scheduler, EXEC or MIR engines and each team was responsible for modeling

all spacecraft subsystems for their engine. Test Engineers had to meet with individuals from each team to gain a complete understanding of how a subsystem was commanded by RA.

- Due to time constraints and the experimental nature of this mission, Official Reviews were limited to the following:
    - Issues or change requests were recorded via Problem Reports.
    - The Change Control Board reviewed Problem Reports and made "go-no go" decisions.

3.2.6. Validation Methods

Validation of RA was very rigorous in order to qualify to run onboard DS1.[4] Because of the success of RA, the following validation methods, specifically Operations Scenarios, have been incorporated in the latest version of NASA Procedures and Guidelines [18]. This is a big step toward getting Advanced Software certified for use onboard spacecraft. Validation Methods are described in the following sections:

- Operations Scenarios
- Testing Environment (includes types of tests performed in each environment)
- Testing Tools
- V&V Team
- Testing Methods and Procedures

3.2.6.1. Operations Scenarios

Operations Scenarios carefully designed to test nominal and off-nominal events. These scenarios are described below:

- **Twelve-hour scenario – used for testing until March, 1999** [4]
    - Imaging asteroids with MICAS (Miniature Integrated Camera and Spectrometer) to support optical navigation. ***Note:*** *Tests Planner-Scheduler's ability to reject low-level unachievable goals because the optical navigation window had time only to image a subset of the asteroid goals*
    - Simulated sensor failure
    - Low-level command to flip a switch
    - No onboard planning nor thrusting with IPS (ion propulsion system)
      Note: *Plan generated on the ground and uplinked to the spacecraft*
- **Six-day scenario – used for testing until March, 1999** [4]
    - On-board planning
    - Operating IPS
    - Divided into 2 Planning horizons
        - Horizon 1
            - Planner-Scheduler generates a plan for the following:
                - MICAS imaging for optical navigation
                - IPS thrusting
                - Switching off MICAS where a stuck-on-failure injection in the camera switch prevents RA from turning off the camera leading to a plan failure
                - Attempt to recover from plan failure and re-plan that produces a second plan to leave the camera on and a third plan for the second horizon
        - Horizon 2
            - Undo switch failure and ground informs MIR that switch is unstuck
            - IPS thrusting
            - Optical-navigation imaging
            - Two simulated failures
            - Communication failure on the 1553 bus (demonstrates successful recovery of communication with a device)

        - Thruster-valve-stuck-closed failure (demonstrates how MIR infers from an attitude error and models of the spacecraft dynamics that one particular pair of thruster valves is stuck shut. MIR then recommends switching ACS control modes to mitigate the problem)
- **Two-Day scenario – used for testing after March, 1999**[4]
    - The two-day scenario compressed the six-day scenario except MICAS was not switched off due to concerns regarding thermal effects. Additionally, RA would be required to produce at most 12 hours of thrusting to encounter the asteroid in July 1999.

To cope with time and resource limitations, a "baseline" testing approach was used to reduce the number of tests. Baseline tests were developed for each scenario and run on lower fidelity testbeds when there was a high confidence that test results would extend to higher-fidelity situations.

3.2.6.2. Testing Environment

Tests were distributed among low, medium and high-fidelity testbeds described in Figure 5 below:

Figure 5 - Deep Space One – Remote Agent Testbeds [4,5]

Testbed	**Fidelity**	**CPU**	**Hardware**	**Availability**	**Speed**	**Dates of RAX Readiness on Testbeds**
Spacecraft	Highest	Rad6000	Flight	1 for DS1	1:1	05/99
DS1 Testbed	High	Rad6000	Flight spares + DS1 sims	1 for DS1	1:1	04/99
Hotbench	High	Rad6000	Flight spares + DS1 sims	1 for DS1	1:1	03/99
Papabed	Medium	Rad6000	Flight spares + DS1 sims	1 for DS1	1:1	11/98
Radbed	Low	Rad6000	RAX Simulators	1 for RAX	1:1	04/98
Babybed	Lowest	PowerPC	RAX Simulators	2 for RAX	7:1	02/98
Unix	Lowest	SPARC UNIX	RAX Simulators only	Unlimited	35:1	08/97

Unix Testing [6]

The Planner-Scheduler team used the Unix testbed for unit testing. They repeatedly ran a batch of 269 functional tests with several variations of initial states, goals for the planner and model parameters. [4]

Babybed and Radbed Testing

The following tests were run on Babybed and Radbed [5]

- About 200 variations of the initial state and goals of the Planner-Scheduler while exercising Livingstone in hundreds of the likeliest failure contexts
- Planner-Scheduler and Livingstone tests exercised the EXEC
- System level interaction of all modules was tested with a suite of 20 additional test scenarios
- Total of more than 300 tests repeated for 6 software releases[6]

These tests were run rapidly because Babybed and Radbed used simulators that permitted faster than real-time execution. Even with simulators, testing was time consuming; therefore, to alleviate the time-consuming and error-prone nature of these tests, an automated testing tool was developed to do the following:

- Accept and encode scenario description as input
- Control the simulator and ground tools to execute the scenario
- Stop the test when appropriate by monitoring the telemetry stream
- Store all logs and downlinked files for later examination[4]

Total Run Time: about one week for all tests since tests could be scheduled overnight with no monitoring[4]

Test Schedule: Tests run after each major RAX software release.[5]

Papabed Testing

Once RA code was "frozen", six off-nominal system test scenarios were run on Papabed. These scenarios corresponded to the most likely and highest-impact scenarios. No bugs were detected in these scenarios.[4] A total of ten tests were run once on Papabed.[4]

Hotbench and Testbed Testing

Reserved for testing nominal scenarios and a few requirements for spacecraft health and safety.[5] A total of ten tests were run once on Hotbench. Two tests were run on Testbed for the final release.[4]

3.2.6.3. Testing Tools[4]

The following testing tools were used:

- Planner-Scheduler test suite including a Planner-Scheduler Test Generator that used Planner-Scheduler model knowledge to generate tests corresponding to plans starting at, near, or between boundary times. Boundary times were manually identified and indicate the topology at which the plans would change.
- Custom-built Automated Test Running Capability tool that allowed the team to quickly evaluate a large number of off-nominal scenarios

The following ground tools were also used:

- To provide adequate coverage and visibility into RA's onboard workings, a ground tools suite was designed to interface with the real-time RA-generated telemetry
- To allow the DS1 team to gain confidence in the onboard planner, the RAX team used a ground twin of the planner. It was identical to the onboard planner and could duplicate the onboard twin by tapping into real-time telemetry.
- PS-Graph displayed the problem-solving trajectory by Planner-Scheduler for each of the plans generated by the onboard planner
- A version of Stanley and Livingstone (MIR) was run on the ground to infer MIR's full internal representation of the spacecraft state from the telemetry
- Public Outreach via the Web – emailed summaries of events were presented in simple English with a Java-Applet timeline on the web

***Note:** The DS1 RA team identified the need for better testing tools and some work has begun on these types of tools at NASA ARC. Section 3.2.9, Suggestions for Improving V&V, contains a list of these tools.*

3.2.6.4. V&V Team[4]

Four half-time engineers

V&V Team Training

All testers were trained on testing procedures

3.2.6.5. Testing Methods/Procedures[9]

Testing Methods and Procedures for DS1 RA included the Operations Scenarios described in Section 3.2.6.1 and Operational Readiness Tests and the Other Tests described below:

Operational Readiness Tests[5]

Operational Readiness Tests (ORT) included a "dress rehearsal" of the following:

- Operational procedures
- Sequences for running the experiment
- Identifying contingency procedures

Two ORTs were performed as follows:

1. Ran through the first several hours of the 12 hour scenario focused on procedures for starting RAX
2. Ran through the entire 2 day scenario

During ORT, RAX ran on the Hotbench testbed and data was sent to workstations in the mission control center where RAX ground tools were running.

Result: A few problems were discovered and resolved prior to flight

Other Tests [5]

RAX was designed with "safety net' that allowed it to be completely disabled with single command sent either by ground or by onboard flight software. The only way RAX could affect spacecraft health was by consuming excessive resources (memory, downlink bandwidth and CPU) or by issuing improper commands. These two items were tested as follows:

- Executing a LISP script that consumed resources tested resource consumption.
- Subsystem engineers reviewed the execution traces of the nominal scenarios and performed automated flight rule checking to test issuing of improper commands.

3.2.7. Methods for Analyzing Test Results

Engineers reviewed output from test tools and compared to expected results (specific examples described above)

3.2.8. Review Methods [4]

Throughout 1998, the goal of testing was to discover bugs so they could be repaired. Beginning January 1999, the discovery of a bug did not automatically imply it would be fixed. Instead, a Change Control Board (CCB) composed of senior RAX project members reviewed every bug and the proposed fix in detail including specific lines of code to be changed. The CCB voted on whether of not to fix the bug depending upon the associated risk. Closer to flight, the DS1 instituted another CCB to review RAX changes. The CCB became increasingly conservative near mission launch date.

The effectiveness of testing process was analyzed through the Problem Reports filed between April 1997 and April 1999. Problem reports were grouped into categories and analyzed. The result is a list of "lessons learned" and ideas for improving future projects listed in Appendix B.

3.2.9. Success of V&V Process

The V&V process for Deep Space One resulted in the following:

- Successful V&V process contributed to the DS1-Remote Agent team becoming co-winners of the NASA 1999 Software of the Year Award
- Operations Scenarios were used effectively to test nominal and off-nominal events.
- Baseline testing and effective use of different fidelity testbeds resulted in project team agility and reduced testing costs
- Operational Readiness Tests resulted in identifying procedural problems during "dress rehearsal" so they could be corrected before the actual mission
- Formal Verification was also conducted. It included tools and processes to analyze and verify complex dynamic systems such as advanced flight software, using mathematically sound analysis techniques. Formal Methods applied to RAX are described in *Report 2 – NASA V&V of Advanced Systems.* [7]

Suggestions for Improving V&V

The following list summarizes the Lessons Learned by the DS1 team performing V&V. Detailed information for each item in this list is contained in Section 5 Special V&V Requirement for 2nd Generation RLV IVHM.

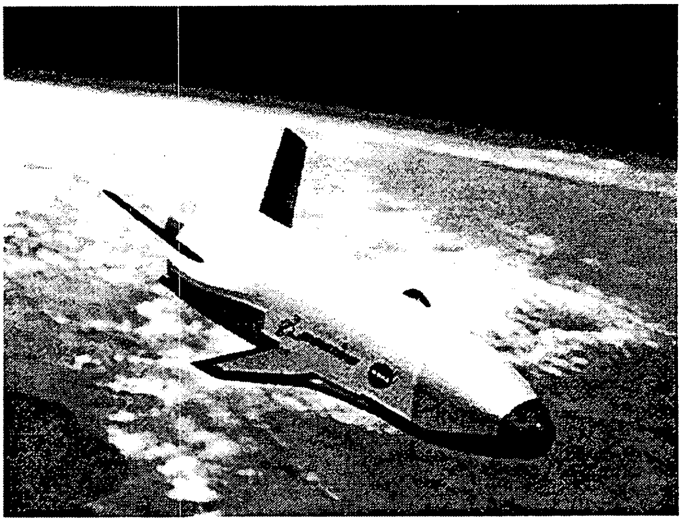
- Educate mission operators about autonomous onboard planning technology in order to move beyond the mindset of predictability from an autonomous system and to provide a basis for acceptance of rigorous V&V as appropriate for certification so Advanced IVHM Software can fly onboard 2nd Generation RLV.
- Organize modeling teams with responsibility for entire sub-systems to ensure internal coherence of the resulting model and communication about models to the V&V team.
- Evaluate testing coverage of autonomous software
- Develop tools to mitigate the effect of late changes to requirements because the V&V effort for changes is currently a laborious process. The DS1 RA team was forced to forego some late changes because there was insufficient time for V&V.
- Develop ground tools early and use them during testing
- Design telemetry early and use during testing
- Develop better model validation processes and tools (some tools under development at NASA)
- Use new graphical tools being developed to provide visual inspection and modification of mission profiles, as well as constraint checking
- Develop tools and simplify the modeling languages so spacecraft experts can encode models themselves and explain the models to test engineers more effectively.
- Simplify the specification of goals (New graphical tools being developed at NASA) and automate consistency checking

3.3. X-37 IVHM Experiment

Information, diagrams and pictures presented below are from references 8 - 11 in the references section.

3.3.1. Program Description

8



The NASA X-37 IVHM (Integrated Vehicle Health Management) Technology Experiment will involve running Integrated Vehicle Health Management software on-board the X-37 spacecraft. Mission objectives as listed below:

- Demonstrate benefits of in-flight IVHM to the operation of a Reusable Launch Vehicle
- Advance the Technology Readiness Level of this IVHM technology within a flight environment
- Operate IVHM software on the Vehicle Management Computer

3.3.2. Program Maturity[8]

Software on X-37 is being tested. First flight scheduled for 2002

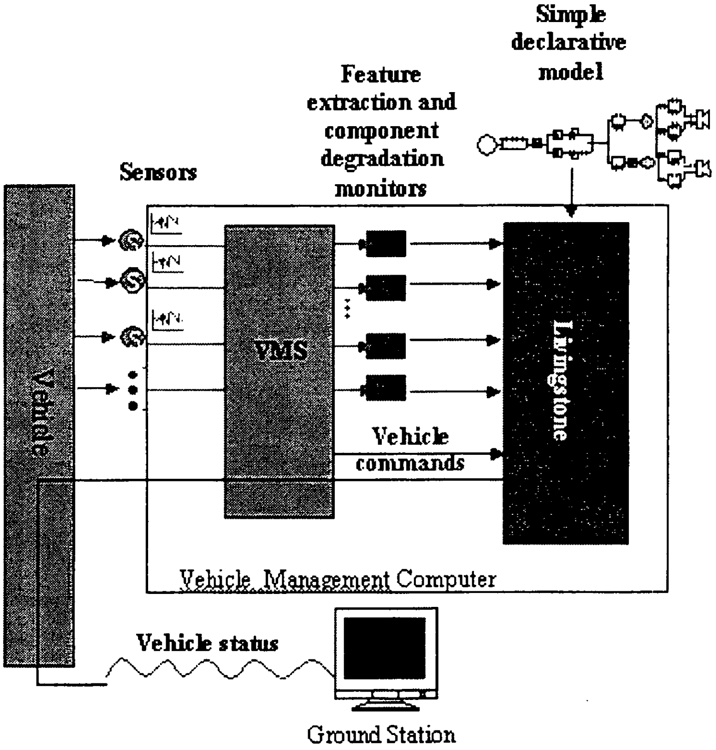
3.3.3. Program Software[8]

The IVHM software consists of Livingstone 2.3 under the VxWorks 5.4 operating system running on PowerPC microprocessors. It is being designed to do the following:

- Perform diagnosis using qualitative, Model-based Reasoning
- Search system-wide interactions to detect and isolate failures
- Reason about complex system interactions within a real-time monitoring and control loop, rather than requiring an engineer to reason through all possible interactions and 'hardwire' the appropriate response to a pre-defined set of failures. ***Note:*** *Updating and verifying the model is straightforward and less labor intensive than the task of identifying changes required in explicit procedural code.*
- Streamline software development process and maximize code reusability across vehicles
- Facilitate the generation of an explanation or justification of the diagnosis, allowing the human operator to decide whether the diagnosis is reasonable before selecting or confirming the appropriate recovery action

As shown in Figure 6, the IVHM software will run as a task on the Vehicle Management Computer (VMC). The Vehicle Management System (VMS) will be another task running on the VMC. The VMS will be responsible for telemetry and power management. The IVHM task will communicate with the VMS task in order to obtain sensor data and vehicle commands, and to send telemetry to the ground.

Figure 6 – X-37 Hardware and Software [8]



Scope of the Experiment [8]

- Monitor and diagnose Electro-mechanical Actuators and associated Electrical Power System components
- Real-time fault detection and isolation
- Diagnosis, not prognosis
- Shadow mode only (no reconfiguration commands)
- Generate advisory recommendations for ground operations

Challenges [8]

- Limited resources (CPU, memory, telemetry bandwidth)
- Rigorous software safety standards

3.3.4. Organization Performing V&V [9]

NASA Ames Research Center

3.3.5. Verification & Validation Overview [9]

Verification and Validation exercises were combined for the X-37 IVHM Experiment. Specific V&V activities have been defined for each phase of software development. Software development phases are listed below:

- Software Requirements Specification Phase
- Software Architectural (Preliminary) Design Phase
- Software Detailed Design Phase
- Software Implementation Phase

- Software Integration and Test Phase
- Software Acceptance and Delivery Phase
- Software Sustaining Engineering and Operations Phase (revision and update of software)

For purposes of this survey, verification methods were separated from validation methods. Verification Methods are described in section 3.3.6 and Validation Methods are described in section 3.3.7.

3.3.6. Verification Methods[9]

Verification methods include the following:

- Informal Reviews as needed. Software Developer or Livingstone Modeler may select the review panel and present informal material like computer listings or hand-written documentation.
- Official Reviews at the end of each life cycle phase by an official review panel empowered to make "go/no go" decisions. Official reviews include:
    - Software Requirements Specification Review
    - Software Preliminary Design Review
    - Software Critical Design Review
    - Software Test Readiness Review
- An Acceptance Review is conducted at the end of the project to indicate customer acceptance of the product

3.3.7. Validation Methods[9]

Validation methods include informal and official testing described below.

