Management of Independent Software Acceptance Test in the Space Domain: A Practitioner's View

Kristin Wortman
Johns Hopkins Applied Physics Laboratory
11100 Johns Hopkins Road
Laurel, MD 20723-6099
240-228-9634
Kristin.Wortman@jhuapl.edu

Abstract—**This paper discusses management of the independent acceptance test activity in a space domain software development cycle. The goal of acceptance test is to independently verify functional software requirements to achieve a high level of confidence in the software system before release for operational use. Key management areas have been identified through internal independent acceptance test practice at Johns Hopkins University Applied Physics Laboratory for NASA's Radiation Belt Storm Probes software system. The key management areas have been categorized as People, Constraints, Needs and Quality (PCNQ). The supporting framework for management of the PCNQ factors encompasses the established lines and means of communications for the project, and the organizational culture. The PCNQ areas and their relationships must be strategically managed within the framework for an effective independent acceptance test of a spacecraft's mission software system. The concepts presented in this paper will promote understanding of the management role of an independent software acceptance test activity in the space domain.**

TABLE OF CONTENTS

1. INTRODUCTION

The software system for NASA's Radiation Belt Storm Probes (RBSP) mission was developed by the Space Department at the Johns Hopkins University Applied Physics Laboratory (JHU/APL) located in Laurel, Maryland. An integral phase of our unmanned spacecraft system development process includes an independent verification of requirements for the mission operations, flight and simulations software. The goal of our acceptance test is to independently verify the software requirements in order to achieve a high level of confidence in the critical functionality of the spacecraft's mission software system. The software engineering group considers acceptance test as the final verification layer before the software is released to the validation phase in the project, spacecraft integration and test. On previous spacecraft projects within the software engineering group, it was noted that management of acceptance test encountered challenges in meeting schedule and budget constraints. Through the RBSP software acceptance test experience, these management challenges were identified. A model was developed to provide management guidance for acceptance test leads who will work on future spacecraft projects.

The discussion commences with an overview to provide the context of acceptance test relative to the overall test structure followed for verification and validation of our spacecraft's mission software system. Following the test structure discussion, the acceptance test process adopted for independent verification of the critical RBSP software requirements is explained in detail. The focus of the discussion will shift to this practitioner's model developed to highlight the key management areas found to be crucial to effective acceptance test. Elaboration on these key management areas will include a discussion of their contributing elements and the relationships that exist between them and the supporting framework.

After reading this paper, the reader will gain an understanding of our spacecraft's mission software test structure and the independent verification process being implemented to verify the RBSP critical software requirements. Most importantly, the reader will learn about the software acceptance test management areas that were identified through a practitioner's experience. The model was developed to assist new software acceptance test leaders to effectively manage the day to day activities. The model can also benefit various levels of management by

978-1-4577-0557-1/12/$26.00 ©2012 IEEE

enhancing understanding of the independent software acceptance test process and the associated management challenges.

2. Test Structure Overview

The overall test structure followed in the Space Department at JHU/APL for verification and validation of a spacecraft's mission software system can be viewed as four distinct layers. The first three test layers in the test structure will be referred to in this paper as unit, system and acceptance. Completion of the first three layers is the responsibility of the software engineering group during the Software Development Life Cycle (SDLC). The fourth test layer will be referred to as integrate (or integration). Integration test is the responsibility of the JHU/APL's system engineering group and is performed during the project's spacecraft integration and test phase. Figure 1 provides the graphical representation of the four layer test structure. Successful completion of the four test layers provides the informal and formal verification and validation process being implemented for the RBSP mission's software system.

