Thinking Ahead to Verification and Validation

Louis S. Wheatcraft
Requirement Experts
(281) 486-9481
louw@reqexperts.com

Abstract Second only to re-work, a project's largest costs are involved in verification and validation. Failing to adequately plan for verification and validation from the beginning of the project places the project at high risk for cost overruns and schedule slips. To mitigate these risks project teams must outline their concepts for verification and validation during the scope definition activities; addressing the following questions: "Who will verify the requirements, and where and when will it be done?" "Who will do system validation, and where and when will it be done?" "What facilities, support equipment, simulators, emulators, or models are required?" "What are the interfaces involved during verification and system validation?" When you start writing your requirements, addressing verification as the requirements are written insures a better set of requirements and can reduce the cost of verification and system validation. As each requirement is written, the following questions need to be addressed: "Is this requirement verifiable?" "What constitutes proof that the requirement has been verified?" "What primary method will be used to verify this requirement?" Failing to address these questions early can result in unpleasant surprises during your verification and system validation activities, putting the project at risk of rework, schedule slips, and budget overruns. The emphasis of this paper is to present best practices, tools, and proven methods project teams can use to plan for verification and system validation from the beginning of their project and thus avoid the risks of not doing so.

Verification and Validation

It is common to see verification and validation used together as "V&V" as though they are processes done concurrently. Some organizations use the terms interchangeably. This is not correct. Verification is the process of proving the designed and built system of interest (SOI) meets its requirements. Verification addresses the question: "Did we build it right?" We don't verify requirements; we verify that our system meets its requirements.

Defining validation is more complicated because the word "validation" is ambiguous unless it is used with a modifier that indicates what is being validated. Are you validating your requirements, your design, or your system? If you are talking about "requirements validation", you are assessing whether or not the set of requirements clearly, completely, correctly, and consistently communicates stakeholder expectations in a language (requirements) that can be understood by the designers. Requirement validation activities address the question: "Did we define the right thing [to be built]?" The Systems Requirements Review (SRR) is a requirement validation activity. If you are talking about "design validation", you are assessing whether or not your design completely addresses the requirements. Major design reviews (System Design Review, Preliminary Design Review, and Critical Design Review) are examples of design validation activities. In contrast, "system validation" is the process of proving the designed, built, and verified System of Interest (SOI) meets the stakeholder expectations and can accomplish its intended purpose. You are addressing the question: "Did we build the right thing?"

The focus of this paper is on verification and "system" validation planning. (For more insight into requirement validation, see references 1 & 2.)

A Winning Product vs. Risk

At the end of a project, all project teams want to be able to say they have built the right thing and that the product meets stakeholder expectations while accomplishing its intended purpose. I call this delivering a winning product. A winning product is defined herein as: "*A product that delivers what is needed, within budget, within schedule, and with the desired quality*." The goal of all projects should be to deliver a winning product.

A simple and practical definition of risk is: "*Anything that can prevent you from delivering a winning product!*" Failing to plan for verification and validation from the beginning of the project is a major risk to the project that can result in massive cost overruns and schedule slips as well as a product that does not meet requirements and fails to meet stakeholder expectations. Given these impacts, all project managers need to mitigate this risk from the beginning of their project. Recognizing this is critical to being able to deliver a winning product.

Requirements best practices include the planning for verification and validation activities from the beginning of the project and continuing to address verification and validation activities throughout the product life cycle. To reduce the risk to your project associated with failing to address verification and validation activities, it is critical that you follow these best practices by including verification and validation planning from the beginning of your project starting with scope definition and continuing throughout the requirement definition and management processes.

The following sections introduce these best practices as well as discuss some of the tools and activities that will help you plan for verification and validation.

Impact of requirement defects on Verification and Validation Cost and Schedule

Requirements are the single element that ties all the product development lifecycle processes together. Defective requirements have a direct and significant impact on your project's cost and schedule at the end of the project when verification and system validation activities are performed.

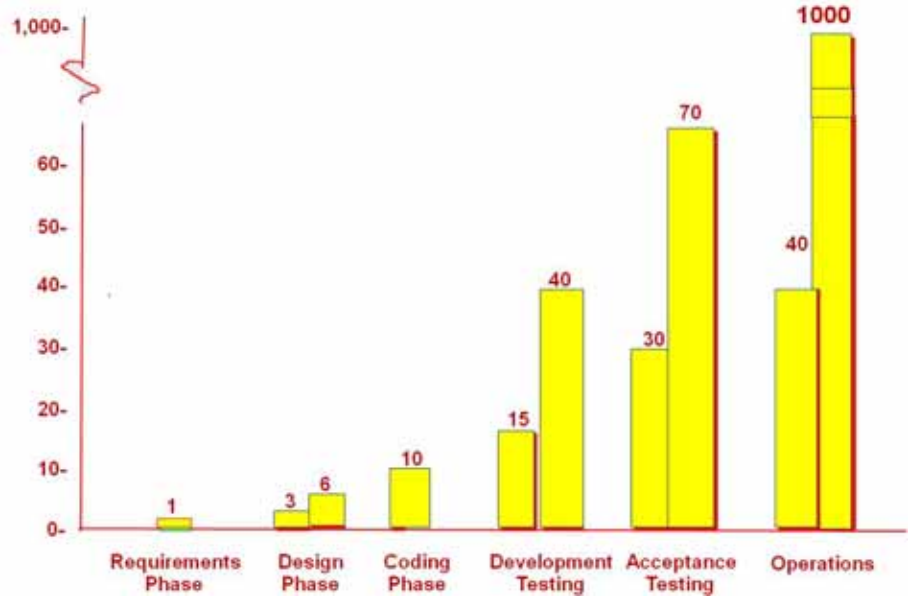

Figure 1: Cost to fix requirement defects

As shown in Figure 1[Hooks 2001], the cost to fix requirement defects (re-work) increases exponentially as the product lifecycle progresses. This chart illustrates the order of magnitude costs of finding and fixing defects as we progress in the life cycle. What this figure shows is that the cost of finding and fixing requirements defects in the coding phase is 10 times more expensive than finding and fixing them during the requirements phase. Finding defects during Development Test, 15-40 times more, Acceptance Test, 30-70 times more and so-forth. In the context of the lifecycle phases depicted in Figure 1, verification and validation activities occur during development and acceptance testing as well as during initial operations.

Verification and validation are major cost and schedule drivers for any project, often as much as 50 percent of the total delivery cost for a new system [Hooks 2001]. Early identification of defective requirements prevents re-work and results in significant cost savings. Conversely, allowing these defects to go undetected until verification and validation will result in significant cost and schedule risk.

Considering your verification and validation approach early provides a foundation for estimating your cost and schedule for these activities.

Resist the temptation to reduce your requirement verification or validation activities due to cost or schedule overruns. As shown in Figure 1, to do so could result in much larger costs and schedule slips due to rework if requirement defects are not discovered until system validation or operations.

