Thinking Ahead to System Verification and System Validation

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

Note: [Feb 2016] This is an update to this paper which was originally published in 2012. The updates are a result of a maturing of the author's thoughts on the subject and a clearer understanding of what is meant when using the terms "verification" and "validation". The major change is the addition of a more detailed definition of the terms and the addition of Figures 1 & 2. Most of the rest of the changes were to make the rest of the text consistent with the proposed definitions to make clear what the context is when using these terms.

Abstract Second only to re-work, a project's largest costs are involved in system verification and system validation. Failing to adequately plan for system verification and system 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 system verification and system validation during the scope definition activities; addressing the following questions:

- "Who will verify the system meets the requirements that drove the design, and where and when will these activities be done?"
- "Who will validate that the designed, built or coded, system will meet its intended purpose in its operational environment, and where and when will these activiies be done?"
- "What facilities, support equipment, simulators, emulators, or models are required for system verification and system validation?"
- "What are the interfaces involved during system verification and system validation?"

When you start writing your requirements, addressing system verification as the requirements are written insures a better set of requirements and can reduce the cost of system verification and system validation. As each requirement is written, the following questions need to be addressed:

- "Is this requirement verifiable,.i.e., is the requirement written such that its intent is clear and a system verification activity can be performed that will prove the system meets the requirement?"
- "What will constitute proof that the system has been verified to meet the requirement?"
- "What primary method will be used to verify the system meets this requirement?"

Failing to address these questions early can result in unpleasant surprises during your system 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 system verification and system validation from the beginning of their project and thus avoid the risks of not doing so.

Verification and Validation

A question I hear often is “What is difference between verification and validation?”

In general, verification refers to the basics (structure) of the item being verified, making sure it meets requirements that drive the creation of the item, whether it be rules on writing well-formed requirements, standards and best practices (external and internal) on the design, or requirements on the system. Then validation goes beyond the basics (structure) to how well the item (requirements, design, system) communicates or addresses stakeholder needs and expectations.

My use of the word system in this paper refers to the system of interest, no matter which level the system of interest may exist within the architecture (system, subsystem, assembly, component). Requirements are developed during the concept phase and represent a conceptual view of the system of interest. By design, I mean the transformation of the conceptual view of the system as communicated by the baseline requirement set into a physical realization of the system. This process evolves from a preliminary design, to a final design, and then to development, which transforms the design into the physical system (hardware) or code (software).

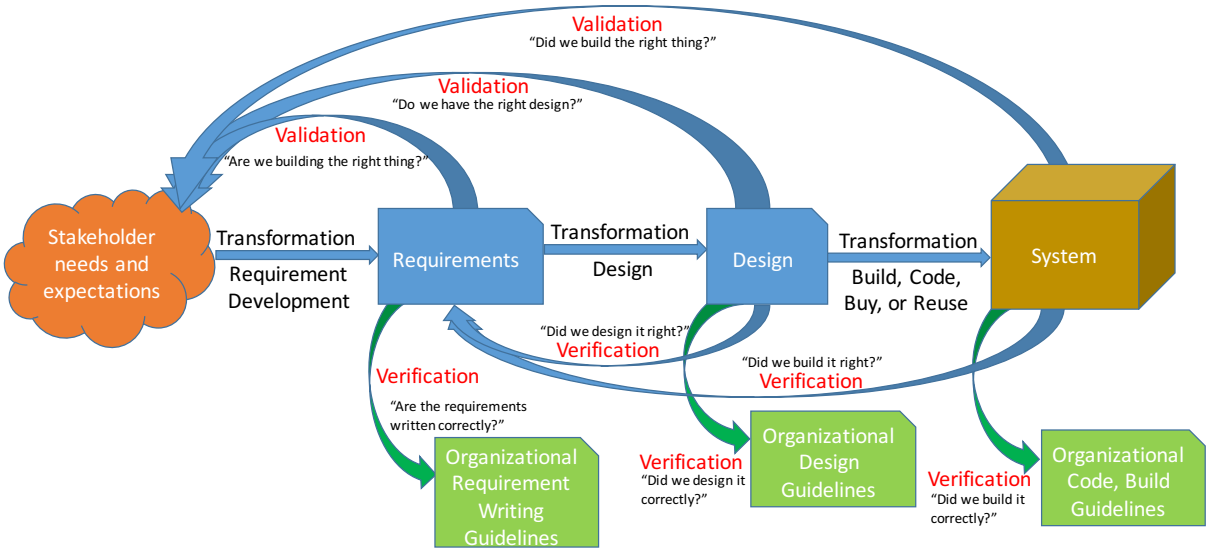

Figure 1. Verification and validation are the processes of confirming that artifacts generated during the transformation processes are acceptable.