- Informal Testing performed by Software or Livingstone Model Developer to measure progress and find errors. Witnessing of these tests is not required. Tests include the following:
    - Unit Tests
    - Component Tests to ascertain the following:
        - Computational correctness
        - Proper handling of boundary conditions like extreme inputs/conditions that cause extreme outputs
        - Expected state transitioning
        - Proper behavior under stress
        - Adequate error detection, handling and recovery
    - Subsystem Integration Tests
    - Black Box testing or requirements-driven testing. (Select system input and observe system output/reactions)
    - White Box testing or design-driven testing (examine internal workings of code) to ascertain the following:
        - Correctness of all paths through the code or Livingstone model
        - Bit-by-bit functioning of interfaces
        - Size and timing of critical elements of software code
- Official Testing demonstrates that software and Livingstone models are ready for intended use. It includes the following:
    - IVHM team approved Test Plan and Procedure
    - Quality Assurance (QA) witnesses
    - Record of discrepancies
    - Test Report

3.3.7.1. Testing Environment [9]

Tests were and will be distributed among low and high fidelity testbeds described in Figure 7 below:

Figure 7 – Testbeds [9]

Testbed – Hardware	Fidelity	CPU Speed	Operating System	Tested
PowerPC 603e	High	240 MHz	Flight (VxWorks 5.4)	Not at this time – success or failure inferred from 120 MHz test
Force Computer PowerCore 6603e	High	120 MHz MPC603e	Flight (VxWorks 5.4)	Yes
Dell Dimension XPS B600r	Low		Simulator (Windows 98)	Yes

3.3.7.2. Testing Tools [9]

The following testing tools will be used:

- Concurrent Version System (CVS) – can record the history of source files. If a bug or "side-effect" occurs when software or Livingstone models change, old source code can be reviewed to determine which change caused the bug.
- Bug Tracking Software (GNATS) – several utilities for formulating and administering a database of Problem Reports (PR)
- CodeTEST for Tornado with coverage and memory tool. The coverage tool can pinpoint untested code and identify additional test cases to reduce likelihood of missing this code in future tests. The memory tool can examine dynamic memory management behavior.

3.3.7.3. V&V Team [9]

Two testers will be required: One to test models and one to test software integration

V&V Team Training [9]

All testers listed above must be trained on testing procedures

3.3.7.4. Testing Methods/Procedures[9]

The IVHM team will conduct official inspections involving line-by-line evaluation of the software. A team composed of members with specific roles described below will perform inspections:

- Moderator – facilitates meeting
- Reader - leads team through the item
- Reviewers - look for faults
- Recorder - notes faults
- Author - explains the item being inspected

The above described inspection process may be applied to software design, code, documentation and software product. A sample Code Inspection form is included in Appendix I, Code Inspection Form.

Types of tests

The following types of tests will be conducted:

- Recovery Testing – Livingstone stopped at various times when sharing memory etc and re-started
- Performance Testing – measures time for Livingstone engine to make a diagnoses when under stress
- Code Coverage – every line of software will be checked by CodeTEST
- Memory Usage – memory overwrites etc checked by CodeTEST

- Error Handling – checks graceful handling of errors
- Regression – to test for side effects from code or Livingstone model changes

3.3.8. Methods for Analyzing Test Results [9]

Tests will be analyzed using Pass/Fail criteria where passing is defined as

- System doesn't hang
- System doesn't crash
- Program doesn't write outside its memory allocation
- Software doesn't harm vehicle
- Livingstone makes correct diagnosis

3.3.9. Review Methods

Reviews will be conducted during and at the end of each phase of the Software Life Cycle to determine whether established requirements, design concepts and specifications have been met. Reviews consist of the presentation of material to the IVHM Team and customer. Reviews are most effective when conducted by personnel who have not been directly involved in the development of the software or Livingstone models being reviewed.

Specific reviews are listed in Section 3.3.6 and copies of the associated review forms have been included in Appendix H, Review Forms.[9]

3.3.10. Success of V&V Process [10]

The X-37 IVHM Experiment is still in the early stages. To date, the project team has developed a comprehensive V&V Plan based on NASA standards and lessons learned from Deep Space One Remote Agent Experiment.

Acceptance tests were run on May 22, 2001 from 09:30 to 19:00 at Moffett Field, California with the following results:

- Version 2.3 of IVHM software was able to run a nominal scenario script for 4.5 days without crashing
- Isolation of a single component failure resulted in Livingstone finding the failure
- Completion of the following software is required before the next acceptance test on October 17, 2001:
  - Integration of monitors and I/O queue software and Livingstone software
  - Fixes to Livingstone software
  - Ground Processing Unit software
  - Version upgrade of CodeTEST[11]

Suggestions for Improving V&V

No suggestions available because the X-37 IVHM Experiment is still in the early stages.

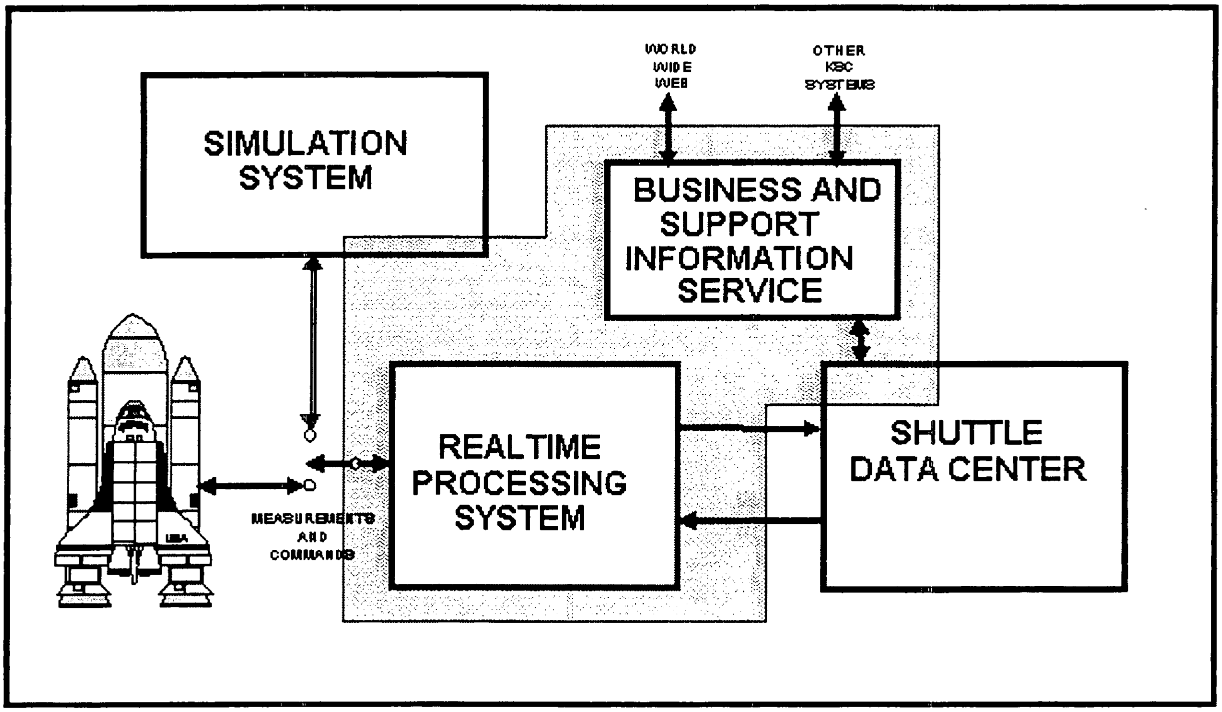
3.4. CLCS – Checkout and Launch Control System

Information, diagrams and pictures presented below are from references 2 and 12 - 16 in the references section.

3.4.1. Program Description [12]

The objective of the CLCS Project is to replace major portions of the Launch Processing System (LPS), the current Space Shuttle Ground computer checkout system. The CLCS system will be a real-time computerized checkout system used throughout Shuttle processing to control and monitor test operations and launch as shown in Figure 8.

Figure 8 – CLCS Subsystems [12]


WORLD WIDE WEB
OTHER KSC SYSTEMS
SIMULATION SYSTEM
BUSINESS AND SUPPORT INFORMATION SERVICE
REALTIME PROCESSING SYSTEM
SHUTTLE DATA CENTER
MEASUREMENTS AND COMMANDS
USA

The four CLCS subsystems are explained below:

- Real-time Processing System (RTPS) provides the capability to monitor and control the elements of the Space Shuttle flight vehicle and Ground Support Equipment (GSE)
- Shuttle Data Center (SDC) records and archives test data, hosts various databases and provides the capability to build test packages for configuration of the RTPS
- Simulation System provides the capability to debug and certify RTPS software and to aid in training the test team
- Business and Support Information Service (BASIS) provides connectivity and access from support workstations to non-RTPS applications [13]

3.4.2. Program Maturity

CLCS was recently reorganized. To date only the Hypergolic Maintenance Facility has been validated. Results of this validation will be available Fourth Quarter, 2001.

3.4.3. Program Software

Software developed for CLCS was divided into two major layers described below with Applications Software running on System software:

Applications Software

- User Displays
- Automated Applications
- Web-based data analysis tools [12]

System Software

- Functionality to acquire and distribute time, data and commands
- Interface application programs and displays to Operating Systems and Network Architecture, System Configuration, and Health Operations & Maintenance features [12]

Other layers of CLCS software include the following:

Hardware & Networks

- Data Distribution Processors (DDP)
- Command & Control Processors (CCP)
- Command & Control Workstations (CCWS)
- Gateways
- Data Recording Processors (DRP)
- Support Workstations

Support Systems

- Building software loads
- Model support for Simulation
- Recording & Retrieval

3.4.4. Organization Performing V&V

Qualified engineers perform V&V for Application and System Software. The NASA Independent Verification & Validation (IV&V) team is performing IV&V for System Software. [14]

3.4.5. Verification & Validation Overview

Because CLCS is a very large project, V&V documents were reviewed and the following were selected as being representative of the V&V effort:

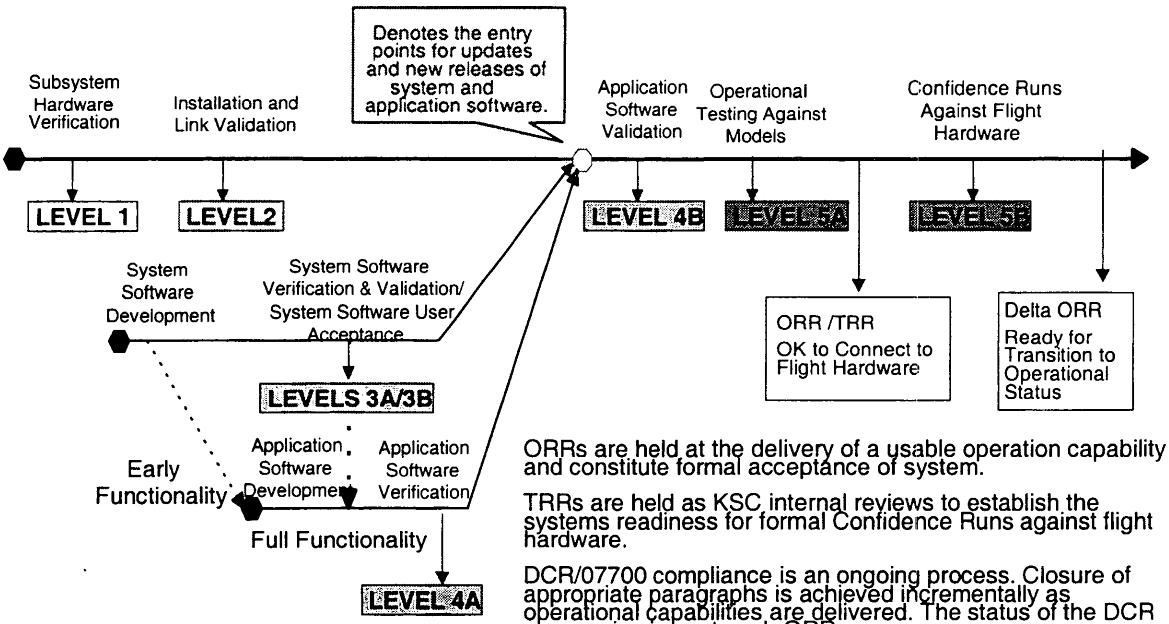
- Application Control Board (ACB) Charter Level IV Change Control Board (CCB), Checkout and Launch Control System (CLCS) Document Revision A, dated August 23, 2000
- System Validation Plan, Checkout and Launch Control System (CLCS) 84K07490-000-01, dated July 31, 2001
- CCWS Design Verification Test (DVT), Command and Control Workstation Checkout and Launch Control System (CLCS) 84K06548-005-02, dated August 24, 2000

Other V&V documents, called TEST SPECS, may be found on the following website: http://clcs.ksc.nasa.gov/docs/test-specs.html. This website is part of an online document management system that provides users the capability to search through project documents using key words.

To cope with the size and complexity of the CLCS project, V&V activities were grouped into Testing Levels described in Figure 9. This figure also shows the flow of testing through each Testing Level. For example, hardware testing begins at Level 1 with Subsystem Hardware Verification and continues to Level 2, Installation and Link Validation.

Figure 9: Testing Levels and Flow [12]

Testing Levels and Flow


Denotes the entry points for updates and new releases of system and application software.
Subsystem Hardware Verification
Installation and Link Validation
Application Software Validation
Operational Testing Against Models
Confidence Runs Against Flight Hardware
LEVEL 1
LEVEL2
LEVEL 4B
LEVEL 5A
LEVEL 5B
System Software Development
System Software Verification & Validation/ System Software User Acceptance
LEVELS 3A/3B
ORR /TRR
OK to Connect to Flight Hardware
Delta ORR
Ready for Transition to Operational Status
Early Functionality
Application Software Development
Application Software Verification
Full Functionality
LEVEL 4A

ORRs are held at the delivery of a usable operation capability and constitute formal acceptance of system.

TRRs are held as KSC internal reviews to establish the systems readiness for formal Confidence Runs against flight hardware.

DCR/07700 compliance is an ongoing process. Closure of appropriate paragraphs is achieved incrementally as operational capabilities are delivered. The status of the DCR process is given at each ORR.

HARDWARE	INSTALLATION	SYSTEM SOFTWARE		APPLICATION SOFTWARE		SYSTEM	
LEVEL 1 Process that ensures delivered COTS & custom hardware meets spec and operates correctly. Custom subsystem meets derived subsystem level spec and requirements	**LEVEL 2** Process verifying installed hardware and COTS software at site correct for operation & subsystems & networks. Includes verification of all installed external connections to ensure correct signal routing & control & levels	**LEVEL 3A** Process verifying the DP2 software requirements allocated to either CIT or System Test and verifying the CLCS functional and performance spec and requirements in a production set	**LEVEL 3B** Process of ensuring the system level user perspective is supported in system validation activities	**LEVEL 4A** Process of verifying the application software to ensure it meets product and system level functional and performance spec and requirements in a development environment	**LEVEL 4B** Process of validating the application software to ensure it meets system level functional and performance spec and requirements in a full up configuration using models	**LEVEL 5A** Process of validating the system in operational mode using models and GSE not connected to flight hardware	**LEVEL 5B** Process of performing confidence runs to validate system in operational mode connected to flight hardware
Hardware Test	**HIT/HVT**	**System Software Verification**	**System Software Validation**	**Application Software Verification**	**Application Software Validation**	**User Acceptance**	

Acronyms used in Figure 9 are listed below in alphabetical order:

CIT	- CSCI (Computer Software Configuration Item) Integration Test
COTS	- Commercial Off The Shelf
DCR	- Design Certification Review
DP2	- Design Panel 2 – Requirements Design Panel
GSE	- Ground Systems Equipment
H/W	- Hardware
HIT	- Hardware Installation Test
HVT	- Hardware Validation Test
ORR	- Operational Readiness Review
SW	- Software
TRR	- Test Readiness Review

3.4.6. Verification Methods

Verification will be accomplished via one of three possible means: 1) inspection, 2) demonstration, 3) Vendor documentation (Certificate of Compliance falls under the Vendor documentation section). In some instances, requirements may need to be verified by a combination of techniques.

Detailed verification test documents have been written for each component of CLCS and include the following types of verification:

- Hardware Verifications
- System Software Verifications
  - Informal Reviews
  - Official Reviews
    - Design Verification
    - Early User Evaluation conducted during development
    - User Evaluation conducted during integration
    - Test Readiness Review
- Power Specification Verifications
- Cold Boot Verifications
- Human Factors Verifications
- Reliability, Maintainability, Availability (RMA) Verifications
- Safety Verifications
- Design And Construction Verifications
- Other Verifications depending upon the component

3.4.7. Validation Methods

The CLCS Validation Process is shown in Figure 10 and described below.

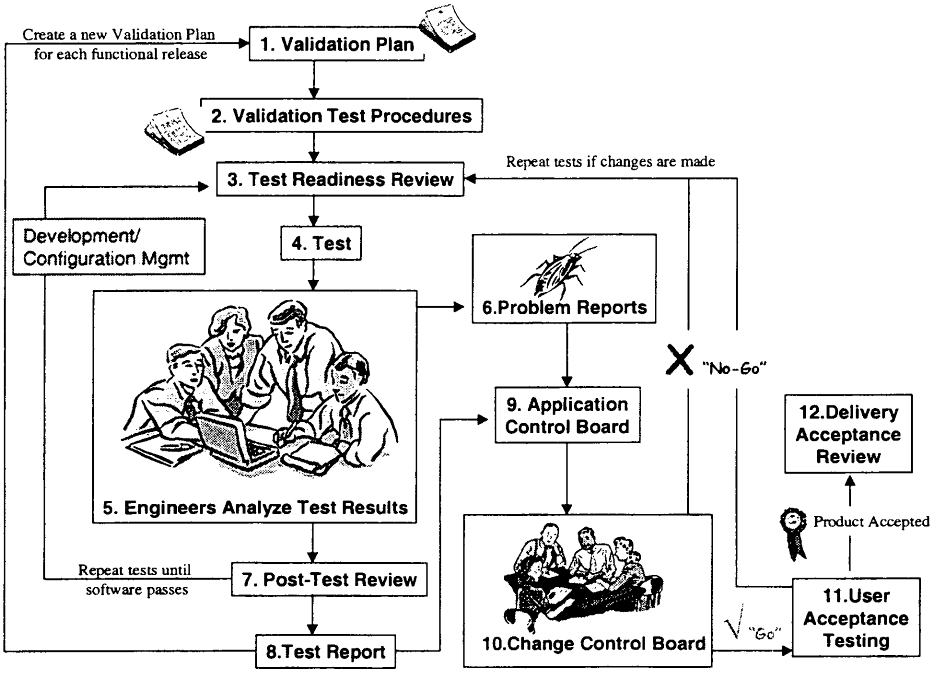

Create a new Validation Plan for each functional release
1. Validation Plan
2. Validation Test Procedures
3. Test Readiness Review
Repeat tests if changes are made
Development/ Configuration Mgmt
4. Test
6.Problem Reports
X "No-Go"
9. Application Control Board
12.Delivery Acceptance Review
5. Engineers Analyze Test Results
Product Accepted
Repeat tests until software passes
7. Post-Test Review
11.User Acceptance Testing
√ "Go"
8.Test Report
10.Change Control Board

Figure 10: CLCS Validation Process

1. Validation Plan is developed for each functional release. The System Validation Plan[14] dated July 31, 2001, was used as a sample for survey purposes and includes the following types of information.