Failing to validate requirements before design can result in requirement verification and validation activities being performed together, during what was supposed to be just a verification activity. What happens if you verify that your SOI meets its requirements, but the requirement set is defective with ambiguous, missing, or incorrect requirements? Verification and design validation may not expose these defects. Because the set of requirements is defective, they don't correctly and completely communicate stakeholder expectations resulting in your SOI failing to pass system validation. If your SOI fails system validation, it will not meet the stakeholder expectations and will not accomplish its intended purpose. You will not have delivered a winning product.

In the commercial sector, warranty costs are a major concern. Warranty costs as a function of sales can eat into a company's profits. Allowing defective requirements in your requirement set and failing to do adequate verification, design validation, and system validation prior to product launch increases the risk of problems not being uncovered until after the product is provided to customers and released in to the marketplace. A high defect rate results not only in high warranty costs and reduced profitability, but also can impact a company's reputation resulting in reduced sales.

The risks of cost overruns and schedule slips associated with failing to develop defect free requirements and failing to plan ahead for verification and validation activities are mitigated by adopting good requirement development practices and planning ahead for verification and validation activities during the scope definition and requirement development and management lifecycle phases.

The importance of thinking ahead to verification and validation

Unfortunately, there are many examples of a product being delivered that met the "customer" requirements, yet did not meet stakeholder expectations and were either not used or required significant changes to be useful. What went wrong?

Addressing Verification and Validation Activities during Scope Definition Phase

One area to look at is in the scope definition phase of the project. How well has the scope of your project been defined? Does it clearly reflect stakeholder expectations? In the final analysis, system validation is based on making sure the product addresses the stakeholder expectations defined during the scope definition phase , agreed-to by the relevant stakeholders, and baselined in the system Scope Review (SR) or Mission Concept Review (MCR) before the requirements are written.

The scope of a project constitutes the vision: the "Need" to develop or procure a product or service; the goals and objectives of the stakeholders; information about the stakeholders, especially customers and users of the product or service; and how the product will be developed or purchased, tested, verified, validated, deployed, used, maintained, and disposed of. The scope also includes the identification of product boundaries and constraints.

Thinking ahead to verification and validation begins during the scope definition phase. During this phase, in addition to defining stakeholder expectations, the project is developing a draft of the Program/Project Management Plan (PMP), Systems Engineering Management Plan (SEMP), and the Master Integration, Verification, and Validation Plan (MIVVP). Schedules and budgets are being formulated and the Work Breakdown Structure (WBS) is being defined. Because verification and validation activities require a significant amount of the project's resources, planning for these activities must begin during the scope definition phase.

Special Scope Definition Phase considerations:

- *Define Need, goals, and objectives (NGOs)*

The "Need" for a project defines the "why" – why are we doing this? What are we trying to accomplish? The "Need" is based on an analysis of a problem that the project is supposed to solve for some stakeholder or group of stakeholders. Goals are those things that you plan to accomplish that will result in meeting the "Need" including Measures of Effectiveness (MOEs) - the things you plan to use to measure success. Objectives are Measures of Performance (MOPs), including Key Performance Parameters (KPPs) that show that you have met the goals. Objectives show that you have "gotten there", i.e., your product accomplishes what was expected of it by the stakeholders and can fulfill its intended purpose.

Requirement validation addresses how well the requirements communicate the stakeholder expectations and system validation activities ultimately address how well the designed and verified product has met the project's NGOs. The project team cannot claim they delivered a winning product if the Need, goals, and objectives were not met. (For more information on defining NGOs and other scope definition best practices, see references 1, 2, 13, 14, and 15.)

- *Indentify and involve relevant stakeholders*

Stakeholders include key representatives from various organizations and groups that have a "stake" or "interest" in your project. From a verification and validation perspective, the stakeholders that will have anything to do with verification and validation activities need to be identified and involved from the beginning of your project. Key stakeholders that will be "signing off" and accepting the product must agree that the planned verification and validation activities will result in sufficient "proof" such that they will approve and accept the product.

- *Identify drivers and constraints*

Drivers and constraints are those things that are imposed from outside your project that you have little control over, but are required to show compliance. Drivers and constraints represent a major source of requirements. These requirements must all be verified and the validation activities accomplished to show that the system is compliant with the identified drivers and constraints.

You need to make sure you include appropriate industry standards and any other standards mandated by government regulatory agencies as part of the project drivers and constraints. For systems targeted for use in the United States, chemical containment and emissions requirements from the Environmental Protection Agency (EPA), operator protection requirements from the Occupational Safety and Health Administration (OSHA), clinical trials requirements from the food and Drug Administration (FDA), materials flammability requirements from the Federal Aviation Administration (FAA) or National Aeronautics and Space Administration (NASA), or accessibility requirements mandated by the Americans with Disabilities Act (ADA) are a sampling of the type of outside drivers that must be reflected in the requirement set for your system.

If you are developing systems for use internationally, you need to find and incorporate the relevant standards and regulations in your requirements. Your verification and validation planning must include producing the data required by the regulatory agencies to show compliance with their requirements.

What if you are developing a medical product? What verification and validation requirements will the government and medical insurers place on your product? What extra reviews by outside experts and what liability insurance will be required before you can conduct tests involving human and animal subjects?

Other products may require outside experts for certification. Does your product require an Underwriters' Laboratories (UL) certification or maybe you have a pressure vessel that requires an American Society of Mechanical Engineers (ASME) Class 1 Certification?

Even if not explicitly stated by the customer or other stakeholders, be aware of and research the relevant outside drivers. Compliance is expected and failure to address relevant drivers and constraints can result in failure during system validation and product rejection by the customer *even if the customer did not explicitly communicate the drivers and constraints to you*.

- *Assess the technology maturity of the system to be developed*

A key driver and constraint is technology. To meet your objectives, especially high priority MOPs and KPPs, an assessment of the current maturity level of all the technology development needs for the project is necessary. Out of this process assign a Technology Readiness Level (TRL) for your system and develop a Technology Maturation Plan. All requirements tied to the MOPs and KPPs are high priority. If the TRL level is low, they are also high-risk requirements. Planning for verification and system validation of high priority and high-risk requirements is extremely important. The associated activities and resulting resource needs can have big impacts on your project's cost and schedule. Include in your planning a plan for maturation of key technologies requirement to meet your NGOs. This is necessary as a major risk mitigation activity to protect against cost and schedule overruns. (For more information on TRLs and the development of technology driven products see reference 16.)

- *Define feasible scenarios and concepts to meet the stakeholder expectations*

A major activity of scope definition is the development of a set of scenarios and concepts that address the expectations of all the stakeholders, each of the product's lifecycles, for both nominal and off-nominal conditions. Scenarios and concepts for verification and validation need to be defined and documented so you have the information needed to develop your WBS, budget, and schedule. A major reason for a project's cost overruns and scheduling delays can be contributed to the failure of the project to adequately plan ahead for verification and validation activities.

When developing scenarios that address verification and validation activities, pay particular attention to other systems that "enable" you to accomplish your verification and validation activities. You may need unique support or test equipment, special facilities, or models to perform your verification and system validation activities. These systems are referred to as enabling systems. These are products that need to be identified, defined, designed, modified, built, procured, verified, and validated in time to meet your system's verification and validation schedule. Will these systems be ready to support your verification and system validation activities when you need them? Early verification and validation assessments during the scope definition phase will help you identify these enabling systems as well as additional project requirements needed to support verification and validation activities and allow you to include them in your project budget and schedule.