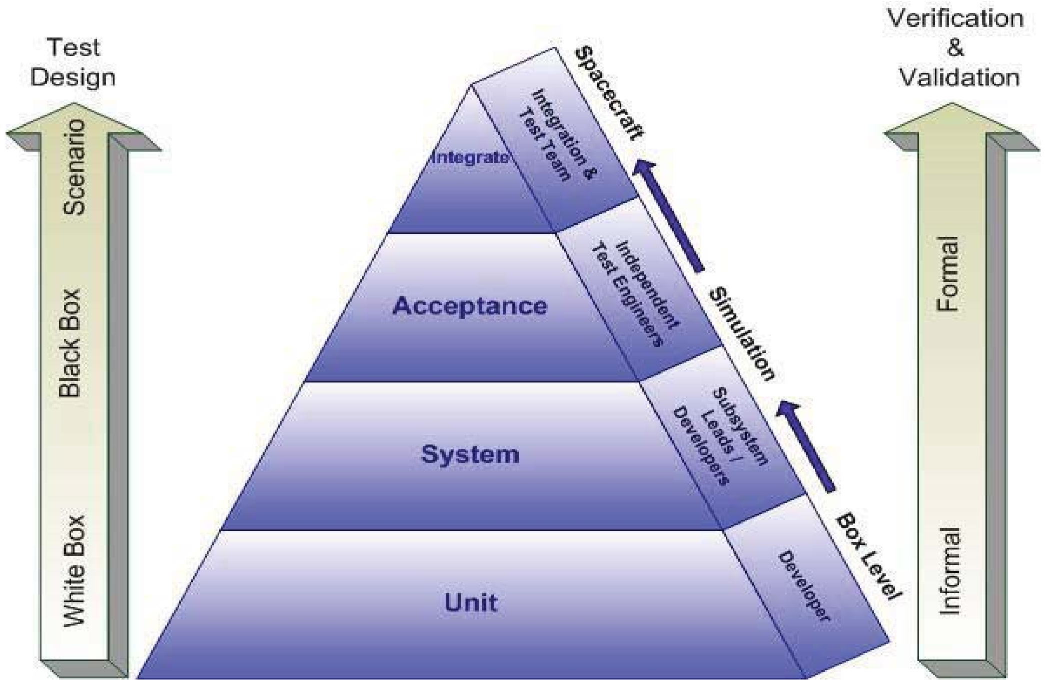


Figure 1. Graphical Representation of Spacecraft Mission Software Test Structure

In Figure 1, the four layers of the test structure are represented together to form a pyramid. The significance of the pyramid models the sequential occurrence of each layer of test and the level of functional detail covered by the testing. The first layer, unit test, is viewed as the lowest level (bottom of the pyramid) and provides the widest functional coverage (e.g., code logic path testing). The last layer, integrate test, is viewed as the highest level (top of the pyramid) with the focus on operational functionality (e.g., scenario). Four important aspects are also noted in this layered test structure:

- The approach followed for test design.
- The responsible test team or person who designs and executes the tests.
- The test environment used for test execution.
- The formality of the verification and validation activities.

The comprehensive goal of the first three test layers (unit, system, acceptance) in the test structure is to verify the required software functionality is correct and complete. At the acceptance test layer, each software requirement is targeted and verified following a defined test process.

The goal of the fourth test layer (integrate) is to verify and validate the mission level requirements. The software system is used by the integration and test team to check out the hardware components as they build the spacecraft. These tests include comprehensive performance and operational scenario tests. In addition, the mission operations team designs and executes simulation tests on the spacecraft in preparation for pre-launch and post-launch activities.

As the test structure evolves from unit to the integrate test layer, the approach followed for test design also evolves. The vertical directional arrow on the left side of Figure 1 depicts the gradual change in the test design approach for each test layer. The test design approach used at the unit test layer is white box. The transition in test design approach to black box begins during the system test layer and fully matures during the acceptance test layer. Finally, the test design evolution completes during the integrate test layer as operational scenario.

The distinction in the approach between white box and black box test design can be measured by the amount of internal knowledge of the code required by the person who is to design the test. The white box test design exercises logic paths through the code and is efficiently conducted by the software developer. In contrast to the white box test design, the black box test design requires little or no internal knowledge of the code workings. Designing a black box test requires limited knowledge to pair a set of data inputs with expected output. In addition, white box tests are executed earlier in the software development cycle compared to black box tests that are executed at a later phase in the software development cycle [3].

The responsible test team or person for each test layer is noted in Figure 1. The first two test layers (unit and system) are performed solely by the development team. At the third test layer (acceptance) the responsibility transitions to a team of independent test engineers. We define independent as not being involved in the development of the software component under test. The responsible team changes again at the fourth layer of the test structure to the integration and test team. As mentioned earlier in this section, at the fourth layer (integrate) of the test structure, the test responsibility transitions from the software engineering group to the system engineering group.