- Validation Test Objectives [14]
  - Validate the operational capabilities of a delivered system software baseline to ensure that the baseline is in an acceptable state, ready to support official application validation testing (Level 4B).
  - Validate that the hardware and software delivered meets or exceeds the system user's needs with respect to operability, functionality, reliability, and stability.
  - Comprehensively test the CLCS software as a total system. This testing should include system failure scenarios, typical system load conditions, and proper operational response for normal and error environments.
  - Perform regression tests on system software updates prior to release of those updates back into an operational environment.
- Validation Ground-rules [14]
  - Testbeds or hardware to be used
  - Software releases
  - Safety constraints
  - Description of discrepancies found during testing
  - Designation of Test Conductor (TC)
  - Explanation of TC responsibilities

2. Validation Test Procedures [14] are written including the following:

- Test scope and intent (e.g., delivery identification, any deviations from planned test scope, relationship to other test documents).
- Test cases, procedure steps and test dependencies (e.g., test tools, special configurations)
- Specific test results and sign off

3. Test Readiness Review (TRR) is performed for the major software delivery milestones. The goal of this review is to obtain concurrence from the CLCS and user communities that the system is ready for validation testing. This concurrence provides authority to commence testing.

4. Test - A Test Conductor is appointed to coordinate the test. Using Software Validation Procedures (SVP), engineers and customers perform the Official Validation.

5. Engineers analyze test results by comparing actual results to expected results. The Test Conductor (TC) provides a recommendation about the suitability of the system software to support further testing or to be released as a deliverable.

6. Problem Reporting. Official Problem Reports are generated and tracked using the RAZOR tool. They are reported to CLCS project management on a regular basis and included in delivery documentation.

7. Post-Test Review (PTR) may be held after testing. This review will summarize the testing completed, problems found during testing, and any follow up actions required to declare the successful completion of the test.

8. Test Report (TR) will be released after the completion of all Validation Testing and will be included in the delivery package for the given system delivery. This report will include the following:

- Summary of the test procedures
- Test results
- Recommendation about the suitability of the System Software
- Problems found
- Copy of the "as run" test procedures, including any additions or modifications to the procedures generated during the test

This report is the 3B Validation organization's input to the Delivery Acceptance Review (DAR).

9. Application Control Board (ACB)
The ACB consists of representatives from the following organizations:

- CLCS Real Time Control (RTC) Application Software Engineering
- CLCS System Engineering
- NASA and United Space Alliance (USA) Sustaining Engineering
- NASA and United Space Alliance (USA) Shuttle Engineering

The CLCS RTC Application Software Manager or delegated representative will serve as the ACB Chairperson and is the final approval authority for this Board. The ACB has the authority to approve application software standards. All deviations and waivers to the approved standards will require CCB approval.

The ACB also has authority to implement and maintain the Configuration Management (CM) of all Software Requirements Specifications documentation (including any documents where changes have potential impact to multiple organizations or design activities) and process flows, as well as, the authority to establish and maintain product baselines with certain restrictions.

The ACB is responsible for

- Ensuring baseline changes are processed in a timely manner with appropriate assessments of cost, schedule, technical, and risk impact across all teams and working groups
- Maintaining appropriate documentation and records and a list clearly identifying members with voting authority
- Changes to the hardware and/or software under development that will impact operational hardware and/or software

10. Change Control Board (CCB)
The NASA CLCS Project Manager chairs a Change Control Board (CCB). The CCB must approve any Application Control Board (ACB) changes exceeding defined cost or schedule limits. [15] The CCB structure is shown in Figure 11. [15]

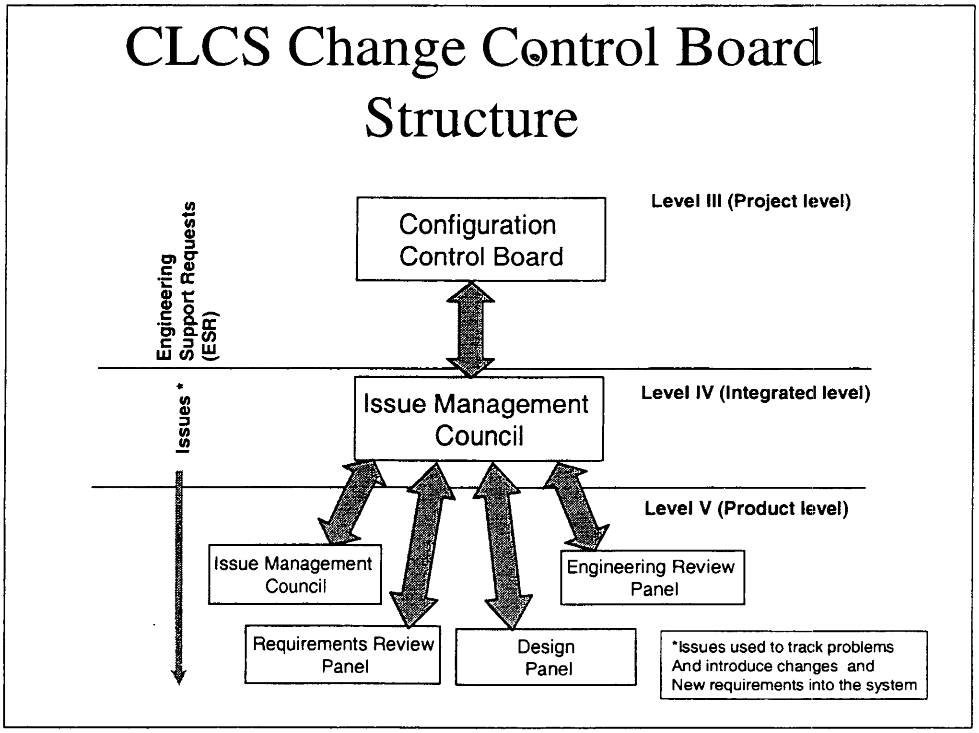

CLCS Change Control Board Structure
Engineering Support Requests (ESR)
Issues *
Configuration Control Board
Level III (Project level)
Issue Management Council
Level IV (Integrated level)
Level V (Product level)
Issue Management Council
Engineering Review Panel
Requirements Review Panel
Design Panel
*Issues used to track problems And introduce changes and New requirements into the system

Figure 11: CLCS Change Control Board

11.User Acceptance Testing is an official test performed by end users and is a key part of the system certification process.

12. Delivery Acceptance Review (DAR) is a meeting, chaired by the CLCS Delivery Manager, at which the delivery package is reviewed and the Delivery is declared complete. The delivery package includes such things as magnetic tape copies of software to be delivered, all development documentation (users guides, requirements, design documents), all test documentation (subsystem and system level tests) and known problems.

3.4.7.1. Testing Environment

Figure 10 shows the different environments including the following:

- Development – environment where developers build individual software components
- Integration – environment where individual components are combined into a system
- Operations – production environment where system works as specified

Testing begins with Design Verification/Unit Test (top left) and goes through User Acceptance Testing (bottom right). Multiple boxes indicate the activity occurs more than once.

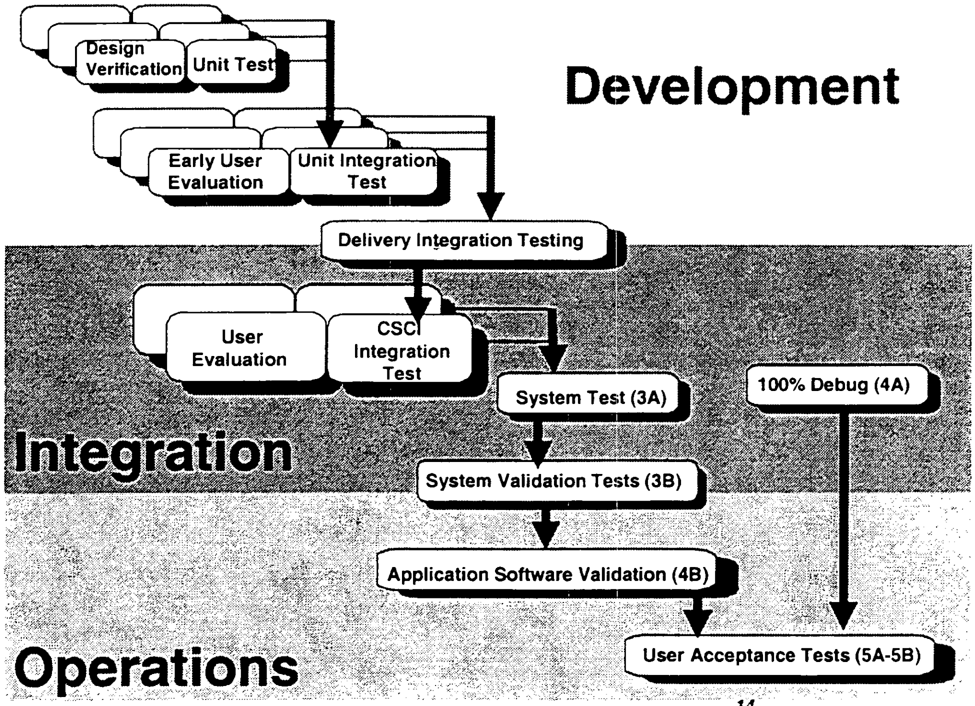


Figure 12: CLCS Testing Environments [14]

Acronyms used in Figure 12 are listed below in alphabetical order:

- CSCI – Computer Software Configuration Item (a term used in NASA or Military standards to describe a product like a jet engine or a computer system. CSCI in Figure 10 refers to a functional release of software)

3.4.7.2. Testing Tools

The new CLCS architecture, will be tested using the following tools:

- ParaSoft (www.parasoft.com) including the following components:
  - Insure memory checker
  - Total Coverage Analysis
  - Code Wizard for static analysis of code
  - C++ Test for automated unit testing (not confirmed at this time) [15]

Note: *The tools listed below are also used on the CLCS project:*

- *TogetherSoft to capture our UML design models*
- *Razor for configuration management*
- *DOORS to capture our requirements* [15]

3.4.7.3. V&V Team [9]

Qualified engineers perform V&V. The NASA Independent Verification & Validation (IV&V) team is performing IV&V for System Software. [14]

3.4.7.4. Testing Methods and Procedures

Testing Methods and Procedures were documented in Section 3.4.7 Validation Methods.

3.4.8. Methods for Analyzing Test Results

Engineers analyze tests to ensure test results match expected results. If test results do not match expected results, the Test Conductor must ensure that appropriate documentation is generated. This documentation is called "Problem Reports". Problem Reports must include proper classification of bugs, anomalies, or issues, as well as, the impact of these problems. Testing may be suspended and rescheduled depending upon the severity of the problem. Retesting will be scheduled if one or more of the following occurs:

- Excessive number of bugs, anomalies or issues
- Failure of software to meet expected operability, stability and reliability expectations

3.4.9. Review Methods

Review methods are explained in detail in Section 3.4.7 and summarized below:

- Peer Review conducted on individual software components or a group of integrated components
- Test Readiness Review conducted on a software product deemed ready for testing
- Post Test Review to summarize the results of the test
- Review of each delivery package by the appropriate change control board described in Section 3.4.7
- Delivery Acceptance Review conducted on the final delivery package

3.4.10. Success of V&V Process

In 1999, it became apparent that the Atlas software delivery (part of CLCS) was late and over budget. An Independent Assessment Team reviewed CLCS processes and recommended improvements. As a result the CLCS project was re-structured with an emphasis on functional releases rather than system software deliveries. ***(Note:*** *The Survey includes information from the re-structured CLCS project.*) The following were also implemented:

- Increased Direct Contractor Accountability
  - "Broke out" development activity to make contractors more accountable
  - Improved an award/performance fee approach
- Established System Engineering and Integration Team
  - Provided project-wide system engineering and technical integration
  - Provided focal point for systems architecture and requirements (Chief Engineer)
  - Provided focal point for system integration, delivery and user interface (Operations Integration Manager)
- Reviewed Project Execution Approach
  - Clearly defined roles and responsibilities across the project
  - Reviewed and improved project engineering and management processes as required
  - Coordinated frequently with KSC and Shuttle management teams
- Attained Shuttle System Engineering Resource Commitments
  - Developed agreements with NASA/USA to ensure resource availability
- Established Project Advisory Council
  - Ensured contractor Project Manager's participate in decision-making
  - Improved top down communication on key issues

The CLCS project has comprehensive V&V plans based on NASA standards and contained in an online repository (http://clcs.ksc.nasa.gov/docs/test-specs.html). To date, only the Hypergolic Maintenance Facility has been validated under the new project structure.[16] Specific information regarding lessons learned from V&V of the Hypergolic Maintenance Facility will be available in third quarter 2001.

Suggestions for Improving V&V

One important lesson learned to date is the necessity for evaluating IV&V budget requirements early in the project before the budget is frozen. A manned mission and any mission or program costing more than $100M will require IV&V. Appendix D, "Independent Verification and Validation (IV&V) Criteria" contains recent criteria for this evaluation.[17]

Other suggestions include the following:

- Use a combination of the Spiral and Waterfall methodologies when presenting new technology. V&V is different for each methodology as described in Section 5.
- Enforce accountability between partners to ensure V&V is performed effectively

4. APPLICABILITY OF NASA V&V TO 2nd Generation RLV IVHM

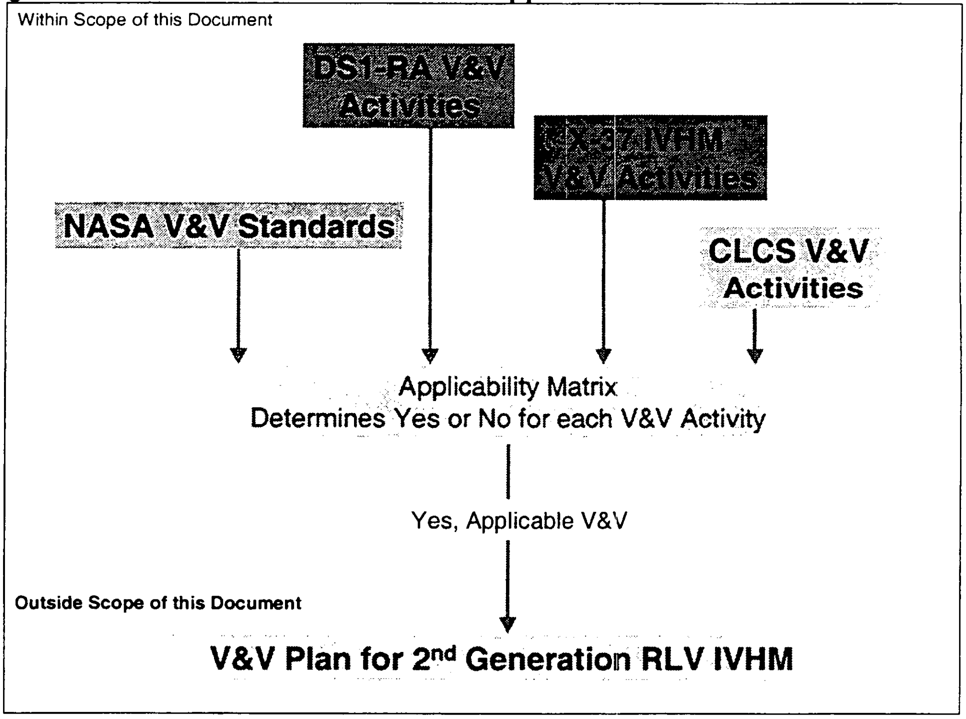
The purpose of the Applicability of NASA V&V to 2nd Generation RLV IVHM is to determine which NASA V&V processes and methods are applicable or beneficial to the IVHM system planned for the 2nd Generation RLV.

In order to properly evaluate the applicability of V&V to IVHM, the following assumption was made:

- IVHM will include functionality described in the following document:

2nd Generation RLV, TA-5 IVHM, NASA In-House Task 1, IVHM Concept of Operations Document

Figure 13: Derivation of V&V Activities Applicable to 2nd Generation RLV IVHM


Within Scope of this Document
DS1-RA V&V Activities
X-37 IVHM V&V Activities
NASA V&V Standards
CLCS V&V Activities
Applicability Matrix
Determines Yes or No for each V&V Activity
Yes, Applicable V&V
Outside Scope of this Document
V&V Plan for 2nd Generation RLV IVHM

As shown in Figure 13, in order to provide a comprehensive list of V&V tasks, NASA V&V standards were reviewed, as well as, V&V activities described in the Surveys of the Deep Space One Remote Agent Experiment, X-37 IVHM Experiment and CLCS. Each V&V activity was categorized and listed in a table called an Applicability Matrix. Tasks designated as being applicable may be incorporated into the V&V Plan for the 2nd Generation RLV IVHM system.