Your system may need specific features dedicated to support your verification and validation activities especially when using test or demonstration as methods for verification and system validation. Additional interfaces may be needed to give you access to data that is needed for verification and validation, but not normal operations. This could result in the need for additional connectors and wiring harnesses to connect to special test instrumentation or external power, extra data to display or record, or a database to give you visibility into an internal process, an inspection portal, or a bracket to hold a part in a test fixture. [Hooks 2001]

Requirements that are verified by analysis using models can result in additional development effort to produce those models. Model types include physical, graphical, mathematical, and statistical [Larsen 2009]. In some cases, the validation of models can be very expensive and take up a large portion of the projects resources. To validate the model you may have to develop a simulation of your system. This simulation is used to run tests to collect data needed to validate the model before the model can be used to verify your system's requirements or for system validation.

What is the needed fidelity of the system you plan to use for verification and system validation? For your system, you need to assess which requirements can be verified using an engineering test model, qualification unit, delivery-like hardware, or actual delivery hardware. What fidelity of equipment is needed during development? Qualification? Acceptance? Operations? Each of these types of equipment or versions of your product needs to be included in your budget and schedule.

These are examples of features or additional systems that are needed to make verification and validation possible in some cases and reduce costs in other cases. Addressing the need for equipment and facilities needed to perform verification and validation early avoids possible delays in actual product delivery. Like models discussed above, you must verify and validate this equipment prior to your use during your system's verification and validation activities. Planning for verification and validation early permits the execution of concurrent activities that will allow

you to complete your verification and system validation per your schedule and avoid delivery schedule delays and resulting cost overruns.

- *Define product boundaries and external interfaces*

An interface is a boundary across which systems interact. Serious problems can, and all too often do, arise at the interfaces. Your project is particularly vulnerable when interfacing with systems over which you have no control. Because of this, your product is at most risk at the interfaces. Identifying interfaces facilitates definition of your system's boundaries and clarifies the dependencies your system has on other systems and dependencies other systems have on your system. Indentifying interfaces helps you ensure compatibility between your system and those with which it interacts. Identifying interfaces also helps to expose potential project risks.

Because of this, it is extremely important that you define your concepts for how you will verify all interface requirements and validate that your system will work with all the external systems with which it must interact in the intended operational environment.

In your planning, be sure to address both internal and external interfaces. An emulator or simulator may be needed to complete your verification and validation activities. As discussed earlier, these must be either developed or procured and verified and validated prior to your use during your system's integration, verification, and system validation activities.

Addressing Verification during the Requirement Definition Phase

Identification of the proposed verification method is an essential attribute of a well-written requirement. Assessment of verification as you develop your requirements improves requirement quality, ensures your requirements support verification, provides the basis for estimating verification cost and schedule, and can help reduce the cost of these activities. Careful assessment of the requirements that are most costly to verify can often result in modification to the associated verification activities to reduce the verification cost while still meeting your system's Need, goals and objectives.

During the scope definition phase discussed above, I discussed verification and validation from the 10,000 foot level. Basic concepts and scenarios for completing your verification and validation activities were defined. This information was used to develop a high level budget and schedule for these activities in your PMP, define your high level processes in the SEMP, and provided a detailed process definition in the MIVVP.

However, the devil is in the details. As requirements are developed, the verification method that will be used to show the system meets the requirements must be addressed. Let's look at the characteristics of good requirements from a verification perspective.

- *Requirement is needed*

A mandatory characteristic of a requirement is that it is needed. If a requirement is not needed, why is it in the requirement set? By its very existence, a requirement has a dollar value associated with it. "Each requirement carries a cost. It is therefore essential that a complete but minimum set of requirements be established from defined stakeholder requirements early in the project life cycle. Changes in requirements later in the development cycle can have a significant cost impact on the project, possibly resulting in cancellation." [INCOSE 2010].

All requirements need to be maintained, managed, and verified. Requirements that are not necessary result in increased management cost of the requirements that, in turn, increases overall project costs and leaves fewer resources for needed requirements. Un-needed requirements result in work being performed that is not necessary, taking resources away from the implementation of those requirements that are needed. In addition, implementing requirements that are not necessary can result in degraded system performance as well as introducing a potential source of failure and conflict.

To test for requirement need, ask:

1) Why is the requirement needed? Why is it in the requirement set? If there is no rationale, or a weak rationale for its existence, delete it.
2) What would happen if we deleted this requirement – would the developer address this concern anyway? If nothing would happen or the developer would have to address this concern anyway to meet other requirements, then why is it in the set? Delete it.

- *Requirement is verifiable*

Another mandatory characteristic of a requirement is that it must be verifiable. Why ask for something if you can't prove it has been implemented as intended? A requirement that is not verifiable may result in the implementation of an incorrect solution or the system may be verified as doing something other than was intended. If the true intent of the requirement is not clear, stakeholder expectations may not be met.

To test whether a requirement can be verified, ask:

1) How can this requirement be verified? If no method can be identified, can it be re-written so that it can be verified?
2) Do we want to verify this requirement? If not, delete it.
3) Is this what we really want to verify? If not, change it to what you do want to verify or delete it.

- *Requirement is attainable*

A third mandatory characteristic of a requirement is that it must be attainable given the existing budget, schedule, and level of technical maturity. A requirement that is not attainable will fail verification. If you know you will not be able to meet the requirement and thus know you will fail to verify that the system meets the requirement, why state it? Stating a requirement that is not attainable will result in wasted effort, cost and schedule impacts, as well as unmet stakeholder expectations (performance).

- *Requirement is written correctly*

If a requirement is poorly written, it will most likely not be verifiable. If it contains ambiguous terms, it can be understood in more than one way and is thus not verifiable. If it is not clear, concise, or uses poor grammar and you are leaving room for multiple interpretations, it is not verifiable. If it is stated negatively, it may not be verifiable. (It is difficult to prove that a system never, ever does something.) If the requirement contains more than one thought, which thought do you verify? This is especially problematic if each thought needs a different method of verification.

To guard against poorly written requirements make sure your team is trained to write defect free requirements and define your requirement development process as one based upon the use of the

"Rules of Writing Good Requirements." (See reference 2, as well as Appendix C of the *NASA Systems Engineering Handbook*, "How to Write Good Requirements". [NASA 2007].)

- *Requirements are documented at the level where they will be verified.*

Proper allocation of requirements to system architectural levels is very important to effective verification and validation. All requirements must be documented at the level where they will be implemented and verified. The System Engineering "Vee" Diagram in Figure 2 illustrates the overall Systems Engineering process, and the relationship among the integration, verification, and validation processes that result in system acceptance.