The environment used to conduct test executions is another consideration in our test structure. At the unit test layer, white box tests are executed using desktop development tools and the target hardware component, when it becomes available. During the system test layer, the development team integrates the software components and migrates to the simulated test environment. The simulated test environment consists of various hardware components, interface emulators, flight software and ground software. Acceptance tests are executed in the simulated test environment. Eventually the flight hardware components are integrated on the spacecraft and this environment is used to execute integration tests and to conduct mission operation simulations. The evolution of the test environment is scaling upward along the right side of each layer in the test structure modeled in Figure 1.

The evolution of the formality of our verification and validation process for each layer in the test structure is modeled with the vertical directional arrow located on the right side of the pyramid in Figure 1. Our SDLC process requires each version of software be documented, and approved by the Mission Software System Engineer (MSSE), Software Quality Assurance (SQA) and the software development lead before release to acceptance test for requirements verification. The role of the MSSE is to provide oversight for the development and verification of the spacecraft's mission software system. The role of SQA is to ensure the software processes are followed during the SDLC. In comparison to the unit and system test layers, the acceptance test layer follows a rigid process to independently verify the software's requirements.

3. Independent Software Acceptance Test Process

Test engineers follow an established SDLC process referred to by our software engineering group as Independent Software Acceptance Test (ISAT). Figure 2 provides a graphical representation of the ISAT process being implemented for verification of the critical functionality of the RBSP software system.

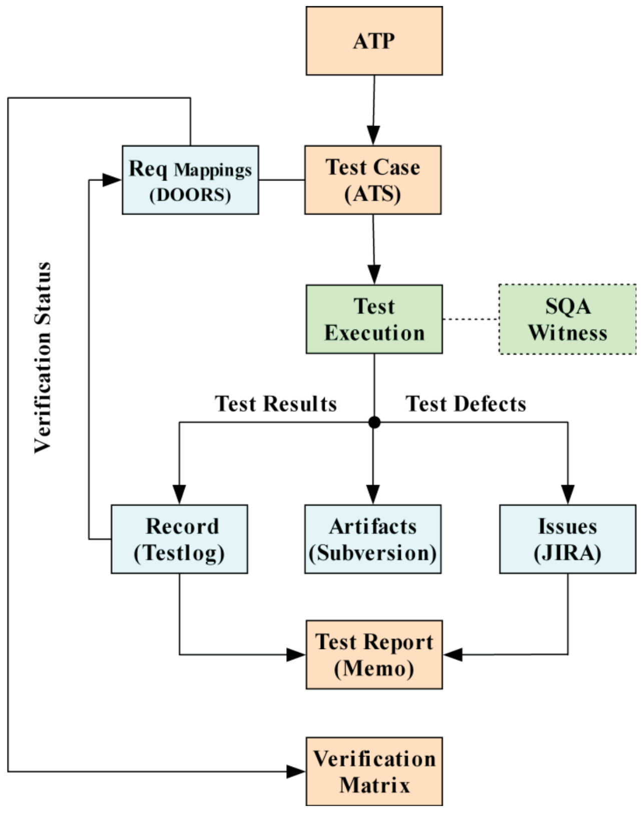


Figure 2. RBSP Independent Software Acceptance Test Process

The ISAT process incorporates a set of test documentation and records. The first document, Acceptance Test Plan (ATP), is written by the acceptance test lead to provide project management information and to establish the strategy and procedures to be followed by the test engineers. The second document, Acceptance Test Specification (ATS), is written by the test engineers to provide the test designs and procedures to be implemented for formal test executions.

The ATP is completed, reviewed and approved by the MSSE, SQA, and software lead before the test engineers commence development of the ATS. The first step in composing the ATS is to identify each critical software requirement and to establish a link to a unique test case identifier. A requirements management tool (Telelogic DOORS) is used to establish this linkage. The requirement set established for each test case defines its verification objectives. The verification objectives are used by the test engineer to develop the details of the test case. The ATS evolves with each release of software when test cases are added to verify new functionality. The ATS is complete when test cases are defined to cover the full requirement set.

Both the ATP and ATS are subject to a peer review process. The peer review process requires a panel consisting of a chair person, subject matter expert and independent reviewer. SQA is also invited to attend all peer reviews. Any concern that results from a peer review is captured and tracked through concurrence on the concern's resolution.

Concurrence on all issue resolutions for a test case is required prior to test execution. All scheduled test executions are conducted by the acceptance test engineers and SQA is notified to witness.