This section includes the following:

- Overview of NASA V&V Standards
- Applicability Overview
- V&V Processes including a description of V&V Processes, applicability criteria and V&V Processes Applicability Matrix
- V&V Methods including a description of V&V Methods, applicability criteria and V&V Methods Applicability Matrix

4.1.1. Overview of NASA V&V Standards

Specific NASA V&V Standards include the two documents listed below:

- NASA Procedures and Guidelines (NPG) 2820.DRAFT, NASA Software Guidelines and Requirements[18]
- NASA Procedures and Guidelines (NPG) 8730.DRAFT 2, Software Independent Verification and Validation (IV&V) Management [18]

NPG 2820.DRAFT references the following IEEE/EIA Standards[19] developed in accordance with ANSI:

- *12207.0 - Standard for Information technology – Software Life Cycle Processes (March, 1998)12207.1 - Standard for Information technology – Software Life Cycle Data (April, 1998)*
- *12207.2 - Standard for Information technology – Software Implementation Considerations (April, 1998*

The IEEE documents reference the ISO and IEC standards published as ISO/IEC 12207 in 1995.

In order to effectively use the NASA standards, they must be tailored to each specific project. Appendix F includes an Introduction to IEEE 12207.0 with suggestions for tailoring it to 2nd Generation RLV IVHM.

In addition to the NASA standards, DO-178B, "Software Considerations in Airborne Systems and Equipment Certification" contains guidance for determining that software aspects of airborne systems and equipment comply with airworthiness certification requirements. Written in 1980 by the Radio Technical Commission for Aeronautics (now RTCA, an association of aeronautical organizations of the United States from both government and industry), it was revised in 1985 and again in 1992. During the 1992 revision, it was compared with international standards: ISO 9000-3 (1991), "Guidelines for the Application of ISO 9001 to the Development, Supply and Maintenance of Software" and IEC 65A (Secretariat) 122 (Draft – 11-1991), "Software for Computers in the Application of Industrial Safety-Related Systems" and considered to generally satisfy the intent of those international standards.

NASA V&V Standards Acronyms:

ANSI	- American National Standards Institute
EIA	- Electronic Industries Association
IEC	- International Electro-technical Commission
IEEE	- Institute of Electrical and Electronics Engineers
ISO	- International Organization for Standardization
NPG	- NASA Procedures and Guidelines

4.1.2. Applicability Overview

V&V can be divided into the two categories listed below[21]:

- V&V Processes – Management tasks like the following:
  - Planning (Project Plans, V&V Objectives, Tasks required for Certification, etc)
  - Organization (Staff, Project Meetings, Status Meetings, etc)
  - Documentation (V&V Plans, Test Plans etc)
  - Metrics (Project Statistics)
- V&V Methods – Technical tasks like the following:
  - Engineering Reviews (analyzing, evaluating, etc)
  - Testing Methods (engineering techniques and tools)
  - Testing Environment (testbeds, etc)

4.1.3. V&V Processes

The following table called the V&V Processes Applicability Matrix, contains a combination of V&V activities from the items listed below:

- Survey of Current NASA V&V Processes/Methods (Section 3 above)
- RTCA/D0-178B [20]
- NASA Software Guidelines and Requirements [18]

It includes the columns described below:

- Applicability to IVHM – contains either "Yes" or "No". "Yes" designates this task is applicable to IVHM. "No" indicates this task is not applicable to IVHM. Tasks were evaluated as being applicable if they have been proven necessary to mission success or believed to improve the likelihood of mission or project success.
- Task – briefly describes the task
- Project – contains the project identifier listed below
  - DS1-RAX - Deep Space One Remote Agent Experiment
  - X-37 - X-37 IVHM Experiment
  - CLCS - Checkout and Launch Control System
- Notes – any additional information regarding this task or its applicability to IVHM

Tasks are grouped into the following categories and some tasks contain examples from the Survey:

- Planning
- Organization
- Documentation
- Metrics

4.1.3.1. V&V Processes Applicability Matrix

Applicable to IVHM	Task	Project	Notes
	PLANNING		
Yes	Plan to conduct V&V throughout the Software Life Cycle. The Software Life Cycle is defined as the steps or phases required to develop software, starting with a concept and ending with a working product or system. Standard Software Life Cycle phases are described in IEEE 12207.0, a very comprehensive guide including five primary processes, eight supporting processes and four organizational processes. Therefore, it must be tailored for each project. Because some of these processes are applicable to 2nd Generation RLV IVHM, Appendix F contains an introduction to IEEE 12207.0 and recommendations for tailoring this standard. Additional V&V may also be needed for 2nd Generations RLV IVHM as described in Section 3, *Special V&V for IVHM.* Appendix C includes a diagram showing the relationship between Software Life Cycle phases and V&V activities. Appendix G contains a summary of specific V&V activities for each phase of the Software Life Cycle. These activities are noted here and included as appropriate in subsequent tasks. Section 1.2 of this document contains an example of how the standards were tailored for the X-37 IHVM experiment, a smaller, experimental project. Section 1.3 of this document contains an example of how the standards have recently been re-tailored for the CLCS project, a large, complex software development project.	DS1-RAX X-37 CLCS	

Applicable to IVHM	Task	Project	Notes
Yes, if appropriate	Plan to conduct independent V&V (IV&V) for projects meeting criteria described in Appendix D and other projects as appropriate depending upon cost, size, complexity, life span, risk and consequences of failure. Independent V&V are V&V processes/methods performed by an organization that is "not related to" or "independent of" the organization developing software. This ensures no conflict of interest or other hindrance to finding and reporting bugs or anomalies.	CLCS	For more information: *NPG 8730.DRAFT2 Software Independent Verification and Validation (IV&V) Management dated 2/26/01*
Yes	Define and document specific, detailed verification objectives. See Appendix G for specific verification information.	DS1-RAX X-37 CLCS	Verification is *"the process of evaluating a system or component during or at the end of the development process to determine whether it satisfies specified requirements"*[21]
Yes	Define and document specific, detailed validation objectives. See Appendix G for specific validation information	DS1-RAX X-37 CLCS	Validation is *"the process of evaluating a system or component to determine whether the products of a given development phase satisfy the conditions imposed at the start of that phase"* [21]
Yes	Create an information website including an online document manager	DS1-RAX X-37 CLCS	

Applicable to IVHM	Task	Project	Notes
ORGANIZATION			
Yes	Appoint a V&V Project Manager to ensure the following: • V&V requirements are followed • Adequate V&V staff available and trained • Communication of problems, failures and anomalies to affected project teams • Problems are tracked to closure	DS1-RAX X-37 CLCS	
Yes	For smaller projects create a Change Control Board (CCB) to review bugs and manage change. For larger projects, create a hierarchy of Change Control Boards and clearly communicate and document the authority and responsibility of each CCB.	DS1-RAX X-37 CLCS	For More Information: DS1-RAX *[Section 3.2.8]* X-37 *[Section 3.3.7]* CLCS *[Section 3.4.8]*
DOCUMENTATION			
Yes	Verification Plan. Appendix G includes the IEEE 12207.0, Paragraph 6.4 standards that describe specific content for the Verification Plan.	DS1-RAX X-37 CLCS	
Yes	Software Verification Results Report. Appendix G includes the IEEE 12207.1 Paragraph 6.23 standards that describe specific content for the Software Verification Results Report		
Yes	Validation Plan. Appendix G includes the IEEE 12207.0, Paragraph 6.5 and IEEE 12207.1 Paragraph 6.27 standards that describe specific content for the Validation Plan. In addition to the above mentioned IEEE guidelines, include end-to-end testing plans (Operations Scenarios like DS1-RAX) per NPG: 2820.DRAF1[21]	DS1-RAX X-37 CLCS	Similar to MIL STD 498 Software Test Plan (STP)[22]

Applicable to IVHM	Task	Project	Notes
Yes	Software Test Description describes the test preparations, test cases and test procedures [23] It also includes information like the following to ensure that all tests are traceable to requirements: • Trace-ability between the software requirements and test cases is accomplished by the requirements-based coverage analysis[2] • Trace-ability between the code structure and test cases is accomplished by the structural coverage analysis[2]	DS1-RAX	Similar to MIL STD 498 Software Test Description (STD)[23]
Yes	Test or validation procedures as described in IEEE 12207.1 Paragraph 6.28 included in Appendix G		
Yes	Problem Reports (PR) used to track and report bugs **Note:** *CLCS used RAZOR tool for PR*	DS1-RAX X-37 CLCS	For more information: DS1-RAX *[Section 3.2.8]* X-37 *[Section 3.3.7]* CLCS *[Section 3.4.8]*
Yes	Test Report should be released after the completion of official testing and should be included in the delivery package for the given system. This report includes the following: • Summary of the test procedures • Test results • Recommendation about the suitability of the System Software • Problems found • Copy of the "as run" test procedures, including any additions or modifications to the procedures generated during the test	X-37 CLCS	For more information: • CLCS *[Section 3.4.8]* • See Appendix G - IEEE 12207.1 Paragraph 6.29 Test or Validations Results Report • Similar to MIL STD 498 Software Test Report (STR)

Applicable to IVHM	Task	Project	Notes
Yes	Post Mortem or Project Review Document including "lessons learned" and ideas for improvement	DS1-RAX	
METRICS			
Yes	Compute Metrics per NASA requirements listed in Appendix E [21]		
Yes	Enter Metrics in NASA Software Metrics web site: http://swmetrics.gsfc.nasa.gov/index.cfm		Metrics must be entered on a quarterly basis within 45 days of the end of the quarter [18]

4.1.4. V&V Methods

The following table, called V&V Methods Applicability Matrix, contains a combination of V&V Methods from each mission/project in the Survey of Current NASA V&V Processes/Methods. It includes the columns described below:

- Applicability to IVHM – contains either "Yes" or "No". "Yes" designates this task is applicable to IVHM. "No" indicates this task is not applicable to IVHM. Tasks were evaluated as being applicable if they have been proven necessary to mission success or believed to improve the likelihood of mission or project success.
- Task – briefly describes the task
- Project – contains the project identifier listed below
  - o DS1-RAX - Deep Space One Remote Agent Experiment
  - o X-37 - X-37 IVHM Experiment
  - o CLCS - Checkout and Launch Control System
- Notes – any additional information regarding this task or its applicability to IVHM

Tasks are grouped into the following categories and some tasks contain examples from the Survey:

- Verification Methods
- Validation Methods
- Types of Tests
- Test Criteria
- Test Environment
- Testing Tools
- Tests and Reviews

4.1.4.1. V&V Methods Applicability Matrix

Applicable to IVHM	Task	Project	Notes
VERIFICATION METHODS			
Yes	Ensure requirements are accurate and consistent[20]		From DO-178B[20]
Yes	Ensure requirements are verifiable[20]		From DO-178B[20]
Yes	Ensure requirements conform to standards[20]		From DO-178B[20]
Yes	Ensure software design meets requirements[20]		From DO-178B[20]
Yes	Ensure software architecture meets requirements[20]		From DO-178B[20]
Yes	Ensure software code meets requirements[20] Typical errors are listed below: • Failure of algorithm to satisfy software requirement • Incorrect loop operations • Incorrect logic decisions • Failure to correctly process legitimate combinations of input conditions • Incorrect handling of exceptions such as arithmetic faults or violations of array limits • Incorrect computation sequence • Inadequate algorithm precision, accuracy or performance[20]		From DO-178B[20]
Yes	Ensure use of proper naming conventions		Enhances communication between teams and aids in software maintenance
Yes	Verify expected test results[20]	DS1 RAX X-37 CLCS	From DO-178B[20]
Yes	Ensure discrepancies between actual and expected test results are explained[20]	DS1 RAX X-37	From DO-178B[20]
Yes	Ensure accuracy and consistency of source code including the following: • Stack usage • Fixed point arithmetic overflow and resolution • Use of un-initialized variables or constants • Resource contention • Worst-case execution timing • Exception handling • Unused variables or constants • Data corruption due to task or interrupt conflicts[20]		From DO-178B[20]

Applicable to IVHM	Task	Project	Notes
	VALIDATION METHODS		
Yes	Develop Operations Scenarios for all operational modes, mission phases (e.g., installation, startup, typical examples of normal and contingency operations, shutdown and maintenance) and critical sequences of activities for all classes of users. Each scenario should include events, actions, stimuli, information and interactions as appropriate to provide a comprehensive understanding of the operational aspects of the system. [21] Operations Scenarios span the following items (during nominal, off-nominal and stressful conditions) that occur during a mission: • Mission phase, mode and state transitions • First-time events • Operational Performance Limits • Fault Protection routines • Failure Detection, Isolation and Recovery (FDIR) logic • Safety Properties • Operational responses to transient or off-nominal sensor signals • Ground-to-spacecraft uplink and downlink[21] Use Operational Scenarios to test the software items, interfaces and end-to-end performance as early as possible in the software development cycle. In early stages of the development cycle, stubs may be created to implement a full scenario. In order to manage project resources effectively, it is extremely important to produce a skeleton of the actual system to run full scenarios as soon as possible with both stubbed out and actual configuration items.	DS1-RAX	
Yes	Develop baseline tests for each scenario to optimize use of expensive and scarce hardware resources like high-fidelity testbeds. Baseline tests are tests that run on a lower fidelity testbed with confidence that test results extend to higher-fidelity testbeds. *[Section 3.2.5]*	DS1-RAX	Good method for optimizing use of expensive and scarce hardware resources

Applicable to IVHM	Task	Project	Notes
Yes	Test system level interaction of all modules *[Section 3.2.6]*	DS1-RAX	
Yes	Create a complete diagram and inventory or all interfaces to ensure adequate testing of all interfaces[20]		From DO-178B[20]
No	Develop a “safety net' to allow the system to be completely disabled with single command sent either by ground or by onboard flight software. *[Section 3.2.6]*	DS1-RAX	***Note:** This test was required for the RAX, but is not applicable because IVHM is not an additional experiment but part of the main functionality.*
TYPES OF TESTS			
Yes (See notes)	Test variations of initial state and goals of the Planner-Scheduler *[Section 3.3.7]*	DS1-RAX	This is applicable based on Paragraph 6.1, *2nd Generation IVHM System Description* in IVHM Concept of Operations Document (ConOps)[24] which describes requirements for Planner-Scheduler technology
Yes (See notes)	Use Planner-Scheduler and Livingstone tests to exercise the EXEC *[Section 3.2.6]*	DS1-RAX	This is applicable based on Paragraph 6.1, *2nd Generation IVHM System Description* in ConOps[24] which describes requirements for a Smart EXEC
Yes (See notes)	Test Livingstone failure contexts *[Section 3.2.6]*	DS1-RAX	This is applicable based on Paragraph 6.1, *2nd Generation IVHM System Description* in ConOps[24] which describes requirements for Livingstone technology
Yes (See notes)	Test recovery by stopping Livingstone at various times when sharing memory etc, then restarting and checking the model *[Section 3.3.7]*	X-37	This is applicable based on Paragraph 6.1, *2nd Generation IVHM System Description* in ConOps[24] which describes requirements for Livingstone technology

Applicable to IVHM	Task	Project	Notes
Yes (See notes)	Test performance by measuring time for Livingstone engine to make a diagnoses when under stress *[Section 3.3.7]*	X-37	This is applicable based on Paragraph 6.1, *2nd Generation IVHM System Description* in ConOps [24] which describes requirements for Livingstone technology
No	Test code coverage by checking every line of Livingstone reasoning engine software *[Section 3.3.7]*	X-37	***Note:*** *If used, "as is", the Livingstone reasoning engine has been tested. Additional validation should be based on maturity of Livingstone software. Changes due to integration with other software should be tested.*
Yes	Test Livingstone or other Model code coverage by checking every line of the software model *[Section 3.3.7]*	X-37	
Yes	Test error handling by checking graceful handling of errors *[Section 3.3.7]*	X-37	
Yes	Perform regression tests to check for side effects from code or Livingstone model changes *[Section 3.3.7]*	X-37	
Yes	Test Livingstone Model against FMEA or FMECA to ensure Livingstone model accurately reflects the system being developed. ***Note:*** *FMEA – Failure Mode Effects Analysis* *FMECA: Failure Mode Effects and Criticality Analysis* *FMEA and FMECA aides in determining what loss of functionality occurs due to an unremediated fault state*	X-37	
Yes	Perform normal range tests[2] to test the ability of software to respond to normal or expected inputs and outputs		From DO-178B[20]
Yes	Perform robustness tests[2] to test the ability of software to respond to invalid input values, abnormal operating conditions, failure modes, loops, exceeded time frames, arithmetic overflow, invalid state transitions		From DO-178B[20]

Applicable to IVHM	Task	Project	Notes
Yes	Verify software works on target hardware[2]. Typical errors include the following: • Incorrect interrupt handling • Failure to satisfy execution time requirements • Incorrect software response to hardware transients or hardware failures like start-up sequencing, transient input loads and input power transients • Data bus and other resource contention problems • Inability of built-in test to detect failures • Errors in hardware/software interfaces • Incorrect behavior of feedback loops • Incorrect control of memory management hardware or other devices under software control • Stack overflow • Incorrect operation of mechanisms used to confirm the correctness and compatibility of filed-loadable software • Violations of software partitioning[20]		From DO-178B[20]
Yes	Ensure no dead or deactivated code[2]		Code that is not used should be removed
Yes	During integration testing, ensure the following: • All hardware addresses are correct • No memory overlaps • No missing software components[20]		From DO-178B[20]
Yes	During integration testing, ensure software components interact correctly with each other[2] Typical errors are listed below: • Incorrect initialization of variable and constants • Parameter passing errors • Data corruption, especially global data • Inadequate end-to-end numerical resolution • Incorrect sequencing of events and operations[20]		From DO-178B[20]

Applicable to IVHM	Task	Project	Notes
TEST CRITERIA			
Yes	Use consistent and well-documented test criteria like a Pass/Fail criteria where Passing is defined as: • System doesn't hang • System doesn't crash • Program doesn't write outside its memory allocation • Software doesn't harm vehicle • Livingstone makes correct diagnosis • Test doesn't result in an excessive number of issues or problems • Test doesn't result in failure to meet expected operability, stability and reliability expectations	X-37 CLCS	
TEST ENVIRONMENT			
Yes	Set up the following software environments: • Development Environment • Integration Environment • Operations Environment *[Section 3.4.5]*	CLCS	Software development environments may also be called: • Development • Test or Staging • Production
Yes	Distribute tests among low, medium and high-fidelity testbeds	DS1-RAX	Good method for optimizing use of expensive and scarce hardware resources