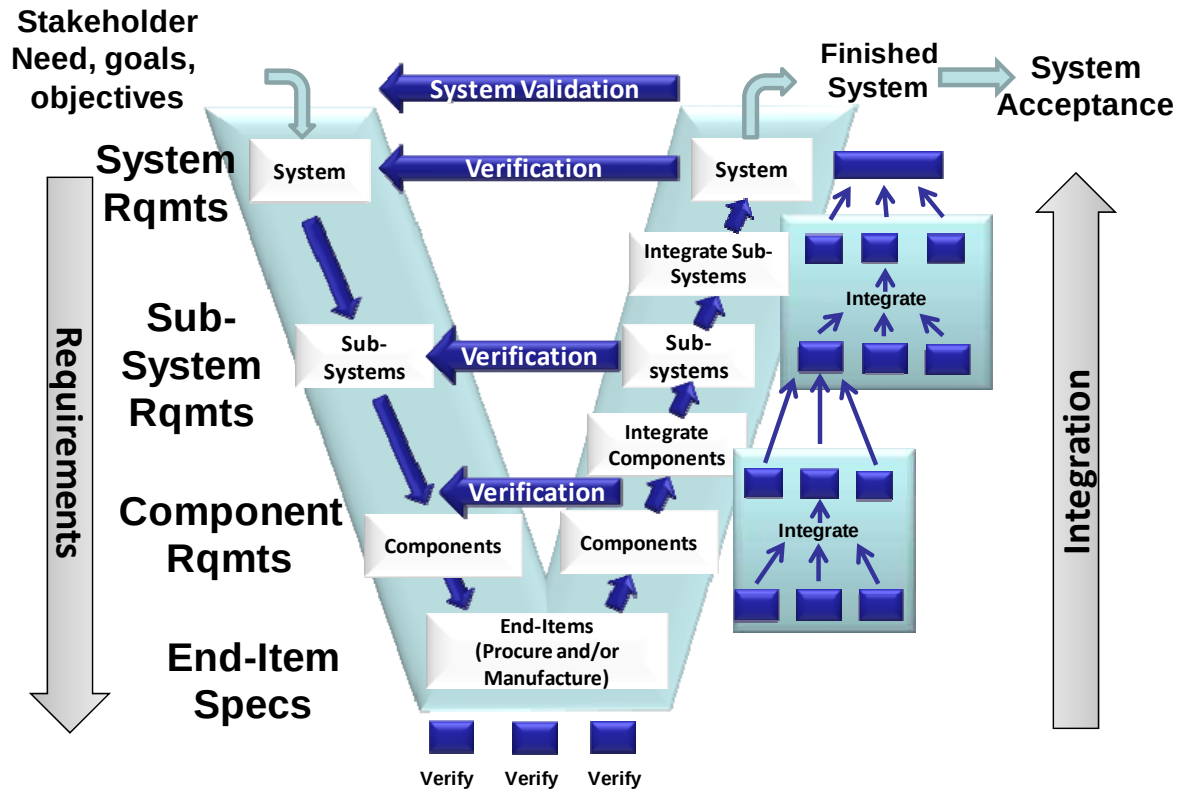

Figure 2: Systems Engineering "Vee" Model

Requirement definition and system design activities descend down the left side of the Vee. System integration, verification, and validation ascend the right side of the Vee. Requirement definition and design is a top-down activity during which requirements for the system as a whole are defined and then decomposed to sub-systems while flowing down (allocating) requirements to each subsequent level of the architecture.

Once the system has been decomposed into the various end-items that make up the system, integration, verification and validation activities begin. During integration, lower level end-items are integrated together to form the subsystems. Requirements are verified and validated and compatibility is assessed to ensure the lower level entities will successfully combine at defined interfaces. Successfully combined subsystems result in the system required at the next higher level as depicted on the Vee shown in Figure 2. Once the entire system has been verified, the system is ready for validation and acceptance by the customer. In some cases, final system validation and system acceptance activities are combined.

Specifying a requirement at a level higher than where it will be verified, may make it difficult or impossible to verify at that level. When writing requirements, always ask: "Does this requirement apply to the level you are writing requirements for?" The requirement may be valid, but not for that level. If the requirement is not being stated at the correct level, move it to the correct level.

- *Requirements are allocated (flowed down) to the next level of the system architecture*

The flow down of requirements from one level to the next level of the architecture is an important activity that needs to be done correctly. This flow down activity is called allocation. Allocation is the process of apportioning resources or assigning responsibility for implementation of requirements from an upper level in the hierarchy to parts at the next level of the system architecture. All requirements must be allocated until the final level of the architecture is defined.

Failing to allocate your requirements can result in requirements not being implemented at the next level of the architecture. Failing to allocate requirements can also result in missing lower level requirements (derived children requirements that are necessary and sufficient to meet the parent requirement.)

- *Requirements address internal interfaces*

A key part of the allocation process is the identification and definition of internal interfaces among end-items within your system as well as external interfaces between your system and the outside world. When functional requirements are allocated to more than one part of the architecture at the next level, there may be an interface. If so, the interface must be identified and defined. The respective requirements document for each part then will include their respective interface requirements. Interface requirements require special attention to ensure that they are written so as to permit verification. (The inclusion of a reference to where the interface is defined is a common attribute of all interface requirements. This definition is needed so that the interface requirement can be verified. The interface definitions can be contained in an Interface Agreement Document (IAD) or Interface Control Document (ICD).) (For more information on identifying and defining interfaces and writing and documenting interface requirements, see reference 17.)

- *Requirements are traceable to a parent*

Traceability is the concept that all requirements need to be linked (traced) to their source which is their parent. Tracing to a parent is one way to determine if a requirement is needed. When there is no trace to a parent, it could mean that the requirement is not needed and there is gold plating. "Gold plating" is the act of adding features to a system that are not needed. Gold plating is a major source of requirements creep and can impact the cost and schedule of your project. This is especially true when it comes to verification and validation activities.

Once requirements have been allocated to a part at the next level of the architecture, child requirements are derived such that when implemented results in implementation of the parent requirement. Access whether or not the children requirements linked (traced) to the parent requirement are necessary and sufficient to meet the intent of the parent to determine whether or not a parent requirement is properly implemented. Without proper allocation and traceability documentation, this assessment cannot be done.

Understanding and then correct implementation and documentation of the concepts of allocation and traceability are the keys to a successful verification and validation process. As shown in Figure 2, Systems Engineering Vee model, integration is a bottoms-up process. When integrating

at one level, you need to make sure verification and validation are complete at the previous level. Successful verification at the lower level is a prerequisite to successful verification at the next level. This is especially true when it comes to interfaces. *When planning verification at one level, always include a step in your process to look at the verification documentation at the previous level to make sure it has been successfully completed before you do verification at your current level.*

Planning your verification activities

This section provides more of the "how" associated with planning for verification by introducing some of the tools and activities that will help you plan for and accomplish verification and validation. Two excellent sources that go into much more detail can be found in references 3 & 4.

Thinking about verification while developing your requirements

As you write your requirements, address the characteristics of good requirements described in the previous section: requirements are: needed, verifiable, attainable, correctly written, at the correct level, allocated, and traced. Also ensure that all internal interfaces addressed.

Addressing verification as the requirements are written ensures a better set of requirements. The following questions need to be addressed as each requirement is written:

1) "What will be the primary method used to verify this requirement?"