Commercial software tools are used by test engineers to track verification coverage and to record test execution information. Upon completion of test execution, artifacts are stored in a repository using GForge Subversion, software defects are documented using Atlassian JIRA and a record of the test execution is created in a Passmark's Testlog database. The commercial software tools are noted in the ISAT process depicted in Figure 2.

Acceptance test is performed on each software release, therefore a test report is prepared at the end of each acceptance test cycle. The written report summarizes test activities and is approved and distributed to project management and spacecraft subsystem leads. A typical ISAT report contains:

- A summary of the test cases
- A record of the test case executions
- A record of uncovered software defects
- Any recommendation for improvement of the software functionality
- A conclusion on the fitness for use

4. Management of Independent Software Acceptance Test Process

The ultimate goal of our internal ISAT process is to independently verify the critical software functionality in order to achieve a high level of confidence in the spacecraft's mission software system. Achieving this confidence level is necessary before release of the software for use during integration test activities and mission simulations. Through RBSP's experience, key areas for ISAT management were identified as being crucial to achieve this confidence level and to remain within the project's budget and schedule. A model was developed to represent these key ISAT management areas along with their contributing elements and relationships. There are two objectives for this model. The first objective is to provide direction to software acceptance test leads in support of future spacecraft missions. The second objective is to promote management awareness of the ISAT process and its management challenges.

Four ISAT management areas, People, Constraints, Needs and Quality (PCNQ), were identified and are referred to in this practitioner's model represented in Figure 3.

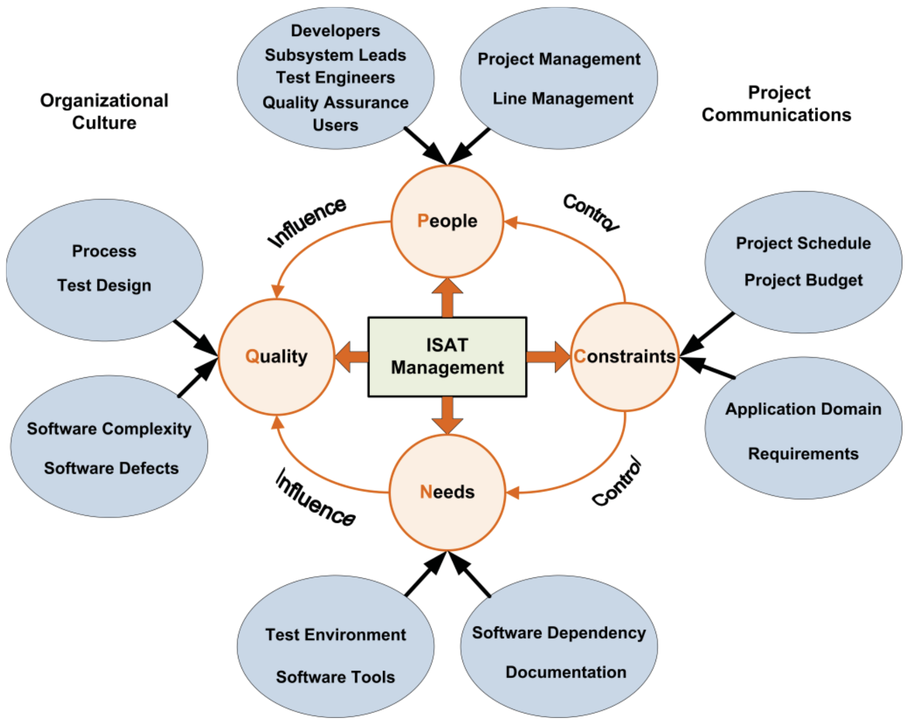


Figure 3. Practitioner's PCNQ Model

The organization's culture and the established lines of project communication comprise the supporting framework for ISAT management. The four management areas and their relationships with the supporting framework comprise what is referred to in the remaining sections of this paper as the PCNQ Model.

In Figure 3, the center rectangle (yellow color) represents ISAT management. The arrows flowing outward from ISAT management indicate each area a lead needs to be aware of, communicate with, or provide direction to, in order to successfully complete the activities defined in the acceptance layer of the test structure. For RBSP, the role of the ISAT lead is performed under the direction of the MSSE who reports to project management.