Applicable to IVHM	Task	Project	Notes
TESTING TOOLS			
Yes ***Note:*** *Specific tools included for informational purposes only and not intended as a recommendation.*	Use automated testing tools for Operations Scenarios as follows: • Accept and encode scenario description as input • Control the simulator and ground tools to execute the scenario • Stop the test when appropriate by monitoring the telemetry stream • Store all logs and downlinked files for later examination Use testing tools for the following tasks. Names of existing tools are listed when applicable. • Record the history of source files. If a bug or "side-effect" occurs when software or Livingstone models change, old source code can be reviewed to determine which change caused the bug. ***Note:*** *X-37 uses Concurrent Version System (CVS)* • Track Problem Reports. ***Note:*** *X-37 uses GNATS bug tracking software and CLCS uses RAZOR* • Test code coverage and memory tool. ***Note:*** *X-37 uses CodeTEST for Tornado* and *CLCS uses ParaSoft (www.parasoft.com) including the following components:* • *Ensure memory checker* • *Total Coverage Analysis* • *Code Wizard for static analysis of code* • *C++ Test for automated unit testing* [16]	DS1-RAX X-37 CLCS	*For more information:* *X-37: [Section 3.3.7]* *CLCS: Section 3.4.5*
Yes	Develop Ground Tools to monitor spacecraft in flight	DS1-RAX	For more information see *Section 3.2.6*
No	Public Outreach via the Web including email containing summaries of events were presented in simple English with a Java-Applet timeline *[Section 3.2.6]*		***Note:*** *Excellent public relations tool*
TESTS and REVIEWS			
Yes	**Informal Testing** performed by Software or Livingstone Model Developer to measure progress and find errors. Witnessing of these tests is not required. Tests include the following: • Unit Tests • "Black Box" or requirements-driven testing • "White Box" or design-driven testing *[Section 3.3.7]*	DS1-RAX X-37 CLCS	

Applicable to IVHM	Task	Project	Notes
Yes	**Test Readiness Review (TRR)** to obtain concurrence from development and user communities that the system is ready for official testing. This concurrence provides authority to commence official testing. *[Section 3.4.6]*	CLCS	
Yes	**Official Testing** Official Testing occurs at the end of each life cycle phase and demonstrates that software and Livingstone models are ready for intended use. Official Testing includes the following: • IVHM team approved Test Plan and Procedure • Quality Assurance (QA) witnesses • Record of discrepancies (Problem Reports) • Test Report Official Testing may also include an official code inspection involving line-by-line evaluation of the software. A team composed of members with specific roles described below perform inspections: • Moderator – facilitates meeting • Reader - leads team through the item • Author - explains the item being inspected • Reviewers - look for faults • Recorder - notes faults *[Section 3.3.7]*	X-37	
Yes	**Post-Test Review** (PTR) to summarize problems found during testing and any follow up actions required to declare successful completion of the test *[Section 3.4.8]*	CLCS	
Yes	**Delivery Acceptance Review (DAR)** The DAR is a meeting, chaired by the Project Manager or Delivery Manager. The delivery package is reviewed and declared complete. The delivery package includes the items listed below: • Copies of software to be delivered • Development documentation (users guides, requirements, design documents) • Test documentation (subsystem and system level tests) • Known problems. *[Section 3.4.7]*	CLCS	***Note:*** *DAR sometimes called "Final Sign-off" or "Customer Sign-off"*
Yes	**Operational Readiness Tests (ORT)** includes a "dress rehearsal" of operational and contingency procedures	DS1-RAX CLCS	For more information: DS1-RAX *[Section 3.2.6]*

5. SPECIAL V&V REQUIREMENTS for 2nd Generation RLV IVHM

The purpose of the Special IVHM V&V Requirements is to identify unique V&V issues and requirements specific to the 2nd Generation RLV IVHM system.

In order to properly identify special IVHM V&V requirements, the following assumptions were made:

- IVHM will include functionality as depicted in the following document: *2nd Generation RLV, TA-5 IVHM, NASA In-House Task 1, IVHM Concept of Operations Document* which describes the types of IVHM software listed below:
    - Livingstone
    - Planner-Scheduler similar to DS1-RAX
    - Smart Executive similar to DS1-RAX
- The latest version of Livingstone (L2) written in C++ will be used
- A discussion of Formal Methods is beyond the scope of this section and will be included in *Report 2 & 3*

The following table includes Special IVHM V&V Requirements. The first column includes the name of the issue for easy reference. The second column contains the issue description and the third column includes a recommendation. The last column includes a specific reference for the issue and/or recommendation, if applicable.

Issue Name	Issue	Recommendation	Reference
V&V PROCESSES - PLANNING			
Onboard Planning	Autonomous onboard planning technology challenges the comfort level of mission operators. It is difficult to move past the mindset of expecting complete predictability from an autonomous system. RAX demonstrated that the paradigm shift is indeed possible because it performed a flawless demonstration of onboard planning.	Plan to educate mission operators about autonomous onboard planning technology and to build confidence by consistently proving this technology. This will provide a basis for acceptance of rigorous V&V as appropriate for certification so autonomous software like onboard planning technology can fly on 2nd Generation RLV.	Paragraph 2.10.4 Lessons Learned, Deep Space 1 Technology Validation Report – Remote Agent Experiment

Issue Name	Issue	Recommendation	Reference
Waterfall versus Spiral Methodology	The CLCS customer was accustomed to older technologies. For example, floppy diskettes were used for storing and sharing data. In order for the CLCS customer to feel comfortable that new networking technologies were secure and user-friendly, the technical team used the Spiral methodology and developed a series of prototypes. The Spiral methodology was used throughout the project until the customer became convinced about new technologies. Then, the CLCS team moved to a Waterfall approach.	Use a Spiral methodology for presenting new technology or when customer culture is changing. Use Waterfall methodology with stable customers who have well-defined requirements Use a combination approach as the customer culture changes and new technology is proven. Methodology selection is important to the V&V effort because the V&V plan cannot be written and the V&V team cannot be effectively organized until the methodology is chosen.	Phone interview with Ric Hurt, Deputy Project Manager on September 6, 2001
IV&V Budget	CLCS planned to use an in-house V&V team and did not budget for IV&V	Evaluate IV&V requirements before budgets are finalized	Phone interview with Ric Hurt, Deputy Project Manager on September 6, 2001
Accountability	In 1999, it became apparent that the Atlas Software Delivery, part of CLCS, was in trouble and CLCS was subsequently reorganized.	Enforce accountability between partners by having clearly defined roles and responsibilities to ensure V&V is performed effectively. ***Note:*** *Specific responsibilities should be referred to or included in a Contract signed by a properly authorized representative from all appropriate parties* Include representatives from each partner at all Official Reviews described in Section 3.4.7. Ensure the Project Manager has strong project management skills/experience and has appropriate authority and responsibility for the project, including the V&V effort.	Checkout and Control Launch System presentation by Bruce Hevey dated June 14, 2001

Issue Name	Issue	Recommendation	Reference
V&V PROCESSES – ORGANIZATION			
Modeling Team	RAX team was organized so team members specialized in only one of the following engines: Planner-Scheduler or Smart Exec or MIR. Each team was responsible for modeling all spacecraft subsystems for their engine. While this organization worked for RAX, it had a few shortcomings that can be mitigated by reorganizing the teams. Shortcomings included knowledge of one spacecraft subsystem spread among three teams requiring a discussion with three team members to gain a complete understanding of how one subsystem worked.	Organize the teams so that a single team has responsibility for developing all models for a subsystem. This will ensure internal coherence of the resulting model and communication about models to the V&V team. Additionally, modelers will understand how to use all three engines and can make effective modeling decisions to exploit strengths of each engine and avoid duplication.	Paragraph 2.10.11 Lessons Learned, Deep Space 1 Technology Validation Report – Remote Agent Experiment
V&V METHODS – TESTING METHODS			
Testing Coverage of Autonomous Software	Testing coverage of autonomous software behaviors becomes more difficult with larger numbers of possible combinations of parameters and higher numbers of possible interactions between subsystems.	Plan to evaluate restricting harmful interaction "by design" to mitigate the testing coverage problem.	Paragraph 2.10. Lessons Learned, Deep Space 1 Technology Validation Report – Remote Agent Experiment
Late Changes	Spacecraft requirements and operating procedures change throughout development and after launch, but it is not always possible to encode late changes due to the time and cost of regression testing.	Reduce validation cost of model changes. Some possibilities include developing tools to evaluate the consequences of the model changes on testing. Since models already support localized changes, a procedure could be developed to uplink and install just the changes.	Paragraph 2.10.8 Lessons Learned, Deep Space 1 Technology Validation Report – Remote Agent Experiment
Ground Tools	Ground tools were not well integrated therefore only tools displaying or interpreting data in the most obvious way were of high value	Develop ground tools well in advance of the actual flight and use these tools as primary means to test and understand how to operate complex systems	Paragraph 2.10.9 Lessons Learned, Deep Space 1 Technology Validation Report – Remote Agent Experiment

Issue Name	Issue	Recommendation	Reference
Telemetry	Adequate information in telemetry is required to ensure sufficient visibility on all platforms, including in-flight.	Design telemetry early and use it as the primary way of debugging and understanding system behavior during integration, test and operations	Paragraph 2.10.10 Lessons Learned, Deep Space 1 Technology Validation Report – Remote Agent Experiment
V&V METHODS – TYPES OF TESTS			
Model Validation	One of the biggest challenges for RAX was model validation because even small changes in models had to be carefully and laboriously analyzed and tested to ensure no expected results. In some cases, it was decided to forego the change and institute flight rules to prevent the situation requiring a model change to occur.	There is a need for better model validation tools like the automated test-running capability that was developed to allow the test team to quickly evaluate a large number of off-nominal scenarios. However, even with this tool, scenario generation and evaluation of test results were still time consuming. Preliminary work at NASA in the area of formal methods for model validation is very promising.	Paragraph 2.10.3 Lessons Learned, Deep Space 1 Technology Validation Report – Remote Agent Experiment
System Level Testing	System level testing proved to be cumbersome on RAX due to the absence of a tool to generate new mission scenarios.	Development has begun on a graphical tool allowing visual inspection and modification of mission profiles, as well as constraint checking to ensure consistency.	Paragraph 2.10.5 Lessons Learned, Deep Space 1 Technology Validation Report – Remote Agent Experiment
Coding Domain Models	Coding domain models for RAX required substantial knowledge acquisition from spacecraft experts.	Develop tools and simplify the modeling languages to enable spacecraft experts to encode models themselves.	Paragraph 2.10.6 Lessons Learned, Deep Space 1 Technology Validation Report – Remote Agent Experiment
Mission Profile	Constructing a mission profile was difficult	Simplify the specification of goals when possible by using a graphical-timeline display and time-ordered listings. Automate consistency checking	Paragraph 2.10.7 Lessons Learned, Deep Space 1 Technology Validation Report – Remote Agent Experiment

6. ACRONYMS

Term	Definition
ACB	Application Control Board
AI	Artificial Intelligence
ANSI	American National Standards Institute
CCB	Change Control Board
CCP	Command & Control Processors
CCWS	Command & Control Workstations
CLCS	Checkout & Launch Control System
CM	Configuration Management
CMM	Capability Maturity Model
COTS	Commercial Off The Shelf
CVS	Concurrent Version System
DAR	Delivery Acceptance Review
DCR	Design Certification Review
DDP	Data Distribution Processors
DP-2	Design Panel 2 – Requirements Design Panel
DRP	Data Recording Processors
DS1	Deep Space One
EIA	Electronic Industries Association
EXEC	Smart Executive or EXEC
FMEA	Failure Mode Effects Analysis
FMECA	Failure Mode Effects and Criticality Analysis
GSE	Ground Support Equipment
HIT	Hardware Installation Test
IEC	International Electro-technical Commission
IEEE	Institute of Electrical and Electronic Engineers
IPS	Ion Propulsion System
ISO	International Organization for Standardization
IV&V	(NASA) Independent Verification & Validation
IVHM	Integrated Vehicle Health Management
LPS	Launch Processing System
MICAS	Miniature Integrated Camera and Spectrometer
MIL STD	Military Standard
MIR	Mode Identification Reconfiguration (also referred to as Livingstone)
NASA	National Aeronautical Space Administration
NASA ARC	NASA AMES Research Center
NASA/KCS	NASA Kennedy Space Center
NPD	NASA Policy Directive
NPG	NASA Procedures and Guidelines
O&M	Operations and Maintenance
ORR	Operational Readiness Review
ORT	Operational Readiness Tests
PCO	Project Controls Office
PR	Problem Report
PTR	Post-Test Review
RA	Remote Agent
RAX	Remote Agent Experiment
RLV	Re-usable Launch Vehicle

Term	Definition
RMA	Reliability, Maintainability, Availability
RTC	Real Time Control
RTCA	Requirements and Technical Concepts for Aviation
STP	Software Test Plan
SVP	Software Validation Procedures
SW	Software
TC	Test Conductor
TR	Test Report
TRR	Test Readiness Review
UML	Unified Modeling Language
USA	United Space Alliance
V&V	Verification & Validation
VMC	Vehicle Management Computer
VMS	Vehicle Management System

Note: *More Acronyms: http://www.ksc.nasa.gov/facts/acronyms.html*

7.GLOSSARY

Term	Definition
Advanced Software	The term "Advanced Software" is used to describe model-based and/or artificial intelligence (AI) software like model based reasoning software.
Algorithm	A rule or procedure for solving a problem[25]
Autonomous Systems	Autonomous systems rely on intelligent inference capabilities to be able to take appropriate actions even in unforeseen circumstances. They enable a whole range of new applications, such as sending autonomous robots to places where it is too dangerous or expensive for humans and/or where human control is difficult or not technically feasible[26] (for example, deep space).
Bandwidth	In analog communications, the difference between the highest and lowest frequencies in a given range. For example, a telephone line accommodates a bandwidth of 3,000 Hz, the difference between the lowest (300 Hz) and highest (3,300 Hz) frequencies it can carry. In digital communications, the rate at which information is sent expressed in bits per second (bps).[25]
Black Box testing	Requirements-driven testing where engineers select system input and observe system output/reactions
Cold Boot	To start a computer. When first turned on, the computer executes the software that loads and starts the computer's operating system, which prepares it for use.
CPU	Central Processing Unit or "brains" of a computer
CSCI	CSCI – Computer Software Configuration Item (a term used in NASA or Military standards to describe a product like a jet engine or a computer system.
Data Bus	A communication line used for data transfer among the components of a computer system. A bus is essentially a highway that allows different parts of the system to share data.
Fidelity	Integrity of testbed. For example: low fidelity testbed may have a simulator rather than actual spacecraft hardware. The highest fidelity testbed is the actual hardware being tested
FMEA/FMECA	FMEA – Failure Mode Effects Analysis FMECA - Failure Mode Effects and Criticality Analysis FMEA and FMECA aides in determining what loss of functionality occurs due to an unremediated fault state
Formal Testing	The term "formal testing" has two meanings. Traditionally, "formal testing" has been used to describe an official test occurring at the end of each life cycle phase and demonstrating that software is ready for intended use. It includes the following: o Approved Test Plan and Procedure o Quality Assurance (QA) witnesses o Record of discrepancies (Problem Reports) o Test Report With the invention of more advanced software, the term "formal testing" also refers to a type of mathematical testing using Formal Methods. Formal Methods include the following types of tests: o Model Checking o Theorem Proving o Static Analysis o Runtime Monitoring Therefore, to avoid on confusion in this document, the traditional use of "formal testing" has been replaced with the term "official testing". The term "formal testing" used in this document means formal mathematical testing (i.e. Formal Methods).