   The primary methods of verification include test, demonstration, inspection, and analysis. (Note: Some organizations further define the various methods of analysis to include analysis by similarity and by model.) "Each requirement should be verifiable by a single method. A requirement requiring multiple methods to verify should be broken into multiple requirements."[INCOSE 2010]. Some people will state more than one verification method, e.g., test and analysis. This is not needed as it is understood that test includes some analysis and analysis may include some testing. The best practice is to state a single primary method for verification.

   Risk is a major consideration when choosing the verification method. This is a very important issue from a project management standpoint. Verification by test provides the most confidence that the system meets a requirement. We would like use test as our primary method for all requirements, but we may not be able to. In some cases test is not appropriate. In other cases in may not be possible until the system is in its real operational environment. In many cases we can't afford the cost or time to do a full-up test for all our requirements. In other cases a special facility to simulate your operational environment or equipment to test your interactions with other subsystems or systems may not be available.

   All of these considerations need to be addressed when deciding on the verification method. Therefore, you must do a risk assessment and determine which requirements represent the highest risk to your project if not properly implemented. Often the requirements that pose the highest risk to project success are requirements that trace to the high priority objectives defined during scope definition as well as the safety and mission assurance requirements. When a requirement poses a lower risk to the project, one of the other verification methods may be acceptable. In my experience analysis is the most overused method and posses the highest risk when good rationale for using analysis isn't defined. Reducing verification activities due to cost overruns or schedule slips can add significant risk to your project.

2) "What activities need to be performed as part of the verification or validation?"

   Develop a scenario or operational concept to determine which activities need to be performed as part of the chosen verification or validation method being sure to address where and by whom. This process will uncover the interface, facility, support equipment, and instrumentation issues associated with verification and validation. Where will the verification or validation activities be performed? Will the system under verification or validation require different power or cooling during a test or demonstration than needed during normal operations? Will it need simulated data streams? How will you address the interfaces – is an emulator or simulator needed? What external command and control capability is needed? What data is needed to control the activity? What data must be recorded and how fast must the system output this data? Who will be involved in the verification or validation activity? Who must observe the test, perform the analysis or demonstration, or inspect the system? What training or certifications do the personnel need to have to participate in their assigned role? What resources are needed to complete this activity? How long will it take? How much will it cost?

   The verification (aka "test") engineers will ask these questions. Project managers need to know the resources needed, the cost, and schedule needs to accomplish the verification or validation activity to make sure they have sufficient funds and time accounted for in the project budget and schedule. It is best to address these questions both as you are defining your project scope and as you are developing your requirements before you unknowingly spend project resources on designing or building your system based on an unverifiable requirement or performing an activity that cannot be accomplished within the project's budget and schedule.

   This is a knowledge-based process that evolves with the design of your system. Some of these questions may not be able to be answered before system design. Requirement development and management is an ongoing task. Even if you cannot answer these questions before some design effort, you can certainly do so during or after design but before manufacture, code, or build.

3) "When and at what level will your verification and validation activities be performed?"

   As part of your verification planning and concept development efforts, consider timing and develop a "when" strategy. Remember integration, verification, and validation are bottoms-up process activities.

   Balance component, subsystem, and system level verification activities. You may be tempted to save verification time and money by doing only subsystem or system level verification. This strategy is very risky; you may not find critical problems until very late in the product integration process or during system validation or customer acceptance. It may be hard to trace the problem to its source in a complex system. Once you find the source of the problem, it will be difficult to resolve, especially if more than one component is involved and you need to de-integrate the system as part of the anomaly resolution. A single problem emerging during system-level verification or validation may cancel all savings projected from eliminating component or subsystem verification activities.

   Alternatively, budget and schedule limits may make the verification of every component-level requirement cost-prohibitive and unnecessary. Again, it is a matter of risk mitigation. If the component level requirement traces to higher-level requirements dealing with high priority

objectives or mission or safety, you will want to verify those requirements. Alternatively, you may be able to accept the risk and wait to do verification at a higher level of the architecture - if possible. Another approach is to use a less costly and time consuming method of verification for the lower risk component requirements, e.g., "analysis" or "demonstration" rather than "test".

4) "What do I need to do to verify the system in the context of the next level "super" system in which my system is a part of in its intended operational environment?"

   Failing to verify the integrated system or failing to verify the system with its external interfaces and in its operational environment can result in major embarrassment when the delivered system fails to perform as expected and doesn't accomplish its intended purpose (validation). Even though every part or component has been verified, you need to ensure they work together as a whole, they work with their external interfaces, and they work in the intended operational environment before system validation activities and before you deliver the system to your customer. Verifying internal and external interfaces often requires test as the method of verification. The cost of this testing can be high, but failure to successfully complete this testing can lead to major, and even catastrophic, problems.

5) "What constitutes proof that the requirement has been verified?"

   Proof is derived from documentation and establishment of confidence. Document the evidence that the system meets this requirement as well as the level of confidence needed before you and others are willing to "sign off" on the verification documentation for the requirement. Consider also what evidence is needed before the customer will accept the product for delivery.

Failing to address these questions early can result in unpleasant surprises late your system development lifecycle putting your project at risk.

Addressing verification, validation, certification, acceptance in Supplier Agreements

If an end-item is to be outsourced to a supplier, the requirements for each end-item are typically documented in a Statement of Work (SOW) and a Technical Data Package (TDP). The SOW contains process and workmanship requirements levied on the supplier on the tasks they need to perform as part of the contract to fabricate, test, transport, and install a specific end-item. The TDP includes equipment lists, parts lists, drawings, and end-item specification requirements on the system to be supplied that can be thought of as the "build-to" requirements. In some cases, the end-item requirements may be contained directly in the SOW. All of these requirements need to be verified and the method of verification defined as part of the supplier agreement process. The system requirements in the SOW/TDP need to be traced to the technical requirements in your System Requirement Document or Specification (SRD/SRS).

Together, the SOW/TDP results in a supplier agreement which is a contract between the customer and supplier. This contract needs to clearly state your expectations for the verification and system validation approach and scope of the effort. If the system is complex, testing every possible case or path may be cost-prohibitive. What are the risks and where do you want to spend funds for verification and system validation? Missing, vague, or open-ended verification and validation plans can overrun the supplier's budget or leave you, the customer, without confidence in the

system. Addressing verification and system validation in supplier agreements protects all parties in the contract.

The documentation that provides proof that these activities have been successfully completed is used as part of your overall verification and validation activities. You can accept the supplier's documentation as proof, or for critical items, you may want to perform your own verification and validation. In some texts, accepting a supplier's documentation as proof the delivered component or system meets its requirements is classified as "verification by certification".

If you are a supplier and your customers will be performing their own acceptance activities, assess your requirements against their acceptance plan. Are you missing requirements that are obvious to the customer? If you are the customer, your acceptance plan must be comprehensive enough to satisfy your organization that the system will perform as advertised or as described in your supplier agreement/contract. Ensure your acceptance plan reflects the actual requirements and does not introduce previously unstated requirements. [Hooks 2001]

Tools to help with your verification and validation planning

This section addresses some of the key tools that are used to help with verification and validation planning.