The four ISAT management areas are represented in the PCNQ Model with a circle (orange color). The relationships that exist between the ISAT management areas are represented with labels on the directional connectors (orange color). The ovals (blue color) represent the elements that contribute to each ISAT management area. The discussion in the remaining sections of this paper will focus on the four ISAT management areas and their relationships.

Understanding the supporting framework for ISAT management is important when considering the contributing elements that are represented in the PCNQ Model. The project (e.g., RBSP) and the organization (e.g., JHU/APL) form the supporting framework for ISAT management. In Figure 3, the PCNQ Model, the supporting framework is represented in the background as the organizational culture and the project communications.

Organizational Culture

The organization's culture directly affects the people who perform the work on a project and how they perform the technical work. A healthy organization's culture provides important motivators and support to the people working on the project [1]:

- Benefits (e.g., health, education)
- Job classifications
- Work environment (e.g., office space, health facilities, social activities)
- Organizational procedures, policies and processes (e.g., flexible work schedules, telecommuting, EEO, certifications)
- Job growth potential

For example, test engineers view a healthy organizational culture as one that nurtures them by not singling them out from the developers and engineers in the rest of the organization [1]. To create a positive culture, test engineers, developers and engineers should be assigned the same job classification. At JHU/APL there exists five technical staff classifications that are determined by education level, years of professional experience and technical contribution. These five technical job classifications promote job growth potential and do not single out the test engineer based on job function.

Physical working conditions, environment, flexible work schedules and telecommuting policies are considered part of the organizational culture. For example, the test simulation environment used by the ISAT team is a very expensive and limited resource. RBSP is not considered a classified project, therefore virtual network connections were allowed that enabled authorized test engineers secure remote access to the test simulation environment. Having the ability to work from home boosted test engineer productivity by allowing a flexible work schedule to use test resources during the evenings and weekends without making the commute to the laboratory.

Project Communications

Procedures to disperse information to the team are established early in a project. Learning the information flow and the commercial tools used to track and store information are essential for effective management of test activities. Part of information flow is knowing the who, when and where for preparing status reports and disseminating information to test engineers. For example, a commercial tool, Windchill Product Lifecycle Management System, was used on RBSP for tracking written communication (e.g., memos and test reports), configuration management and storage of technical documentation (e.g., ATP and ATS). Knowing how to manipulate the tool set allows for mobility in extracting pertinent information. Establishing the communication lines and means with supporting subsystem teams working on the project is of equal importance.

People

Effective software verification relies on a group of people who possess a variety of management and technical skills. The management category includes those people who monitor the activities of the project, and those people who monitor the ISAT activities. The technical category includes those people who provide technical knowledge on the software system, and those people who provide technical knowledge and skill to implement the activities mandated by the ISAT process.

In the management category, there exists a difference in the lines and means of communication used for providing ISAT status reports. Project management oversees the building of the spacecraft system (hardware and software). For RBSP, ISAT status is included in the SDLC project report that is completed by the MSSE. The SDLC status report is provided on a regular basis (e.g., monthly) and is essential for project management to track overall progress and completion of mission milestones. The ISAT status report prepared for the project management audience should contain metrics. Examples of ISAT report metrics are:

- Number of uncovered software issues
- Number of executed tests
- Number of verified requirements
- Number of requirements to be verified
- Percentage of overall ISAT completion for a software release

Line management consists of the overseers of the activities performed in the ISAT process. Face-to-face meetings with line management are normally held on a regular (e.g., weekly, biweekly) basis. The focus for ISAT status reports to line management include status on ongoing and recently completed work. Examples of ISAT reporting items to line management include:

- The current test activity that is in progress
- A positive or negative shift in schedule that affects resource needs
- Any problem or obstacle to complete a test activity in order to meet a planned schedule milestone
- Something is needed from another department
- An ATS review is going to be, or has been, conducted since the last report
- The status of test reports

The technical group refers to those who contribute a skill or provide knowledge directly used to implement the ISAT process activities. Test engineers, leaders of the development teams, the developers and the software quality assurance representative are included in the technical group. Test engineers implement the required activities outlined in the ISAT process and these activities require the most management. Providing direction and feedback to test engineers, and monitoring the progress on test activities are essential to meet schedule milestones.