Term	Definition
Hypergolic Maintenance Facility	Building or facility to management hypergolic fuel used in the space shuttle
Integration Tests	Tests conducted when system components have been integrated or put together
Interrupt Handling	***Note:*** *Assume "standard" hardware, that is, a CPU that follows the hierarchical interrupt architectures adapted by most of today's processors such as the Intel x86 families, or the Motorola 68xxx families, or the VAX family of microprocessors. The term "toaster processor" is used to depict such a processor.* A toaster processor executes only one instruction at any given time, and during normal processing, the sequence of instructions processed is determined by the instructions: Unless the instruction implies a command that branches to somewhere else, the next instruction to be processed is the instruction that immediately succeeds the current instruction. Concurrency in a toaster processor is implemented via interrupts. Given certain circumstances, the toaster processor can decide to suspend the current stream of execution that we discussed above, and transfer control to a different routine called an "interrupt handler" (hereafter frequently referred to as "ISR" or "interrupt service routine"). When the interrupt handler has finished its processing, control is returned to the instruction stream that was previously interrupted. To the original stream, the execution of the interrupt handler is perfectly transparent; the original stream is not aware of the fact that it was suspended and re-awakened. This idea of concurrent execution is similar to application-level concurrency only in that streams of execution will not be aware of any concurrency.
Java-Applet	Small Java program that runs in a web browser.
LISP	AI development software
Nominal	Expected behavior for no failure, for example: nominal behavior for a valve may be "open" or "shut"
Off-nominal	Unexpected failure behavior, for example: off-nominal behavior for a valve may be "stuck open" or "stuck shut"
Paradigm Shift	
Program	The term "Program" is used as a generic term to describe a mission or project conducted at NASA. For example, this document contains a survey of the Deep Space One Program, rather than the Deep Space One Mission.
Regression tests	Extrapolates the impact of the changes on program, application throughput and response time from the before and after results of running tests using current programs and data.[27]
Software V&V	Process of ensuring that software being developed or changed will satisfy functional and other requirements (verification) and each step in the process of building the software yields the right products (validation).[10] In other words: • Verification – Build the Product Right • Validation – Build the Right Product
Stack	A stack is a list where items are added to one end of the list and removed from the same end. Whatever goes on to the list last comes out first. Because the items are removed in last-in-first-out order, stacks are also sometimes called LIFO lists, or LIFOs. For historical reasons, adding an item is called *pushing* the item onto the stack; removing an item is called *popping* it off the stack.[25]
Stack Overflow	The stack overflows because a program does not allocate enough stack space to hold the data the application uses during execution.[25]
Stanley	GUI (graphic user interface) used to create Livingstone models

Term	Definition
Telemetry	Process of measuring data at the source and transmitting it automatically. Telemetric applications include measuring and transmitting data from space flights, meteorological events, wildlife tracking, camera control robotics and oceanography studies.[28]
Unit Tests	Tests conducted on individual software components
Unix	Computer operating system
Validation	Build the Right Product
Verification	Build the Product Right
White Box testing	Design-driven testing where engineers examine internal workings of code
X-37 Plane	The X-37 plane is a reusable launch vehicle designed to operate in both the orbital and re-entry phases of flight

8. FOR MORE INFORMATION

V&V Standards:
IEEE Standard for Software Verification and Validation 1012-1998

Websites:

Capability Maturity Model	http://Sei.com.edu
Charles Pecheur Website:	http://ase.arc.nasa.gov/pecheur
CLCS	http://clcs.ksc.nasa.gov/
Livingstone Website:	http://ic-www.arc.nasa.gov/ic/projects/mba/projects/livingstone.html
MIL STD 498	http://www.pogner.demon.co.uk/mil_498 http://www.comsivam.org/reference/498/
Remote Agent	http://rax.arc.nasa.gov/publications.html

9. APPENDIX A: Missions Incorporating Livingstone

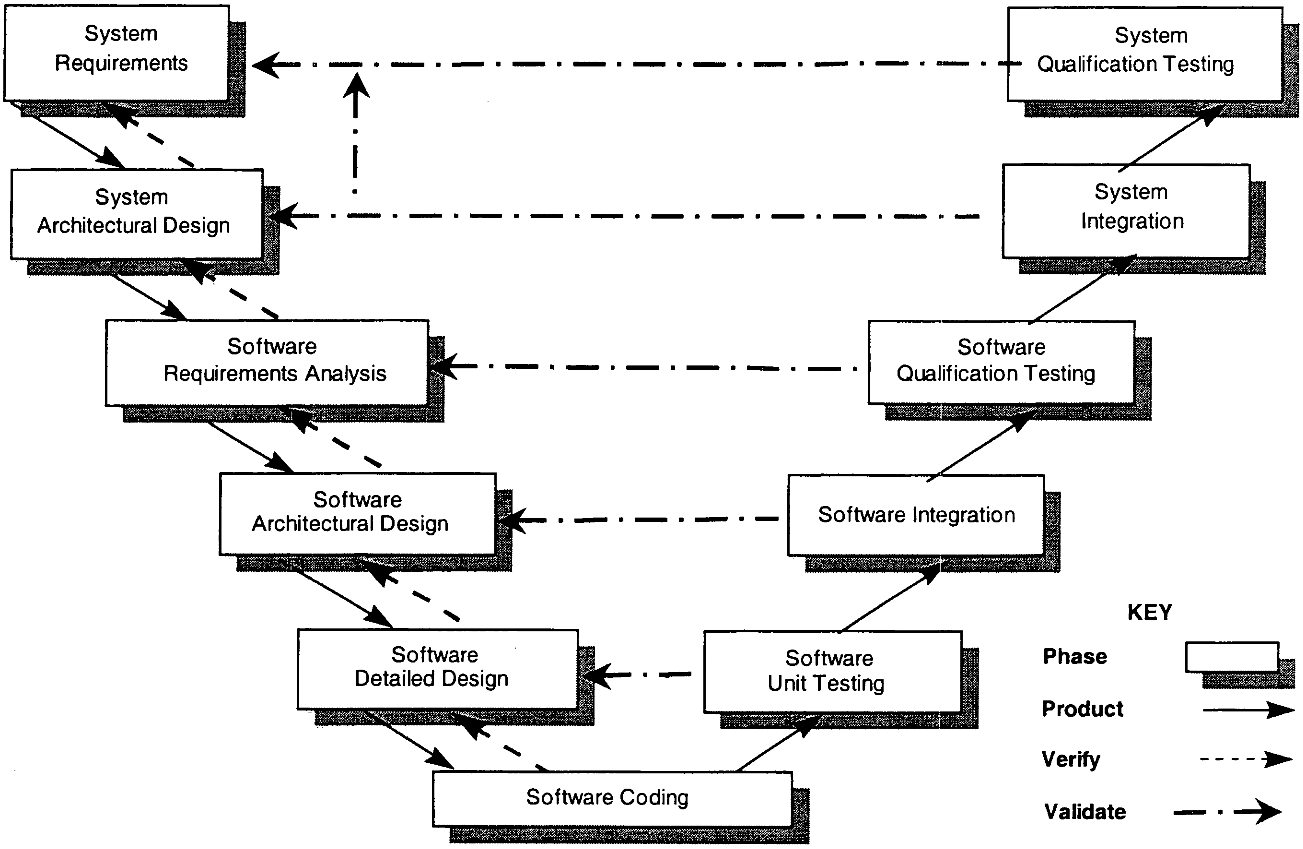
More information about the missions incorporating Livingstone is located at the Livingstone website: http://ic-www.arc.nasa.gov/ic/projects/mba/projects/livingstone.html.

Mission	Website Address
Autonomous Space Flight	
Cassini-Huygens Mission to Saturn and Titan	http://www.jpl.nasa.gov/cassini/english/
Environmental	
Bioreactor	http://ic-www.arc.nasa.gov/ic/projects/Bioreactor/Bioreactor.html
Controlled Ecological Life Support System (CELSS)	http://www.hydrogarden.com/class1/nasa.htm
Mars Life Support Testbed	http://mars.jpl.nasa.gov/
Mars Propellant Plant	
In-Site Propellant Production Plant (ISPP)	http://spaceflight.nasa.gov/mars/technology/mipp/
IN-SITU PROPELLANT PRODUCTION ON MARS: The First Flight Demonstration	http://powerweb.grc.nasa.gov/pvsee/publications/mars/MIP_LPSC.html (Paper presented at the *30th Lunar and Planetary Science Conference*, Houston TX, March 15-19 1999)
Rover Autonomy	
Autonomous Rover Command Generation	http://www-aig.jpl.nasa.gov/public/planning/rover_command/
Advanced Autonomy for Rovers	http://ic-www.arc.nasa.gov/ic/projects/ai-rovers/index.html
Telescopes	
Automatic Telescopes	http://ic-www.arc.nasa.gov/ic/projects/xfr/index.html
Space-based Interferometry Testbed	http://us-space-vlbi.jpl.nasa.gov/tutorial/svlbi.html

10. APPENDIX B: Other Projects

Mission	**Website Address**
X-Planes	
X-38 Lifeboat	http://www1.msfc.nasa.gov/NEWMSFC/xplanes.html
X-33	http://x33.msfc.nasa.gov
X-34	http://www.orbital.com/LaunchVehicles/x34/x-34.htm
Other	
HEDS Technology Experiment	http://www.lpi.usra.edu/lpi/HEDS-UP
GPSS (General Planning & Shuttle Scheduling)[29]	Software to schedule all jobs to refurbish the space shuttle
SCAN (Shuttle Connector Attachment Network)[29]	Preceded GPSS and included a database, manually updated by engineers, with the "state of connectivity" including information about connections like cables and connectors
MDT (MCDS Diagnostic Tool) [29] MCDS (Multi-function CRT Display System) CRT (Cathode Ray Tube)	Rule-based systems written in CLIPS and containing about 50 rules to perform diagnostics. Monitored via telemetry. Because KSC has no policies to certify intelligent systems, this system cannot run in the primary operational firing room (control center for space shuttle). As a validation effort, it runs in the backup firing room using a copy of "live" data from telemetry. This is considered a high fidelity test bed. During launch day, technical experts viewed MTD data for testing purposes.
PAT (Propulsion Advisory Tool) [29]	Advisory tool in development for 8-10 years. GUI presenting information from a neural net application called ANNT. Monitors and detects anomalies in wave patterns.
DLES (DPS LCC Expert System) [29] DPS (Data Processing System) LCC (Launch Commit Criteria)	Similar to MDT but monitors the entire system
KATE (Kennedy Autonomous Test Engineer) [29]	Detects faults and diagnoses liquid oxygen loading. Validation included researchers monitoring in the backup firing room. During validation, this system flagged a fault about 30 minutes before a console engineer.
ALSACT[29]	Autonomous, rule-based system for growing crops on a long duration flight

11. APPENDIX C: Relationship between V&V and Traditional Life Cycle Phases[18]



12. APPENDIX D: Independent Verification and Validation (IV&V) Criteria

Note: *This appendix was copied from 8730.DRAFT2 NASA Procedures and Guidelines Software Independent Verification and Validation (IV&V) Management dated 2/26/01. The official NASA policy, NPD 8730.4 Software Independent Verification and Validation (IV&V) Policy, was approved August 1, 2001. It will be published in NODIS (NASA Online Directives Information System) library in the future.*

1. Introduction

1.1 The purpose of this appendix is to provide quantifiable criteria for determining whether IV&V should be applied to a given software development. Since IV&V should begin in the Formulation Subprocess (as defined in NPG 7120.5, paragraph 1.4.3) of a project, the process described here is based on metrics which are available before project approval.

1.2 All projects containing software shall evaluate themselves against these criteria to determine if a Software IA or an IV&V is required.

1.3 These criteria shall be applied to NPG 7120.5 "projects" as defined in the NPG. Software developments outside the scope of NPG 7120.5 are determined to be within scope of this appendix on a case by case basis. That decision will be made by the NASA Chief Information Officer (CIO), the NASA Office of the Chief Engineer (OCE), and the NASA Office of Safety and Mission Assurance, or Center Safety and Mission Assurance Office.

1.4 Projects meeting the following criteria are not subject to this appendix, and need not be addressed further:

a. The software product is only used for post-mission scientific data analysis.

b. Consequences of software failure do not exceed any of the following:

(1) Potential for loss of life – No.

(2) Potential for serious injury – No.

(3) Potential for catastrophic mission failure – No.

(4) Potential for partial mission failure – No.

(5) Potential for loss of equipment – Less than $2,000,000.

(6) Potential for waste of resource investment – Less than 20 work-years on software.

(7) Potential for adverse visibility – No more than local visibility.

(8) Potential effect on routine operations – No more than a Center inconvenience.

2. Risk Factors and Consequences

2.1 IV&V is intended to assist mitigating risk; hence, the decision to do IV&V should be risk based. NPG 7120.5 defines risk as the "combination of 1) the probability (qualitative or quantitative) that a program or project will experience an undesired event such as cost overrun, schedule slippage, safety mishap, or failure to achieve a needed breakthrough; and 2) the consequences, impact, or severity of the undesired event were it to occur." The exact probability of occurrence and consequences of a given software failure cannot be calculated early in the software lifecycle. However, there are realistically available metrics which give good general approximations of the consequences as well as the likelihood of failures.

2.2 In general, the consequences of a software failure can be derived from the purpose of the software: i.e., what does the software control; what do we depend on it to do. Paragraph 2.2.1 contains a list of factors, which can be used to categorize software based on its intended function as well as the level of effort expended to produce the software. Paragraph 2.2.2 defines the boundaries of four levels of failure consequences based on the rating factors from paragraph 2.2.1.

2.2.1 Factors contributing to the consequences of software failure include the following:

a. Potential for loss of life. Is the software the primary means of controlling or monitoring systems that have the potential to cause the death of an operator, crewmember, support personnel, or bystander? The presence of manual overrides and failsafe devices are not to be considered. This is considered a binary rating: responses must be either yes or no. Examples of software with the potential for loss of life include:

(1) Flight and launch control software for manned missions.

(2) Software controlling life support functions.

(3) Software controlling hazardous materials with the potential for exposure to humans in a lethal dose.

(4) Software controlling mechanical equipment (including vehicles) which could cause death through impact, crushing, or cutting.

(5) Any software which provides information to operators where an inaccuracy could result in death through an incorrect decision (e.g., mission control room displays).

b. Potential for serious injury. Serious injury is here defined as loss of digit, limb, or sight in one or both eyes, loss of hearing, or exposure to substance or radiation that could result in long term illness. This rating is also binary. This rating considers only those cases where the software is the primary mechanism for controlling or monitoring the system. The presence of manual overrides and failsafe devices are not to be considered. Examples of software with potential for serious injury include software controlling milling or cutting equipment, class IV lasers, or X-ray equipment.

c. Potential for catastrophic mission failure. Can a problem in the software result in a catastrophic failure of the mission? This is a binary rating. Software controlling navigation, communications, or other critical systems whose failure would result in loss of vehicle or total inability to meet mission objectives would fall into this category.

d. Potential for partial mission failure. Can a problem in the software result in a failure to meet some of the overall mission objectives? This is a binary rating. Examples of this category include software controlling one of several data collection systems or software supporting a given experiment, which is not the primary purpose of the mission.

e. Potential for loss of equipment. This is a measure of the cost (in dollars) of physical resources that are placed at risk due to a software failure. Potential collateral damage is to be included. This is exclusive of mission failure. Examples include the following:

(1) Loss of a $5 million unmanned drone due to flight control software failure. (Assuming the drone is replaceable, this wouldn't be a mission failure.)

(2) Damage to a wind tunnel drive shaft due to a sudden change in rotation speed.

f. Potential for waste of software resource investment. This is a measure or projection of the effort (in work-years, civil service, contractor, etc.) invested in the software. This shows the level of effort that could potentially be wasted if the software does not meet requirements.

g. Potential for adverse visibility. This is a measure of the potential for negative political and public image impacts stemming from a failure of the system as a result of software failure. The unit of measure is the geographical or political level at which the failure will be common knowledge—specifically: local (Center), Agency, national, international. The potential for adverse visibility is evaluated based on the history of similar efforts.

h. Potential effect on routine operations. This is a measure of the potential to interrupt business. There are two major components of this rating factor: scope and impact. Scope refers to who is affected. The choices are Center and Agency. The choices for impact are inconvenience and work stoppage. Examples include the following:

(1) A faulty firewall which failed to protect against a virus resulting in a 4-hour loss of e-mail capabilities at a Center would be a "Center inconvenience."

(2) Assuming that the old financial management software was no longer maintainable, the failure of the replacement system to pass acceptance testing and the resulting 2-year delay would be a potential "Agency work stoppage." This does not imply that workarounds could not be implemented, but only that it has the potential to stop work Agencywide.

2.2.2 Software Consequences of Failure Rating.

2.2.2.1 Consequences of failure are considered "Grave" when *any* of the following conditions are met:

a. Potential for loss of life – Yes.

b. Potential for loss of equipment – Greater than $100,000,000.

c. Potential for waste of resource investment – Greater than 200 work-years on software.

d. Potential for adverse visibility – International.

2.2.2.2 Consequences of failure are considered "Substantial" when *any* of the following conditions are met:

a. Potential for serious injury – Yes.

b. Potential for catastrophic mission failure – Yes.

c. Potential for loss of equipment – Greater than $20,000,000.

d. Potential for waste of resource investment – Greater than 100 work-years on software.

e. Potential for adverse visibility – National.

f. Potential effect on routine operations – Agency work stoppage.

2.2.2.3 Consequences of failure are considered "Marginal" when *any* of the following conditions are met:

a. Potential for partial mission failure – Yes.

b. Potential for loss of equipment – Greater than $2,000,000.

c. Potential for waste of resource investment – Greater than 20 work-years on software.

d. Potential for adverse visibility – Agency.

e. Potential effect on routine operations – Center work stoppage or Agency inconvenience.

2.2.2.4 Consequences of failure are considered “Insignificant” when *all* of the following conditions are met:

a. Potential for loss of life – No.

b. Potential for serious injury – No.

c. Potential for catastrophic mission failure – No.

d. Potential for partial mission failure – No.

e. Potential for loss of equipment – Less than $2,000,000.

f. Potential for waste of resource investment – Less than 20 work-years on software.

g. Potential for adverse visibility – No more than local visibility.

h. Potential effect on routine operations – No more than a Center inconvenience.

2.3 The probability of failure for software is difficult to determine even late in the development cycle. However, Table 1 contains simple metrics on the software, the developer, and the development environment which have proven to be indicators of future software problems. While these indicators are not precise, they provide order of magnitude estimates, which are adequate for assessing the need for IV&V. (The NASA IV&V Facility and the NASA Software Working Group will further refine these indicators and their associated weighting factors as more data become available.)

Factors contributing to probability of software failure	**Un-weighted probability of failure score**					**Weighting Factor**	**Likely-hood of failure rating**
	1	2	4	8	16		
Software team complexity	Up to 5 people at one location	Up to 10 people at one location	Up to 20 people at one location or 10 people with external support	Up to 50 people at one location or 20 people with external support	More than 50 people at one location or 20 people with external support	X2	
Contractor Support	None	Contractor with minor tasks		Contractor with major tasks	Contractor with major tasks critical to project success	X2	
Organization Complexity*	One location	Two locations but same reporting chain	Multiple locations but same reporting chain	Multiple providers with prime sub relationship	Multiple providers with associate relationship	X1	
Schedule Pressure**	No deadline		Deadline is negotiable		Non-negotiable deadline	X2	
Process Maturity of Software Provider	Independent assessment of Capability Maturity Model (CMM) Level 4, 5	Independent assessment of CMM Level 3	Independent assessment of CMM Level 2	CMM Level 1 with record of repeated mission success	CMM Level 1 or equivalent	X2	
Degree of Innovation	Proven and accepted		Proven but new to the development organization		Cutting edge	X1	
Level of Integration	Simple - Stand alone				Extensive Integration Required	X2	
Requirement Maturity	Well defined objectives - No unknowns	Well defined objectives - Few unknowns		Preliminary objectives	Changing, ambiguous, or untestable objectives	X2	
Software Lines of Code***	Less than 50K		Over 500K		Over 1000K	X2	
Total							

Table 1 Likelihood of Failures Based on Software Environment

3. Risk Assessment

Combining the software consequences of failure and the likelihood of failure rating from Paragraph 2 yields a risk assessment, which can be used to identify the need for IV&V. The indication of whether IV&V is required is obtained by plotting in Figure 1 the intersection of the Consequences of Software Failure determination and the Total Likelihood of Failure determination. Application of these criteria simply determines that a project is a candidate for IV&V – not the level of IV&V or the resources associated with the IV&V effort. These will be determined as a result of discussions between the project and the NASA IV&V Facility.

a. Figure 1 shows a dark region of high risk where software consequences, likelihood of failure, or both are high. Projects having software that falls into this high-risk area shall undergo IV&V. The exception is those projects which have already done hardware/software integration. An IV&V would not be productive that late in the development cycle. These projects shall undergo a Software Independent Assessment (IA). (See paragraph 3.b.) A software independent assessment (IA) is a review of and analysis of the project/program's system software development lifecycle and products. The IA differs in scope from a full IV&V program in that IV&V is applied over the lifecycle of the system whereas an IA is usually a one time review of the existing products and plans.

b. Figure 1 shows three gray regions of intermediate risk. Projects having software that falls into these areas shall undergo a Software IA. The NASA IV&V Facility shall conduct the Software IA according to established procedures. One purpose of the Software IA is to ensure that the software development does not have project-specific risk characteristics that would warrant the performance of IV&V. Should such characteristics be identified, a recommendation for IV&V performance will be made.

c. The following notes and definitions apply to Figure 1:

(1) Organization complexity is an indirect measure of communications challenges inherent in the software developer. A single organization working from multiple locations faces a slightly greater challenge than an organization in one location. When the software development is accomplished by multiple organizations working for a single integrator, the development is significantly complicated. If the developing organizations are coequal such as in an associate contractor relationship (or a similar relationship between government entities) then there is no integrator. Experience has shown this arrangement to be extremely challenging as no one is in charge.