While the terms verification and validation are widely used, I find that the true meaning of the concepts these terms represent are often misunderstood and the words often used interchangeably without making clear the context in which they are used resulting in ambiguity. With a single organization, when engineers are asked to define the terms you will get different, often conflicting, definitions. To avoid this ambiguity, each term needs to be preceded by a

modifier (i.e., the subject) which clearly denotes the proper context in which the term is being used, specifically requirement verification or requirement validation; design verification or design validation; system verification or system validation as shown in Figure 1.

A requirement is the result of a **formal transformation** of one or more stakeholder needs and expectations in to the requirement statement. Correspondingly, design is a result of **formal transformation** of requirements in to an agree-to design, and a system is a **formal transformation** of a design into that system as shown in Figure 1.

As illustrated in Figure 1, the process of creating a requirement involves:

- analysing one or more selected needs and expectations to obtain the necessary elements to be included in the requirement;
- selecting a format for the requirement statement;
- and identifying the characteristics of the desired result against the organizational guidelines and rules by which the requirement statement is to be written.

In this context, Requirement ***Verification*** confirms, by inspection, that the requirement contains the necessary elements and possesses the characteristics of a well formed requirement and conforms to the rules set forth in the organization's requirement development guidelines. *Requirement* ***Validation*** confirms, by inspection and analysis, that the resulting requirement meets the intent of the stakeholder need from which the requirement is derived. Thus a requirement statement (and sets of requirements) is/are confirmed by both verification and validation.

Based on this discussion, to help remove the ambiguity in the use of the terms "verification" and "validation", the following definitions for *requirement verification* and *requirement validation* are proposed in terms of the context of requirements.

- *Requirement Verification*: the process of ensuring the requirement meets the rules and characteristics defined for writing a good requirement. The focus is on the wording and structure of the requirement. "Is the requirement worded or structured correctly in accordance with the organization's standards, processes, and checklists?"
- *Requirement Validation*: confirmation the requirement is an agreed-to transformation that clearly communicates the stakeholder needs and expectations in a language understood by the developers. The focus is on the message the requirement is communicating. "Does the requirement clearly and correctly communicate the stakeholder expectations and needs?" "Are we doing the right things?" or "Are we building the right thing?"

Requirement verification and *requirement validation* activities should be done continuously as you develop the requirements as well as done as part of baseline of the requirement set activities performed during the System Requirements Review (SRR) or similar type of gate review.

Note: Most organizations do not make a distinction between requirement verification vs requirement validation. Rather they use only the phrase "requirement validation" to mean both. Using the phrase "requirement verification" often confuses the conversation in that many interpret "requirement verification" to have the same meaning as "system verification".

Figure 1 also carries these concepts forward to design and realization of the system under development.

Once you have a set of requirements baselined, the requirements are transformed into a design of the system. Most organizations have a set of guidelines or "golden rules" that guide the design process. These guidelines represent best practices and lessons learned the design team is expected to follow. As part of the design process, the design team may develop prototypes or engineering units. They will use these to run tests to fine tune their design.

After the design is complete for the system, there is usually a gate review where the design is both verified and validated (e.g. System Design Review (SDR), Preliminary Design Review (PDR), Critical Design Review (CDR). In this context, *design verification* has two aspects: 1) Does the design clearly represent the requirements that drove the design? and 2) Did the design team follow the organization's guidelines for design? Also as part of the gate review *design validation* is addressed to determine whether or not the resulting design for the system, when implemented, will result in the intended purpose being met in the operational environment and the stakeholder's expectations and needs being met.

Frequently, during these gate reviews, the design team "pushes back" on some of the requirements that proved difficult to meet or were deemed not feasible (cost, schedule, technology), and proposes changes to the requirements. When this happens, not only do the requirements needed to be changed, but also the stakeholder expectations and needs from which the requirement was derived may need to be examined, which could result in a scope change.