Designing the detailed steps required for a test is a challenge. Test engineers need to acquire functional knowledge on the software components. It may be necessary at times to facilitate communications between the other members in the technical group (e.g., developers, subsystem leads). Communications between the technical group members can be delivered orally or written. It is recommended that oral meetings be followed up with written documentation (e.g., email or memo). Written documentation will reinforce understanding and eliminate misunderstanding. When conducting meetings to gather technical information, the agenda should be focused and distributed to invitees prior to the meeting date. The invitees to the meeting should be limited to core people involved in the subject matter. From experience, small-sized meetings targeted to extract technical information are more productive than large-sized meetings.

RBSP's ISAT spanned several subsystems (e.g., spacecraft flight software, mission operations ground software, science operations ground software and the test simulations software). The RBSP ISAT approach was the formation of separate test teams, one per subsystem. This strategy proved effective during the ISAT of earlier software versions, specifically, for the ground and flight subsystems. Each team maintained focus on the subsystem's functionality to be verified. As the later software releases were delivered to ISAT, these test teams merged their acquired knowledge to jointly design tests to verify end-to-end functionality and to put stress loads on the mission software system.

Constraints

Constraints affect the people who perform the work and control the availability of the things necessary to complete ISAT activities. These constraints can be divided into two categories. The first category relates to the project, the budget and schedule. The second category relates to the spacecraft domain and the software functionality required to meet the objectives of the mission.

Project budget and schedule naturally impose constraints on the technical people involved in the acceptance test layer of the test structure. Estimation of the resources for the SDLC are made during an early phase of the project. These estimates are normally derived by referencing software development metrics for a similar mission, and accounting for the ISAT strategy to be followed. The estimates along with the schedule milestones are used to determine the number of test engineers to perform the test activities throughout the duration of the SDLC.

Software testing is proportionally a very small task when building a complex spacecraft system. The project schedule drives when the functionality is to be delivered in a software version to the target user community. The ISAT management challenge is to properly align the number of test engineers with the volume of requirements to be verified in each software release. Prioritizing verification of functional areas will ease workload stress on test engineers. A simple and effective tool is a spreadsheet that details work assignments and test priority. Such a spreadsheet proved valuable on RBSP in maintaining test engineer focus. Updates to this spreadsheet were made to each test case and distributed weekly to the test engineers. Answers to the following questions will assist with ISAT planning for a software release:

- What software requirements are being delivered in the software release?
- Is all the functionality available in this software release to perform and maximize test execution at this time?
- Are the test simulation environment and the right software tools going to be available?
- What test engineers are available during this timeframe and at what percentage of their time?

The constraints also pose limits on what functionality is to be verified and how the functionality is to be verified in the software. The space domain and software requirements are considered in this category and will be discussed in the next two paragraphs.

The space domain (e.g., deep space planetary, low earth orbit) can constrain the definition of the software requirements. Detailed discussion of the space domain is beyond the scope of this paper, but it's important to review the mission objectives to understand the context of the software system. The software requirements are derived from the mission level requirements. Once the software requirements are approved by the designated parties, they dictate what is to be verified in the acceptance test design. In essence, the quality of test design relies on a well-defined software requirement set. Early involvement in the specifications definition phase is important to determine the accuracy, completeness and testability of each requirement. For RBSP, the ISAT lead was invited to attend software requirement reviews for each subsystem during the early phases of the project. Acceptance test planning commenced once the Software Requirements Specification was formally agreed upon and released to the ISAT team.

To summarize, the relationship between people and the constraints can be simply viewed in the PCNQ Model as the people are controlled by the constraints. Understanding this relationship from the ISAT management perspective will aid the lead in planning test activities. Another relationship exists between the constraints and the needs of ISAT to perform software verification. Elaboration on this relationship will be discussed in the next section.

Needs

The next management area in the PCNQ Model defines the needs to be satisfied in order to fully implement the activities outlined in the ISAT process. Hardware and software resources are defined in the first category. The dependency on another subsystem's software functionality and supporting technical documentation is considered in the second category.

The test simulation environment and tool set are the most important resources necessary to execute the acceptance tests and to prove correctness of the software functionality. A reliable test simulation environment includes the software system to emulate spacecraft component hardware interfaces (e.g., RF communications, science instruments, power system), engineering model components (e.g., transceiver, integrated electronics module), the flight software, and a supportive interface to complete the end-to-end test environment (e.g., ground software).