- Program or Project Management Plan (PMP): Defines the overall project activities, critical milestones and events, and process identification. Organization and Work Breakdown Structure (WBS) as well as a preliminary budget and schedule are defined. This information needs to include your high-level verification and validation activities.
- Systems Engineering Management Plan (SEMP): Expands on the PMP from a Systems Engineering perspective. The SEMP establishes the overall plan for the technical development of your system. The ultimate objective of the SEMP is to provide a disciplined framework to meet cost, technical performance, quality, and schedule objectives for the project or program. It is important that the SEMP establish the overall verification and validation philosophy for the project or program.
- Master Integration Verification and Validation Plan (MIVVP): Expands and implements the verification and validation philosophy of the project or program as defined in the PMP and SEMP. The MIVVP defines the detailed processes to be used for verification and validation. The information in the MIVVP is used to manage and plan system verification and validation activities. The MIVVP further defines the integration, verification, and validation schedule contained in the PMP and SEMP.
- System Requirements Document or Specification (SRD/SRS): Contains the system's requirements. The proposed verification method is a key attribute of a each requirement, and needs to be captured and documented with the requirement. You can also include the level and lifecycle phase for verification activities. This information will be used to create the Requirements Verification Matrix.
- Requirements Verification Matrix (RVM): A matrix that lists each requirement, requirement rationale, verification method (test, inspection, demonstration, analysis), responsible organization, level (component, subsystem, system), phase (design, manufacture, integration), and often includes a short description of the success criteria for successful verification. The RVM is often included as part of the SRD/SRS.

- System Integration, Verification and Validation Plan (SIVVP): Implements the project's MIVVP processes for a given component, subsystem, or system's requirements as documented in the SRD/SRS and associated RVM. The SIVVP contains a detailed identification of the tasks to be performed as part of verification and validation of that system. Resources including test equipment, facilities, and personnel are addressed. A detailed schedule consistent with the Integration, Verification, and Validation Schedule defined in the MIVVP is included.

  Products to help develop and implement this plan are discussed below.

- Verification Requirement Definition Sheet (VRDS):, For each requirement in the SRD/SRS/RVM, the VRDS transforms the RVM "what's" into "hows" –"How do you plan on verifying the requirements listed in, and meet the success criteria defined in, the VRM?" The VRDSs document the details of the specific verification activities identified in the RVM. An example of and instructions for completing a VRDS is included in Appendix A. The VRDS captures all of the verification requirements and other supporting information for the specific requirement verification. For each System, the set of VRDSs document the verification requirements from the perspective of the verification activities that are performed to ensure the system meets its requirements. The set of VRDS are contained in a Verification Requirement Document (VRD).

  (Note: Some projects document verification requirements in a separate section of the SRD/SRS. I do not advocate this approach. I advocate a separate document, Verification Requirements Document (VRD) where the verification requirements are documented via the VRDSs defined here in. The concept of a separate VRD containing VRDSs for each requirement was used by NASA's Constellation Program Ares 1X project. See reference 11.)

- Task Definition Sheets (TDS): As an aid to verification activity planning, each system lead develops a list of tasks to be performed to accomplish the needed verification activities defined in the VRDS. For each of those tasks, a Task Definition Sheet (TDS) is developed. The TDSs contain the information needed to develop an integrated schedule for completing all the defined tasks involved in verification, validation, acceptance, and readiness. One TDS can address multiple, related VRDSs. Which VRDSs will be listed in the task objectives section of the TDS. There should be one TDS for each verification or validation task identified in the SIVVP. An example of and instructions for completing a TDS is included in Appendix B.

- Task Performance Sheet) (TPS): The TPS or other approved Work Authorization Document (WAD) contains the actual procedure used to perform the verification or validation activities defined in the TDS. This is the document actually used to accomplish the verification or validation activity and document the results. The TPS is signed off by the appropriate personnel who are identified in the applicable TDS(s). The completed TPS contains the data that represents the proof that a requirement or set of requirements have been verified as defined in the VRDSs or validated as defined in the MIVVP and SIVVP.

  A system will have multiple TPSs. There may be one TPS for verification by inspection while an end-item is at the vendor facility. Completion of that TPS is then one of the items required prior to end-item acceptance and shipment to the customer. Verification by analysis may be performed via a single TPS – a separate TPS for each verification by

analysis. Verification or validation by test may group like requirements together in the same TPS, i.e., leak tests for multiple joints/welds.

TPSs call out checklists and procedures and contain specific actions to be performed. From a verification or validation activity standpoint, TPSs are prepared based on the information contained in the SIVVP, VRDSs, and TDS that the TPS is being prepared to satisfy. All TPS line items involving a verification of a specific requirement or validation activity must include a mandatory inspection point (MIP) that includes a Quality Assurance/Quality Engineer (QA/QE) stamp.

Wrap up and Parting Thoughts

Verification and Validation are a large part of system development. Second only to rework, a project's biggest costs are involved in verification and validation. Failing to address verification and system validation early can result in unpleasant surprises towards the end of the product lifecycle when verification and system validation activities begin, putting your project at high risk of cost overruns and schedule slips.

The emphasis of this paper was to present best practices and tools and proven methods project teams can use to address planning for verification and validation from the beginning of a project and avoid the risks of not doing so.

The case was made concerning the importance of planning for verification and validation from a risk mitigation standpoint and the importance of project management ensuring that requirement development and management process address verification and validation activity risks, cost, and schedule.

In conclusion, avoid the risks associated failing to adequately plan for verification and validation throughout the project lifecycle by addressing the following best practices:

- During scope definition, involve those personnel associated with verification and system validation with the development of a set of scenarios or operational concepts used to successfully complete needed verification and system validation activities. This knowledge is documented in the PMP, SEMP, and MIVVP.
- Address verification and validation early as a risk mitigation activity from a project management perspective.
- Ensuring that you begin with verifiable requirements is the key to reducing risk later in your system development lifecycle. While developing requirements, always think ahead to verification. This improves each requirement's quality and reduces the risk of unpleasant surprises later in the system development lifecycle.
- Use the tools discussed in the paper (PIVVP, RVM, VRDS, TDS, and TPS) to facilitate planning for and completing verification and validation activities.
- Ensure project management is involved in key decisions associated with balancing the risks against cost and schedule when selecting verification methods and then determining at what level and phase requirements are verified.

Addressing verification and validation early is key to delivering a winning product – the first time!!!