(2) Under "schedule pressure" a deadline is negotiable if changing the deadline is possible although it may result in slightly increased cost, schedule delays, or negative publicity. A deadline is nonnegotiable if it is driven by immovable event such as an upcoming launch window.

(3) As the problems identified in IV&V are often mismatches between the intended use and the actual software built, "software lines of code" shall include reused software and autogenerated software.

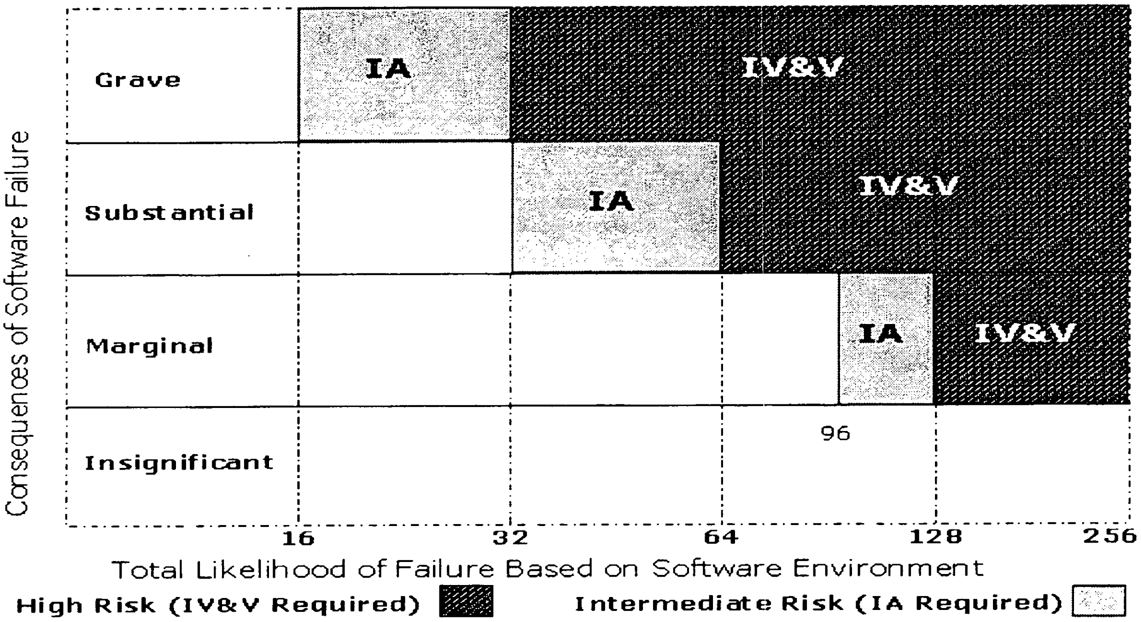


Figure 1 Software Risk

13. APPENDIX E: Required NASA Software Metrics

Note: *This appendix was copied from a draft on http://npg2820.nasa.gov/metrics.html. For the latest guidelines, see http://npg2820.nasa.gov/metrics.html.*

1. Software Characteristics are captured in the initial quarter. (Note: If the data requested is not available at initial submission, complete it at the next quarterly reporting period.)

2. Software Planning, Tracking, and Oversight Data are collected on a quarterly basis.

3. Delivery Data are collected only once. They are collected when the system or component is released to its customer or intended user.

4. Operational Reliability data are collected quarterly after delivery.

5. Comments about this project are collected at any time.

6. Closeout Data items are captured only once. They are captured when the CSCI has left operation or service.

Metrics Data to be Collected	**Program/ Project Manager Goals and Questions**
1.0 Software Characteristics	
1.1 Project CSCI Identification	G1.1: Classify the project in order to uniquely reference it within the database and generate baseline and comparative project management information.
M1.1.1—Project name	
M1.1.1.1—CSCI name[1]	
M1.1.2—Contact person[2]	
M1.1.2.1—Contact person's E-mail	
M1.1.2.2—Contact person's Center	
M1.1.3—Start date for this CSCI	
M1.1.3.1—Number of planned Spiral/Build Iterations	
M1.1.4—Estimated final delivery date for this CSCI	
M1.1.4.1—Estimated total source code count[3]	
M1.1.4.2—Predominant languages used	
M1.1.4.3—Estimated percent of functionality provided by COTS or M1.1.4.4—Estimated percent of software costs allocated to COTS	
M1.1.5—Type of CSCI (see P.2.5: 1=flight CSCI, 2=ground CSCI, and 3=any other software development item)	Q1.1: Are there other projects in the database that can be used as a basis or guideline to estimate this project?
1.2 CMM Level	G1.2: Identify developer level of processes maturity, performance, and

[1] When entering multiple releases/deliveries, treat each delivery as a separate CSCI under the same project name.

[2] Name of person delegated the responsibility for submitting the NASA Software Metrics on a quarterly basis for this CSCI.

[3] The count will include software that is to be delivered. The count includes executable statements, and does not include comments, blank lines, or commercial off-the-shelf (COTS) software.

	associated level of oversight needed (i.e., additional resources needed for low-CMM-level developers).
M1.2.1—Has your organization had a Software Capability Evaluation (SCE) by an SEI certified Lead Assessor? [4] M1.2.1.1—If yes, what was the rating?	Q1.2.1: At what CMM Level is the developer?
M1.2.2—Estimated CSCI total software cost[5]	Q1.2.2: What is the total software cost?
1.3 First Quarter Planning Data	
M1.3.1—Software accumulated cost for the first quarter ($K)	
M1.3.2—Total software staff hours for the first quarter (civil servant and contractor combined)	
M1.3.2—Number of software products to be completed in the first quarter	
1.4 Authorized person for generating reports	
M1.4.1—Software manager's name	
M1.4.1.1—Software manager's e-mail	
M1.4.1.2—Software manager's Center	
2.0 Software Planning, Tracking, and Oversight Data	
2.1 Software CSCI Management Data	G2.1: Determine project progress against plans in order to: —Understand the accuracy of the original estimates. —Determine whether adequate resources are being applied. —Make course corrections (re-planning decisions) during project development to complete on time and within budget. —Know where to apply reserves. —Determine when to look for factors that are impacting plans. —Look for trends to improve future estimating on this project.
M2.1.1—Reporting Quarter[6]	
M2.1.2—Fiscal year	
M2.1.3—Release	
M2.1.3.1—Spiral/Build Iteration Number[7]	
M2.1.4—Was a new software schedule baseline established during this reporting period?	
M2.1.5—Planned Software Accumulated Cost through the subsequent quarter (see footnote for M1.2.2) M2.1.5.1—Actual Software Accumulated Cost through the recently completed quarter (see footnote for M1.2.2)	Q2.1.5: What is the difference between planned and actual cost?
M2.1.6.1—Planned Completion date for Software Project Management Plan[8]	Q2.1.6: What is the difference between the planned and actual schedule?

[4] If any portion of the software development is contracted out, also complete these items for each contract.

[5] Including the cost of development, COTS and government off-the-shelf (GOTS) software, middle-ware, computers, and contractors and civil servants costs; but not including, however, the cost of maintenance.

[6] Specify the fiscal quarter for which this data is being submitted.

[7] If only one is anticipated, specify 1.If multiple spiral/builds are scheduled, specify the build that this fiscal quarter's data is being submitted under.

[8] After the initial data is entered, only record the changes from quarter to quarter.

M2.1.6.1.1—Actual Completion date for Software Project Management Plan M2.1.6.2—Planned Completion date for Software Requirements Analysis[9] M2.1.6.2.1—Actual Completion date for Software Requirements Analysis M2.1.6.3—Planned Completion date for Software Architectural Design M2.1.6.3.1—Actual Completion date for Software Architectural Design M2.1.6.4—Planned Completion date for Software Detailed Design M2.1.6.4.1—Actual Completion date for Software Detailed Design M2.1.6.5—Planned Completion date for Software Coding & Testing M2.1.6.5.1—Actual Completion date for Software Coding & Testing M2.1.6.6—Planned Completion date for Software Integration M2.1.6.6.1—Actual Completion date for Software Integration M2.1.6.7—Planned Completion date for Software Qualification Testing M2.1.6.7.1—Actual Completion date for Software Qualification Testing M2.1.6.8—Other Planned Completion date[10] M2.1.6.8.1—Other Actual Completion date	
M2.1.7—Total software staff hours planned for the subsequent quarter (civil servant and contractor combined) M2.1.7.1—Total software staff hours actual for the recently completed quarter (civil servant and contractor combined)	Q2.1.7: What is the difference between planned and actual total staff hours?
M2.1.8—Number of planned software products to be completed during the subsequent quarter[11] M2.1.8.1—Number of software products completed during the recently completed quarter	Q2.1.8: What is the actual software production progress against plans?
2.2 Software Requirements Management Data	G2.2: Determine software requirements stability in order to: —Determine when resources need to be allocated to solidify requirements. —Determine the growth in requirements. —Determine when to look for factors that are influencing requirements change and/or growth. —Identify when requirements changes are causing cost, schedule, and workforce impacts.
Was a new software requirements baseline established during this reporting period?	

[9] Note: the terms used in the following items (i.e. Software Requirements Analysis, Software Architectural Design, Software Detailed Design etc.) are taken from IEEE 12207.0.

[10] Examples may include support of System Integration or System Qualification Testing.

[11] Examples of software products are Software Requirements Description, Software Architecture Description, Software Design Description, Source Code Record (e.g. CSC 1, CSC 2, CSC N), Test Plan, Test Procedures, Test Results Report, User Documentation Description, etc.

M2.2.1—Software requirements count[12] at current baseline M2.2.2—Date the current requirements were baselined D2.2.3: Software requirements count at previously entered baseline D2.2.4: Change in total requirements count from previously entered baseline to the current baseline (M2.2.1 – D2.2.3)	Q2.2.1: What is the baseline software requirements count? Q2.2.4: How much has the total software requirements count changed since baseline?
M2.2.5—Total number of changes to software requirements from the previously entered baseline to the current baseline (Total number of additions, deletions, and modifications.)	Q2.2.5: How many software requirements changes have been accepted from the previously entered baseline to the current baseline?
D2.2.6: Percent of change in software requirements (M2.2.5/D2.2.3 × 100)	Q2.2.6: What is the percent of change in software requirements from the previously entered baseline to the current baseline?
M2.2.7—Current requirements count (i.e., end of the quarter)	Q2.2.7: How much has the total software requirements count changed since baseline?
2.3 Software Testing Data	G2.3: Monitor the number of open and closed Problem Reports (PRs)[13] in order to determine reliability, impacts to schedule, cost, and workforce; and evaluate the likelihood of delivery on time based on the rate at which PRs are being opened and closed.
M2.3.1—Total number open PRs M2.3.2—Total number of closed[14] PRs D2.3.3: Total PRs (M2.3.1 + M2.3.4)	Q2.3.1: What is the status of the PRs?
3.0 Operational Reliability	
3.1 Operational Reliability	G3.1: Determine software operational reliability.
M3.1.1—Total number of confirmed PRs associated with software functionality [15]	Q3.1.1: What is the total number of confirmed software problems that have been reported since delivery?
4.0 Delivery Data	
4.1 Project Completion Data	G4.1: Determine this project's actuals for use in planning future projects.
M4.1.1—Actual final delivery date for this CSCI	Q4.1.1: What is the actual delivery date?
M4.1.2—Actual total source code count (see footnote 6)	Q4.1.2: What is the total source code count?
D4.1.3: Difference between actual and planned lines of code (M4.1.2-M1.1.4)	Q4.1.3: What is the difference between actual and planned lines of code at delivery?
M4.1.4—Estimated percent of functionality provided by COTS Or M4.1.5—Estimated percent of software costs allocated to	Q4.1.4 What percent of the functions required are being accomplished by COTS?

[12] The software requirements count is the number of unique *shall* statements or other imperatives in the Software Requirements Description for this CSCI. The Automated Requirements Management (ARM) tool is freely available for performing requirements counts at URL http://satc.gsfc.nasa.gov/tools/arm/index.html.

[13] Problem Reports (PRs) as referenced in this text refer only to the Qualification Testing [IEEE 12207.0] phase for this delivery/release. Other terms such as *Discrepancy Reports*, or *Failure Reports* are frequently used.

[14] A PR is closed once corrective action is successfully implemented and verified.

[15] These PRs are reported after delivery. This data is reported as long as the software is being changed during operations. See IEEE 12207.1, clause 6.10 for details on problem reports.

COTS	
M4.1.6—Date operational	Q4.1.6: What is the date the system went into operation?
5.0 Comments about this project	
M5.1—Comments section	G5.1: Provide an explanation concerning any particular metric that was not entered or any item that you feel needs explanation.
6.0 Closeout Data[16]	
M6.1—If CSCI is closed, please check here.	

Definitions:
Computer Software Configuration Item (CSCI)—an aggregation of software that is designated for configuration management and treated as a single entity in the configuration management process.[IEEE STD 610.12-1990]
Computer Software Component (CSC)—a functionally or logically distinct part of a CSCI; typically an aggregate of two or more software units.[IEEE STD 610.12-1990]

[16] Closeout data is collected only once. It is collected when the CSCI leaves operation or service.

14. APPENDIX F: Introduction to IEEE 12207.0 Life Cycle Processes

Note: *The following information was taken from the current IEEE 12207.0 and IEEE 12207.1 standards. For the most current information see http://mainlib.arc.nasa.gov/browseft.htm or http://www.ieee.org*

IEEE 12207.0 provides a road map for users of the ISO 12207 standard so they can apply it judiciously. It describes the software life cycle processes that can be employed to acquire, supply, develop, operate and maintain software products.

The software life cycle includes five primary processes, eight supporting processes and four organizational processes. Each life cycle process is divided into a set of activities; each activity is further divided into a set of tasks.

Five Primary Life Cycle Processes

- Acquisition – defines activities of the acquirer (organization that acquires a system, software product or service)
- Supply – defines activities of the supplier (organization that provides the system, software product or service to the acquirer)
- Development – defines activities of the developer (organization that defines and develops the software product)
- Operation – defines activities of the operator (organization that provides the service of operating a computer system in its live environment for its users)
- Maintenance – defines activities of the maintainer (organization that provides the service of maintaining the software product, for example: managing modifications to the software to keep it current. Also, includes migration and retirement of the software product)

Supporting Life Cycle Processes

The Supporting Life Cycle consists of eight processes that support another process as an integral part with a distinct purpose:

- Documentation – defines activities for recording project information
- Configuration Management – defines activities for controlling modifications and release of items like source code and documentation. Other activities include reporting status of items, managing modification requests, ensuring completeness and consistency of items and controlling storage, handling and delivery of items.
- Quality Assurance – defines activities for objectively assuring that software meets or exceeds requirements
- Verification – defines activities for ensuring compliance with requirements
- Validation – defines activities for testing software products
- Joint Review Process – defines activities for evaluating the status of products
- Audit Process – defines activities for determining compliance with the requirements, plans and contract.
- Problem Resolution Process – defines a process for analyzing and removing problems (bugs, issues, anomalies, etc)

Organizational Life Cycle Processes

Organizational Life Cycle Processes are employed by an organization to establish and implement an underlying structure of a project. The include the following:

- Management process – defines basic activities of project management
- Infrastructure Process – defines basic activities for establishing the underlying structure of the life cycle process
- Improvement Process – defines activities for measuring, controlling and improving the life cycle process

- Training Process – defines activities for providing adequately trained team members

Because IEEE 12207.0 encompasses many types of projects, it must be tailored to the specific needs of each project. Tailoring processes are included in the standard and summarized below:

1. Identify project environment, for example: the 2nd Generation RLV IVHM is a large, complex project with multiple partners from different companies
2. Solicit inputs
3. Select processes, activities and tasks
    - Processes, activities and tasks not included in the IEEE standards should be specifically included in the contract for the project
    - Standards that contain "shall" or "will" should be carefully considered and a rationale documented if not included
    - Factors to be considered include risk, cost, schedule, performance, size, criticality and human interface
4. Document tailoring decisions and rationale

Tailoring the Development Process

Tailoring of the Development Process requires special attention because different parties with different objectives may use it. As a first-level tailoring of this process, the following is recommended:

- For software product that is embedded in or integral to the system: All the activities in the process should be considered and it should be clarified whether the developer is required to perform or support the system activities
- For stand-alone software product, the system activities described in the paragraphs listed below may not be required but should be considered:
    - 5.3.2 – System Requirements Analysis
    - 5.3.3 – System Architectural Design
    - 5.3.10 –System Integration
    - 5.3.11 – System Qualification Testing

15. APPENDIX G: Overview of IEEE 12207.0 and 12207.1 V&V Standards

Note: *The following information was taken from the current IEEE 12207.0 and IEEE 12207.1 standards. For the most current information see http://mainlib.arc.nasa.gov/browseft.htm or http://www.ieee.org*

IEEE 12207.0, Paragraph 6.4 describes Verification standards. IEEE 12207.0, Paragraph 6.5 describes Validation standards. IEEE 12207.1 describes specific reporting standards.