The ISAT management challenge is to monitor the delivery schedule for the test simulation environment, test tools and the software release to be acceptance tested. The proper alignment of these three factors will determine the test readiness of the software functionality targeted to be verified. If the features of the test simulation environment are not aligned with test design, the consequence is a delay in test execution until a later software release. The test execution load required in the final software delivery may exceed the number of available test engineers. For example, if a feature of the ground software (e.g., uplink or downlink protocol) is delivered in a early software release, test execution may have to wait until the functionality is supported in the flight software. The risk of not meeting the final schedule milestone is escalated with each delay in end-to-end test execution.

Early involvement in software reviews aids planning to properly balance the acceptance test activities and to prioritize tasks. In the case where a feature in a supporting software release is not delivered when expected, a strategy is to have test engineers commence with preliminary test designs. The test details and coding of procedures can be resumed once the complete test simulation environment is delivered.

The test tool set provides the means to manipulate the input data (e.g., spacecraft and instrument commands) and the output data (e.g., telemetry) for effective test design. Test designs require a known set of test data for execution of nominal and exception functional test cases. Forecasting, development, and verification of this tool set can be a challenge. The ISAT lead needs to be proactive to determine the appropriate requirements for the tool set necessary to effectively design requirement verification tests.

The project budget allocated to the software team can limit custom tool development. It may be necessary for acceptance test engineers to develop some of their own test tools. Ideally this requires a test engineer with software development experience or with knowledge of a scripting language. For example, a tool was needed to parse through RBSP test artifacts to identify spacecraft commands in the database, and to quantify how many times each command was exercised during test executions. The study was to be completed late in the ISAT schedule. Early in the project, a test engineer, who was only available at a 25% level for a few months, was assigned to develop this custom tool using the Perl scripting language. Having the test tool available during the final ISAT phase proved valuable in timely completion of the required study.

Lack of complete, accurate and timely documentation to support software, test environment and tool usage has a large impact on test design and test engineers' productivity. Part of the software delivery process to ISAT is for the subsystem to provide supporting usage documentation. If the documentation on how to use the software is not complete when ISAT commences, this will have an adverse affect on the productivity of both the test engineer and the developer. In this case, the test engineer will have to meet frequently with the developer in order to clarify understanding on how to use the software and its expected behavior.

Quality

Quality is the last ISAT management area to be discussed in the PCNQ Model. There are many factors that influence the quality of a software system. The PCNQ Model scopes these factors to two categories. The first category relates to the overall ISAT process and the specific test design used for verification of the software requirements. The second category includes characteristics intrinsic to software, but not within the control of ISAT management. However, knowledge and monitoring of these elements does provide valuable ISAT insight.

The technical people have a major influence on the test design used, and the verification process they followed, during the software development cycle. Having the right balance of technical skill and knowledge on the ISAT team is essential for building quality test designs. Software design reviews provide insight into the scope and complexity of the functionality required by the spacecraft's mission software system. Acquiring this knowledge is instrumental for ISAT leads who need to train and provide technical direction to test engineers.

In section 3, the discussion focused on the activities outlined in the ISAT process. Ensuring these activities are accurately implemented adds the quality factor to the verification process. The essential part of the ISAT process is to provide feedback to the developers, to make recommendations for improvement, and to report defects uncovered in the software. Software defects are a good measure of software quality. Closely monitoring the defects for patterns aids the ISAT lead to determine the right balance for test depth and to identify the regression test suite used in the development cycle. The final round of ISAT will yield the ultimate goal to determine a confidence level in the functionality of the software system.

Quality and availability of resources used to design and execute a test influences the quality of the ISAT process used to verify the correctness of the software functionality. For example, the number of test simulation environments for a mission are often limited due to cost. In order to ensure proper time spent on test activities, the test engineer requires use of the test simulation environment. Test simulation environments are heavily used during peak software development and during the different layers of test. For RBSP, there was a dedicated simulation environment for use by the ISAT team. However, this was not always adequate to accommodate the number of test engineers simultaneously working during core hours (e.g., ground and flight software ISAT). In addition to test engineers working off core hours, negotiation with other teams was necessary to secure time on another test simulation environment for a short period of time.