References and Further Reading

1. Hooks, I. F. and Farry, K. A., *Customer-Centered Products: creating successful products through smart requirements management*; AMACOM Books, NY, NY, 2001.
2. Requirements Experts, *Requirements Development and Management,* Seminar Workbook. 2012.
3. Larson, Kirkpatrick, Sellers, Thomas, and Verma, *Applied Space Systems Engineering*, McGraw-Hill, 2009.
4. Engel, A., Verification, *Validation, and Testing of Engineered Systems*, Wiley, 2010
5. INCOSE, Systems Engineering Handbook - a guide for system life cycle processes and activities, Version 3.2, INCOSE-TP-2003-002-03.1, January 2010, ed, Cecilia Haskins.
6. ISO/IEC 15288, System Engineering-System Life Cycle Processes, October 2002.
7. NASA, *System Engineering Handbook*, SP-2007-6105, Rev. 1, December 2007. <http://education.ksc.nasa.gov/esmdspacegrant/Documents/NASA%20SP-2007-6105%20Rev%201%20Final%2031Dec2007.pdf>.
8. NASA, *Systems Engineering Processes and Requirements*, NPR 7123.1A, March 2007 <http://nodis3.gsfc.nasa.gov/displayDir.cfm?Internal_ID=N_PR_7120_005D>.
9. NASA, *Space Flight Program and Project Management Requirements*, NPR 7120.5D, March 2007. <http://nodis3.gsfc.nasa.gov/displayDir.cfm?Internal_ID=N_PR_7120_005D>.
10. NASA, *Space Flight Program and Project Management Handbook*, NPR 7120.5, February 2010. < http://www.nasa.gov/pdf/423715main_NPR_7120-5_HB_FINAL-02-25-10.pdf >.
11. NASA, *Ares 1-X System Verification Requirements Document (VRD0 for Ares 1-x Flight Test Vehicle (FTV),* AI1-SYS-VRD, version 2.02, December 9, 2008
12. Software Engineering Institute, *CMMI for Development – Improving processes for better products*, Version 1.3, CMMI Product Team, Carnegie Mellon, August 2006.
13. Wheatcraft, L. S., and Hooks, I. F., *Scope Magic*, 2001. <http://www.complianceautomation.com>.
14. Wheatcraft, L. S., *The Importance Of Scope Definition Prior to Developing Space System Requirements*. INCOSE *INSIGHT*, Vol. 4 Issue 4, January 2002. <http://www.complianceautomation.com>.
15. Wheatcraft, L. S., *Delivering Quality Products That Meet Customer Expectations*. Published in CrossTalk, The Journal of Defense Software Engineering, January 2003, Vol. 16 No. 1. <http://www.complianceautomation.com>.
16. Wheatcraft, L. S., *Developing Requirements for Technology-Driven Products*. Presented at INCOSE 2005, July 2005. <http://www.complianceautomation.com>.
17. Wheatcraft, L. S., *Everything You Wanted to Know About Interfaces – But Were Afraid to Ask,* Presented at INCOSE, July 2010.
18. Wheatcraft, L. S., *Triple Your Chances of Project Success - Risk and Requirements.* Presented at INCOSE 2011, June 2011 and NASA PM Challenge 2012, February 2012.

BIOGRAPHY

Lou Wheatcraft has over 40 years experience in the aerospace industry, including 22 years in the United States Air Force. Over the last 12 years, Lou has worked for Compliance Automation (dba Requirements Experts), where he has conducted over 140 seminars on requirement development and management for NASA, Department of Defense (DoD), and industry. Lou has had articles published in the International Council of Systems Engineering (INCOSE) *INSIGHT* magazine and in DoD's magazine, *CrossTalk.*

Lou has made presentations at NASA's PM Challenge, INCOSE's International Symposium, and at the local Project Management Institute (PMI) and INCOSE Chapter Meetings. Lou has a BS degree in Electrical Engineering, an MA degree in Computer Information Systems, an MS degree in Environmental Management, and has completed the course work for an MS degree in Studies of the Future.

Lou is a member of INCOSE, co-chair of the INCOSE Requirements Working Group, a member of PMI, the Software Engineering Institute, the World Futures Society, and the National Honor Society of Pi Alpha Alpha. Lou is the recipient of NASA's Silver Snoopy Award and Public Service Medal and was nominated for the Rotary Stellar Award for his significant contributions to the Nation's Space Program.

If you have further questions, please feel free to contact the author at: louw@reqexperts.com

[Program or Project Name] [System Level Name] Verification Requirement Definition Sheet	[Project Logo]

[Requirement number] Requirement to be Verified	
Verification Requirement Number: VR [Rqmt Number]	**Parent Requirement(s):** [Rqmt Number]
Requirement Text:	

Verification Requirements
Verification method (VM): [ ] Test [ ] Demonstration [ ] Inspection [ ] Analysis
Description of verification activities to be performed: The [VM] shall consist of
Success Criterion: Verification shall be considered successful when the [VM] shows …..
Rationale: [Reason for the VM to be used]:

Verification Implementation
Lifecycle Phase: System: [ ] Manufacturing [ ] Build up **Integration**: [ ] Sys-Sys Interfaces [ ] System Acceptance
Applicable documents: TPS xxxxxxxxx**; Drawing: Num:** xxxxxxxxxxx **Rev:** x **Date:** xx/xx/xxxx **Completed VRDSs:**
Nonconformance history: N/A
Closure data/documentation required: Signed off TPS and data. Electronic/written confirmation that the requirement verification activity has been successfully completed. **Events preceding Verification Activity:** **Estimated Duration of Verification Activity: [TBS]**

Closure Of VR [Rqmt Number]			
Date Closed: mm/dd/yyyy		[ ] **Safety Critical (Y/N)**	
xxxx Quality Engineer:	**xxxx System PM:**	**xxxx Chief Engineer:**	**Customer System Engineer**
[Name]	[TBS]	[Name]	[Name]

Form: Verification Requirement Definition Sheet

Instructions for filling out a VRDS

This section provides the information needed to fill out the VRDS forms. These forms are used to specify the verification requirements as tailored for the verification of the System requirements.

- **VRDS Title**

The title contains the name of the System

- **Requirement to be Verified**

The numbering of each VRDS is comprised of the requirement number contained in System Spec with a VR- inserted at the beginning of the number.

Include the requirement text along with any notes or rationale.

- **Verification Method**
  - The method is selected from Demonstration, Analysis, Inspection, and Test as stated in the RVMs.
  - The Verification Method identifies the final process to obtain the measure of the success criterion. Supporting sub-methods may be called out (e.g., "Analysis and Test," in cases where the success criterion is a Monte Carlo simulation supported by input from Test results). (Generally, most if not all of the four methods will be inherent in any verification process, so it is preferred to focus on the final one that generates the measure of the success criterion.)
- **Verification Requirements**

The verification requirements are requirements levied on the organization responsible for performing the verification. Verification requirements address the activities that must be performed to obtain the information needed to verify the system requirement as well as a statement of the success criterion. These requirements will be implemented via a TPS or other form of Work Authorization Document (WAD).

  - **Description of verification activities to be performed**
    - One or more "shall" statements (requirements) stating the measure (a parameter, quantity, or condition) of the verification success criterion and how this measure is obtained.
    - This part states how to obtain (generally, not calling out a specific tool) values or states for this measure.
    - This part states required assumptions and conditions for the verification.
    - Necessary conditions, assumptions, and other required states or inputs for obtaining the measure of the success criterion are stated clearly in "shall" statements (requirements).
    - Any necessary constraints, certifications, or required training for verification execution personnel are identified in "shall" statements (requirements).
  - **Success Criterion**
    - Include a single "shall" statement (requirement) stating the success criterion. This success criterion should be the same as stated in the VRMs.
      - The success criterion includes the value or state against which the success measure obtained from the verification process is compared to indicate

successful verification (e.g.; measured weight shall be less than 133 pounds 13 ounces; or telemetry received in the control room).