Verification - IEEE 12207.0, Paragraph 6.4
First, a decision must be made whether the project warrants a verification effort and the degree of independence of that effort. To make that decision, the following must be considered:

- Criticality gauged in terms of items listed below:
  - Potential of an undetected error in a system to cause death or injury, mission failure or financial catastrophe
  - Maturity of and risks associated with software technology
  - Availability of funds

If the project warrants verification, then a verification process must be established. If the project warrants an independent verification then IV&V must be included in the verification process.

Verification Process
The verification process requires developing and following a verification plan consisting of the following:

- Contract verification
  - Supplier has capability to satisfy requirements
  - Requirements are consistent and cover user needs
  - Adequate procedures for handling changes to requirements and escalating problems are stipulated
  - Procedures and their extent for interface and cooperation among the parties are stipulated
  - Acceptance criteria and procedures are stipulated in accordance with requirements
- Process verification
  - Project planning requirements are adequate and timely
  - Processes selected for the project are compliant with contract, adequate, implemented and being executed
  - Standards, procedures and environments for the project are adequate
  - Project is staffed with trained personnel
- Requirements verification
  - System requirements are feasible, consistent and testable
  - Requirements have been appropriately allocated to hardware items, software items and manual operations according to design criteria
  - Software requirements related to safety, security and criticality are correct as shown by suitably rigorous methods
- Design verification
  - Design is correct
  - Design is consistent with and traceable to requirements
  - Design implements proper sequence of events, inputs, outputs, interfaces, logic flow, allocation of timing and sizing budgets and error definition, isolation and recovery
  - Selected design can be derived from requirements
  - Design implements safety, security and other critical requirements as shown by suitable rigorous methods
- Code verification
  - Code is traceable to design and requirements, testable, correct, and compliant with requirements and coding standards

        - Code implements proper event sequence, consistent interfaces, correct data and control flow, completeness, appropriate allocation timing and sizing budgets and error definition isolation and recovery
        - Code can be derived from design or requirements
        - Code implements safety, security and other critical requirements correctly as shown by suitably rigorous methods
    - Integration Verification
        - Software components and units of each software item have been completely and correctly integrated Hardware items, software items and manual operations of the system have been completely and correctly integrated
        - Integration tasks have been performed in accordance with an integration plan
    - Documentation verification
        - Documentation is adequate, complete and consistent
        - Documentation preparation is timely
        - Configuration management of documents follows specified procedures

Validation - IEEE 12207.0, Paragraph 6.5

First, a decision must be made whether the project warrants a validation effort and the degree of independence of that effort. If project warrants validation, then establish a validation process. If project warrants an independent verification, then IV&V must be incorporated into the validation process.

The Validation process includes developing a validation plan consisting of the following:

- Items subject to validation
- Validation tasks to be performed
- Resources, responsibilities and schedule for validation
- Procedures for forwarding validation reports

It also includes the actual Validation activities listed below:

- Prepare test requirements, test cases and test specifications for analyzing test results
- Ensure test requirements, test cases and test specifications reflect requirements
- Conduct tests:
    - Test stress boundaries and singular inputs
    - Test software product for its ability to isolate and minimize the effect of errors; that is graceful degradation upon failure, request for operator assistance upon stress, boundary and/or singular conditions
    - Test that representative users can successfully achieve their intended tasks using the software product
- Validate that the software satisfies its intended use
- Test software as appropriate in selected areas of the target environment

IEEE 12207.1 Reporting Standards

IEEE 12207/1 includes reporting standards for all aspects of the Traditional Life Cycle. The following reports are recommended to fully document all V&V activities. Report requirements have been summarized.

IEEE 12207.1 Paragraph 6.23 - Software Verification Results Report

Purpose: Provide a record of the verification performed on software
Content: Report should include:

- Generic report information
- System identification and overview
- Overview of results, including
    - Identification of items verified

  - Dates of verification Detail results
  - Problems encountered
  - Verification criteria
- Verification results
- Rationale for decisions

IEEE 12207.1 Paragraph 6.27 - Test or Validation plan

Purpose: Describe plans for testing software items. Describe test environment to be used for testing, identify tests to be performed and provide schedules for testing activities

Content: Report should include:

- Generic plan info
- Test levels
- Test classes
- General test conditions
- Data recording, reduction and analysis
- Test coverage
- Planned tests
- Test schedules
- Requirements trace-ability
- Qualification testing environment, site, personnel, participating organizations

IEEE 12207.1 Paragraph 6.28 Test or validation procedures

Purpose: Describe test procedures

Content: Report should include:

- Generic procedure info
- Identification of test author
- Identification of test configuration
- Test objectives, requirements and rationale
- Test preparations (hardware, software, other) for each test
- Test descriptions
  - Test identifier
  - Requirements addressed
  - Prerequisite conditions
  - Test input
  - Expected test results
  - Criteria for evaluating results
  - Instructions for conducting procedure
- Requirements trace-ability
- Rationale for decisions

IEEE 12207.1 Paragraph 6.29 Test or validations results report

Purpose: Provide record of the qualification testing performed on a software item

Content: Report should include:

- Generic report info
- System identification and overview
- Overview of test results including
  - Overall assessment of the software tested
  - Impact of test environment
- Detailed test results including

  - Test identifier
  - Test summary
  - Problems encountered
  - Deviations from test cases/procedures
- Test log
- Rationale for decisions

16. APPENDIX H: Review Forms[9]

Software Requirements Specification Review

CONTENTS	VERIFIED	REMARKS
All software requirements completed.		
The requirements exhibit a clear distinction between functions and data.		
The requirements define all the information that is to be displayed to the user.		
The requirements address system and user response to error conditions.		
Each requirement stated clearly, concisely, and unambiguously.		
Each requirement testable.		
A preliminary version of the Acceptance/Software Test Plan, including verification matrix is completed.		
No ambiguous or implied requirements.		
No conflicting requirements.		
Performance requirements (such as response time and data storage requirements) stated.		
The requirements involving complex decision chains are expressed in a form that facilitates comprehension (decision tables, decision trees, etc.).		
Requirements for performing software upgrades specified.		
Real-time constraints specified in sufficient detail.		
Precision and accuracy of calculations specified.		
Possible to develop a thorough set of tests based on the information contained in the SRS.		
Assumptions and dependencies clearly stated?		
The document contains all the information called out in the SRS outline.		

SRS = Software Requirement Specification

Preliminary Design Review

CONTENTS	VERIFIED	REMARKS
The preliminary version of the Acceptance/Software Test Plan and verification matrix has been updated.		
Software design consistent with the software requirements.		
Deviations from the requirements documented and approved.		
All assumptions documented.		
Major design decisions been documented.		
Design consistent with the major design decisions.		
The design adequately addresses real-time requirements; performance issues (memory and timing); space capacity (CPU and memory; maintainability; understandability; loading and initialization; error handling and recovery; user interface issues; and software upgrades.		
Process Spec for each process accurate and complete.		
Dependencies on other functions, operating system kernel, hardware, etc., identified and documented.		
Human factor considerations properly addressed in those functions that provide a user interface.		
Design constraints, such as memory and timing budgets, specified where appropriate.		
Requirements for error checking, error handling, and recovery specified where needed.		
Interfaces consistent with module usage (missing interfaces or extra interfaces).		
Interfaces specified to a sufficient level of detail that allows them to be verified.		

Critical Design Review

CONTENTS	VERIFIED	REMARKS
All action items from the PDR have been resolved.		
Software structures and interfaces have been documented.		
The SDD is consistent and traceable to the IDD.		
Each of the elements in the IDD match the details in the SDD		
The input, processing, and output of each software unit via data and control flow have been supplied.		
Performance requirements, including timing, storage, and similar constraints have been documented.		
Performed independent design verification if applicable.		
Software test plan and verification matrix are completed.		
Any special security requirements have been met.		
Facilities including support and system software, compiler(s), coding and test tools, utilities, libraries, databases, etc. are ready and available for use.		
All related documentation (e.g., user's, operator's, maintenance, and diagnostic manuals) is up-to-date.		
Discrete quality and adequacy checks have been performed.		

PDR = Preliminary Design Review
IDD = Interface Design Document
SDD = Software Design Document

Test Readiness Review

CONTENTS	VERIFIED	REMARKS
All action items from the CDR have been resolved.		
Software integration is completed.		
Code inspection completed.		
Software components are ready for integration testing.		
There is consistency between test descriptions and user's manual(s).		
All user-level documents (e.g., operator and diagnostic manuals) are up-to-date.		
All pending changes to both the code and documents have been resolved.		
The test environment and support facilities including drivers, compilers, libraries, controls, and security are available and ready to use.		
A method of retaining and archiving test data and reporting the results of tests is being used.		
Discrete checks have been performed.		

CDR = Critical Design Review

Acceptance Review

CONTENTS	VERIFIED	REMARKS
All components have been properly updated.		
All components have been fully tested.		
Results of CodeTEST.		
All the review points are documented and addressed.		
Customer accepts the product.		

17. APPENDIX I: Code Inspection Form[9]

Code Inspection

CONTENTS	VERIFIED	REMARKS
No C++ constructs for dynamic memory allocation (the use of new and delete).		
No I/O streams.		
No inline functions substitution.		
No anonymous unions declared.		
No operator overloading.		
No reference objects declared.		
No C++ exception handling (The use of try, catch and throw).		
No pointer arithmetic.		
The design implemented completely and correctly.		
No missing or extraneous functions.		
Loops are executed the correct number of times.		
Each loop terminates.		
All possible loop fall-throughs correct.		
All CASE statements evaluated as expected.		
No unreachable code.		
Any off-by-one iteration errors.		
Any dangling ELSE clauses.		
Pointer addressing used correctly.		
Priority rules and brackets in arithmetic expression evaluation used as required to achieve desired results.		
Boundary conditions considered (null or negative values, adding to an empty list, etc.).		
The units of parameters and arguments match.		
Any input-only arguments altered.		
Any functions called and never return from.		
No string limits exceeded.		
All variables explicitly declared		
All arrays, strings, and pointers initialized correctly.		
All probable error conditions handled.		
The code allows for recovery from error conditions.		
Error messages and return codes used.		

References

[1] Klaus Havelund, Mike Lowry, SeungJoon Park, Charles Pecheur, John Penix, Willem Visser, Jon L. White. 'Formal Analysis of the Remote Agent Before and After Flight". Appeared in *Proceedings of 5th NASA Langley Formal Methods Workshop*, Williamsburg, Virginia, 13-15 June 2000. http://ase.arc.nasa.gov/pecheru/publi.html.

[2] Interview with Ric Hurt, Deputy Project Manager CLCS, September 6, 2001

[3] Deep Space One Website: http://nmp.jpl.nasa.gov/ds1/

[4] Douglas E. Bernard, Edward B. Gamble, Jr., Nicolas F. Rouquette, Ben Smith, Yu-Wen Tung, Nicola Muscettola, Gregory A. Dorias, Bob Kanefsky, James Kurien, William Millar, Pandu Nayak, Kanna Rajan, Will Taylor. *Remote Agent Experiment DS1 Technology Validation Report.* Jet Propulsion Laboratory, California Institute of Technology and NASA Ames Research Center, Moffett Field. http://nmp-techval-reports.jpl.nasa.gov

[5] P. Pandurang Nayak, Douglas E. Bernard, Gregory Dorais, Edward B. Gamble Jr., Bob Kanefsky, James Kurien, William Millar, Nicola Muscettola, Kanna Rajan, Nicolas Rouquette, Benjamin D. Smith, William Taylor, Yu-wen Tung. "Validating the DS1 Remote Agent Experiment". *Proceedings of the 5th International Symposium on Artificial Intelligence, Robotics and Automation in Space (iSAIRAS-99)* http://rax.arc.nasa.gov/publications.html

[6] Interview with Nicola Muscettola, NASA ARC, August 3, 2001

[7] Reference not used

[8] X-37 IVHM Experiment Website: http://ic.arc.nasa.gov/ic/projects/x37ivhm/index.html

[9] *X-37 IVHM Experiment Verification and Validation Plan, July 31, 2000.* NASA Ames Research Center, Moffett Field, California, USA

[10] *X-37 IVHM Experiment Software Test Plan Version: Preliminary, September 11, 2000.* NASA Ames Research Center, Moffett Field, California, USA

[11] *X-37 IVHM Experiment Acceptance Test Report Version: Final, June 1, 2001*

[12] Checkout and Control Launch System presentation by Bruce Hevey dated June 14, 2001

[13] Concept of Operations (ConOps), Checkout and Launch Control System (CLCS) 84K002200, dated January 29, 1998, Kennedy Space Center, Florida, USA (http://clcs.ksc.nasa.gov)

[14] System Validation Plan, Checkout and Launch Control System (CLCS) 84K07490-000-01, dated July 31, 2001, Kennedy Space Center, Florida, USA (http://clcs.ksc.nasa.gov/docs/test-specs.html)

[15] Application Control Board (ACB) Charter Level IV Change Control Board (CCB), Checkout and Launch Control System (CLCS) Document Revision A, dated August 23, 2000, Kennedy Space Center, Florida, USA (http://clcs.ksc.nasa.gov/docs/test-specs.html)

[16] Email from Glenn Semmel, Kennedy Space Center, dated July 19, 2001

[17] Report review sent via email from Ken Costello, project manager, NASA IV&V Facility, dated October 13, 2001

[18] *NASA Procedures and Guidelines NPG: 2820.DRAFT, NASA Software Guidelines and Requirements as of 3/19/01* (Responsible Office: Code AE/Office of the Chief Engineer), NASA Ames Research Center, Moffett Field, California, USA

[19] IEEE Standards 12207.0, 12207.1, 12207.2 located at the following web address (URL): http://ieeexplore.ieee.org/search97/s97is.vts?Action=FilterSearch&SearchPage=VSearch.htm&ResultTemplate=adv_crst.hts&Filter=adv_sch.hts&ViewTemplate=lpdocview.hts&query1=12207&scope1=&op1=and&query2=&scope2=&op2=and&query3=&scope3=&collection=jour&collection=conf&collection=stds&collection=pprint&py1=&py2=&SortField=pyr&SortOrder=desc&ResultCount=15

[20] *Software Considerations in Airborne Systems and Equipment Certification,* Document No RTCA (Requirements and Technical Concepts for Aviation) /DO-178B, December 1, 1992. (Copies of this document may be obtained from RTCA, Inc., 1140 Connecticut Avenue, Northwest, Suite 1020, Washington, DC 20036-4001 USA. Phone: (202) 833-9339)

[21] IEEE 610.12-1990, IEEE Standard Glossary of Software Engineering Terminology (This document may be purchased at http://shop.ieee.org/store/product.asp?prodno=SS13748)

[22] Software Test Plan (STP) – MIL STD 498 DI-IPSC-81438. http://www.comsivam.org/reference/498/

[23] Software Test Description (STD) - MIL STD 498 DI-IPSC-81439. http://www.comsivam.org/reference/498/

[24] *2nd Generation RLV, TA-5 IVHM, NASA In-House Task 1, IVHM Concept of Operations Document,* NASA Ames Research Center, Moffett Field, California, USA

[25] Microsoft Corporation. "Windows 2000 Server Resource Kit Online Books", MSDN Library Glossary. 1985-2000.

[26] Reid Simmons, Charles Pecheur, Grama Srinivasan. *Towards Automatic Verification of Autonomous Systems.* In: Proceedings of the 2000 IEEE/IRSJ International Conference on Intelligent Robots and systems, IEEE 2000.

[27] Jeffrey L. Whitten and Lonnie D. Bentley. *Systems Analysis and Design Methods Fourth Edition Instructor's Edition.* Irwin McGraw-Hill 1998 p. 583.

[28] Whatis?com http://whatis.techtarget.com/definition/0,,sid9_gci284015,00.html

[29] Interview with Peter Engrand, Kennedy Space Center, September 14, 2001

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY *(Leave blank)*

2. REPORT DATE
April 2002

3. REPORT TYPE AND DATES COVERED
Contractor Report

4. TITLE AND SUBTITLE
Survey of NASA V&V PRocesses/Methods

5. FUNDING NUMBERS
NRA 8-30/TA-5 Northrup Grumman

6. AUTHOR(S)
Charles Pecheur (RIACS) and Stacy Nelson (CSC)

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
RIACS and CSC
Mail Stop 269-1
Ames Research Center
Moffett Field, CA 94035-1000

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
National Aeronautics and Space Administration
Washington, DC 20546-0001

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
NASA/CR-2002-211401

11. SUPPLEMENTARY NOTES
Point of Contact: Charles Pecheur, Ames Research Center, MS 269-1, Moffett Field, CA 94035-1000
(650) 604-3588

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Unclassified — Unlimited
Subject Category 61 Distribution: Nonstandard
Availability: NASA CASI (301) 621-0390

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

The purpose of this report is to describe current NASA V&V techniques and to explain how these techniques are applicable to 2nd Generation RLV IVHM software. It also contains recommendations for special V&V requirements for IVHM.

This report is divided into the following three sections

- Survey - Current NASA V&V Processes/Methods
- Applicability of NASA V&V to 2nd Generation RLV IVHM
- Special 2nd Generation RLV IVHM V&V Requirements

14. SUBJECT TERMS
Verification and Validation; Autonomous Systems, Model-Based Reasoning, Formal Methods, IVHM

15. NUMBER OF PAGES
91

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT
Unclassified

18. SECURITY CLASSIFICATION OF THIS PAGE
Unclassified

19. SECURITY CLASSIFICATION OF ABSTRACT

20. LIMITATION OF ABSTRACT