The organizational culture influences the elements of quality. The processes and standards adopted and followed at the organizational level can impact the way things are expected to be done during ISAT. These include achieved process improvement ratings (e.g., SEI CMMI) and certifications (e.g., ISO 9001). For example, during an early RBSP phase, an audit was performed for ISO 9001 certification. During the ISO 9001 audit, it was noted that ISAT did not have a process to provide evidence of a test procedure review. The test procedures are developed to automate the steps, using a scripting language, that are defined in the test design. The test designs are peer reviewed. Our current ISAT process includes a test procedure review conducted by the lead or another designee from the acceptance test team. Evidence of the test procedure review is kept by making a notation in the test execution record database.

Knowledge of the organization and how things were done on previous projects enhances the understanding of lessons learned. Research should be done on legacy ISAT efforts, test simulation environment designs, and software test tools used by ISAT. This is especially beneficial if the software system reuses components from previously completed missions.

5. Summary

The discussion in this paper has covered the spacecraft mission's software verification and validation structure implemented in the Space Department at the Johns Hopkins University Applied Physics Laboratory for RBSP. The overarching test structure was presented to view the four distinct layers (unit, system, acceptance and integrate) together as a pyramid. The test structure provided the context and scope for ISAT. Distinct evolution in the team who performs the testing at each test layer with their verification objective were included in the discussion. The approach to test design in each test layer was accentuated along with the environment used to conduct test execution.

The next topic for discussion focused on the internal acceptance test process being implemented as part of the SDLC for independent verification of the critical RBSP software requirements. The objective of the ISAT process is to efficiently verify the critical functionality to gain a high confidence level in the software before release to the integration and test team and for use in mission simulations.

Four crucial areas of management (People, Constraints, Needs and Quality) were presented as the PCNQ Model. The contributing elements to each management area in the PCNQ Model were discussed in some detail. The supporting framework for ISAT management, the organizational culture and project communications, were mentioned along with their influence and impact on test activities. Examples from the RBSP ISAT experience were cited in this paper to educate those involved in, or planning to be involved in, acceptance test management.

Effective management of independent requirement verification for a spacecraft's mission software system requires a leader:

- To acquire knowledge of the overall verification test structure to be implemented for the project.
- To understand the lines and means of communications to be used throughout the project.
- To acquire knowledge of the independent software acceptance test process to be used in the software development life cycle.
- To understand the impact of the organizational culture on the management areas defined in the PCNQ Model.
- To understand the management areas represented in the PCNQ Model and their role in the success of ISAT.

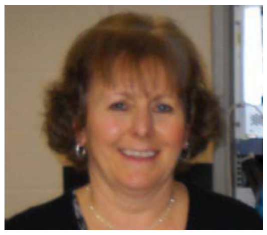
REFERENCES

[1] Srinivasan Desikan, Gopalaswamy Ramesh, Software Testing Principles and Practices, Pearson Education, 2006.

[2] Peter Farrell-Vinay, Manage Software Testing, Auerbach Publications, 2008.

[3] Roger Pressman, Software Engineering A Practitioner's Approach, Seventh Edition, McGraw Hill, 2010.

BIOGRAPHY



__Kristin Wortman__ received a B.S. in Computer and Information Science from University of Maryland University College, in 1990, an M.S in Software Engineering, a joint degree from University of Maryland and University of Maryland University College in 2000. She has been with JHU/APL for more than 4 years. She is the software acceptance test lead for Radiation Belt Storm Probes. Prior to joining JHU/APL senior professional staff, she was a senior computer scientist for Computer Sciences Corporation (CSC) for two tenures totally twenty-two years. She supported a team of NASA Goddard scientists and developed instrument flight and data analysis software for two STEREO spacecraft science instruments. When not working for CSC she was employed by several software companies as a developer for another twelve years supporting data processing and analysis software development for various NASA missions. She is also an adjunct associate professor for University of Maryland University College in the Computer and Mathematical Sciences Department Undergraduate Studies since January 2001.

Kristin can be reached at: kristin.wortman@jhuapl.edu or (240-228-9634).

ACKNOWLEDGEMENTS

The author would like to express her thanks to individuals that have supported this work and offered some good dialogue during the preparation of this paper. Debbie Clancy (JHU/APL) for being a good mentor, for providing direction on the ISAT process, and for providing a constructive technical review of this paper. Roland Lang (JHU/APL) for many lengthy discussions on software requirement verification and lessons learned from previous ISAT efforts on spacecraft software systems.