- The success criterion needs to be measurable or observable by some process.
- A simple "yes/no" answer is all that is required to determine if the success criterion has been met.
- The measure for the success criterion is stated for the "as built" implementation of the system requirement.
- No special skills or subject matter expertise are required to make the decision whether the success criterion has been satisfied.

- **Rationale**

  The rationale is an "attribute" of the verification requirements. One rationale is written on the VRDS form for each System Requirement, but it applies to all of the verification requirements specified on the sheet. The rationale is especially important when a specific verification method "choice" is not the obvious or expected one. For example, a specific system requirement might be one that is universally considered to require rigorous verification testing. However it might be verified by a less stringent method (e.g. analysis or inspection) if the risks of not testing have been fully considered and accepted by the project.

  Rationale includes an explanation and justification for "why" the specific method, success criterion, measure, constraints, conditions, and assumptions have been selected.

- **Verification Implementation**

The Verification Implementation portion of the VRDS contains the following six types of information:

- **Lifecycle Phase**

  The Verification Phase states in which verification/integration lifecycle verification phase the requirement will be verified.

- **Applicable Documents**

  Any documents dealing with the verification of this requirement. These include the specific procedures (TPS or other WAD) used to do the verification in the case of a test or demonstration. These also include lower level children requirements that must be completed and signed off before the verification activities for the requirement can be performed.

- **Nonconformance history**

  Nonconformance history includes the history of previous activities when the ability of the System to meet this requirement was not able to be proven.

- **Closure data/Documentation Required**

  State what closure data and/or documentation is required proving that the system has meet this requirement.

- **Event(s) preceding Verification Activity**

  State any events that must be completed and indicate the system state or configuration that must be achieved prior to being able to perform verification. This includes

verification of children requirements, installation of the system in the facility, assembly, inspections, checkouts, etc.

- **Estimated Duration of the Verification Activity**

  Provide an estimate of the time needed to complete the activities associated with verification. In cases where one activity will involve the verification of multiple requirements, state the time to complete that activity.

- **Closure of the Verification Requirement**

  This section is completed once the verification activity has been completed and the closure data and requirement verification closure documentation have been completed. Enter the date closed (date the form is signed by all four parties) and indicate whether this verification includes a safety critical requirement.

  Each of the defined parties need to sign off the VRDS indicating their acceptance of the verification and their confirmation that system meets the stated requirement.

[Program or Project Name] [System Name] System Task Definition Sheet	[Project Logo]
Task (Inspection/Test/Checkout) Title:	
Task Number: Project [System 3 Ltr designation]-XXX	
Task Description: (Purpose of this task – why is it needed? Include a description of the equipment/ components/ assemblies/ subsystem/ systems involved in the task.)	
Task Objectives: (Expected outcomes as a result of doing this task.)	
Hazardous operation? [ ] Y [ ] N **Hazard Analysis Needed?** [ ] Y [ ] N	
Type of Task: [ ] Mechanical [ ] Integration [ ] Interface [ ] Power [ ] Command/Data [ ] Pressure/Leak [ ] Other:	
Task Methodology: [ ] Test [ ] Demonstration [ ] Inspection [ ] Analysis	
Lifecycle Phase : System: [ ] Manufacturing [ ] Build up **Integration**: [ ] Sys-Sys Interfaces [ ] System Acceptance	
Schedule: (Dates when task will be performed.) Start: End: Duration:	
Predecessor Task(s): (Tasks/installations that must be complete before this task can be performed.)	
Successor/Parent Task(s): (Tasks/installations that require this task be completed before they can be performed. Is this task part of a larger (parent) task? Which task?)	
Constraints: (Anything that limits the performance of the task. Also include impact/constraints on other activities during the performance of this task. e.g., no other work in the facility while this task is being performed.)	
Codes/Standards: (list any codes or standards this task must be compliant with:	
Resources Needed: (utilities (steam, power, LN2, air), other Systems, Portable Equipment, Leak Test Unit, etc. during the performance of this task.)	
Documentation: Checklists/Procedures: [ ] Existing [ ] To be Developed [ ] Use Vendor Procedure	
Organizations/Personnel: (who will be involved in the performance of this task? [ ] Project Engineer, [ ] Test Director, [ ] Supplier/Vendor, [ ] Safety, [ ] Quality, Technician, [ ] Other:	
Training/Certifications: (State any training/certifications required by personnel involved in the task.)	

Form: Task Definition Sheet

Instructions for filling out a TDS

This section provides the information needed to complete the TDS forms used to specify the top level information needed to develop an integrated verification, acceptance, and readiness schedule.

- **Task (Inspection/Test/Checkout) Title**: Include title that clearly communicates the nature of the task.
- **Task Number**: Include a unique identification number for the task. Sequentially number each TDS for your system.
- **Task Description:** Clearly state the purpose/reason for this task – why is it needed? Include a description of the equipment/ components/ assemblies/ subsystem/ systems involved in the task. What does the task consist of? (start the sentence with a active verb - test, checkout, inspect, etc.
- **Task Objectives**: List the expected outcomes from the performance of this task. If verification is an expected outcome, list the requirements that will be verified as a result of doing this task. (Note: Update any applicable VRDSs with the task number of this TDS to provide bi-directional traceability between the verification requirements and the task that will result in verification.)
- **Hazardous operation?** Check the appropriate block to indicate a hazardous operation and whether a hazard analysis is needed.
- **Type of Task**: Check the appropriate block to indicate the type of task (Mechanical, Integration, Interface, Power, Command/Data, Pressure/Leak. If none of these is appropriate, write in the type of task in the "Other" field.
- **Task Methodology**: Check the box that best indicates the overall methodology to be used to complete this task. (Test, Demonstration, Inspection, Analysis).
- **Lifecycle Phase:** Indicate the lifecycle stage where you plan to complete this task.
- **Schedule:** Indicate the dates when you plan to perform the task and expected overall duration (hours/days). These dates must be consistent with the other activities included in the Project's Master Schedule.
- **Predecessor Task(s)**: Indicate which prerequisite tasks must be completed before this task can be performed.
- **Successor/Parent Task(s)**: Indicate which tasks cannot be completed before this task is performed. Indicate whether or not this task part of a larger (parent) task? If so, indicate which task.
- **Constraints:** List any constraints on the performance of this task. Include anything that limits the performance of the task. Also include impacts/constraints on other activities during the performance of this task. e.g., no other work in test area while this task is being performed.
- **Codes/Standards**: List any codes or standards with which this task must be compliant.
- **Resources Needed**: List any resources needed to perform this task (utilities: steam, power, GN2, air) as well as any other Systems, Portable Equipment, Leak Test Unit, etc. that are needed during the performance of this task.
- **Documentation: Checklists/Procedures**: Indicate whether there is a requirement to develop new checklists and/or procedures before doing this task or whether existing checklists or

procedures can be used. Indicate whether you are planning to use any vendor procedures to complete this task.

- **Organizations/Personnel**: Indicate which key organizations need to be involved in the planning and performance of this task. (Project Engineer, Test Director, Supplier/Vendor, Safety, Quality, Technician, Test Operations, Others?
- **Training/Certifications**: State any training/certifications required by personnel involved in the task.