Design verification and *design validatio*n activities should be done as part of a continuous process during the design phase as well as during the the base-lining of the design in the gate review(s).

Based on this discussion, to help remove the ambiguity in the use of the terms "verification" and "validation", the following definitions for *design verification* and *design validation* are proposed.

- *Design Verification*: the process of ensuring the design meets the rules and characteristics defined for the organization's best practices associated with design. The focus is on the design process. "Did we follow our organizations guidelines for doing the design correctly?" The design process also includes ensuring the design reflects the design-to requirements. Thus, design verification is also a confirmation the design is an agreed-to transformation of the design-to requirements into a design that clearly implements those

requirements correctly. “Does the design clearly and correctly represent the requirements?” “Did we design the thing right?”

- *Design Validation*: confirmation the design will result in a system that meets its intended purpose in its operational environment. Will the design result in a system that will meet the stakeholder expectations (needs) that were defined during the scope definition phase that occurred at the beginning of the project? The focus is on the message the design is communicating. “How well does the design meet the intent of the requirements?” “Do we have the right design?” “Are we doing the right things?” “Will this design result in the stakeholder expectations and needs being met?”

Once the design is baselined, the design is transformed; via build, code, buy, or reuse; into the system of interest. Similar to the discussion for the design process, most organizations have a set of guidelines or “golden rules” that guide the build (manufacture or code) process. These include workmanship and quality control requirements on the organization.

After the system has been built or coded, there will be a gate review where the system is both verified and validated. At this stage of the system lifecycle, the concepts of *system verification* and *system validation* take on a more formal meaning. Thus, the Systems Engineering (SE) lifecycle processes include the processes of *System Verification* and *System Validation*. Each of these represent a set of activities (test, demonstration, inspection, analysis) that cumulate with one or more gate reviews associated with the acceptance of the system by the customer. Thus *System Verification* is a formal SE process and has a legal aspect where the developer is proving the baselined requirements, which are a type of contract and are legally binding, have been meet.

In this context, *system verification* has two aspects: 1) Does the built or coded system of interest clearly represent the requirements that drove the design? Did we build the right thing? and 2) Did the build or code team follow the organizations guidelines for manufacturing and coding?

Following *system verification*, *system validation* is performed. Again, *System Validation* is a formal SE process and has a legal aspect where the developer is proving whether or not the built or coded and verified system, results in the intended purpose being met in the operational environment and the stakeholder’s expectations and needs being met. Like the system requirements, the baselined stakeholder needs and requirements defined during the scope definition phase can also be considered part of a contract and are legally binding.

Based on this discussion, to help remove the ambiguity in the use of the terms “verification” and “validation”, the following definitions for *system verification* and *system validation* are proposed. These definitions are also illustrated in Figure 1.

- *System Verification*: System Verification is done after design and build or coding, making sure the designed and built or coded system meets its requirements. The focus is on the built or coded system and how well it meets the agreed-to requirement set that drove

the design and fabrication. Methods used for *system verification* include: test, demonstration, inspection, or analysis. "Did we build the thing right?" Also included in *system verification* is a determination that the team responsible for building or coding the system of interests followed the organization's rules, guidelines, and best practices associated with manufacturing and coding. The focus is on the manufacturing or coding processes. "Did we follow our organizations guidelines for manufacturing or coding correctly?"

- *System Validation*: System validation occurs after *system verification* and makes sure the designed, built, and verified system meets its intended purpose in its operational environment. The focus is on the completed system and how well it meets stakeholder expectations (needs) that were defined during the scope definition phase that should have occurred at the beginning of the project. "Did we build the right thing?"

System verification and *system validation* processes and activities are directly related to the contractual obligation concept for a requirement statement and set of requirements. It is through these activities that we prove we have met both the agreed-to requirements and the agreed-to needs of the entities who are the source of or own them. This is often accomplished as part of certification and acceptance activities.

Verification and validation and the Systems Engineering "V" model

While the previous section presented definitions that are useful in helping address the issues of ambiguity in the use of the terms verification and validation, the system engineering process is more complex. Systems Engineering is an iterative and recursive process. Requirements development and design occur top-down as shown on the left side of the SE "V" as shown in Figure 2.

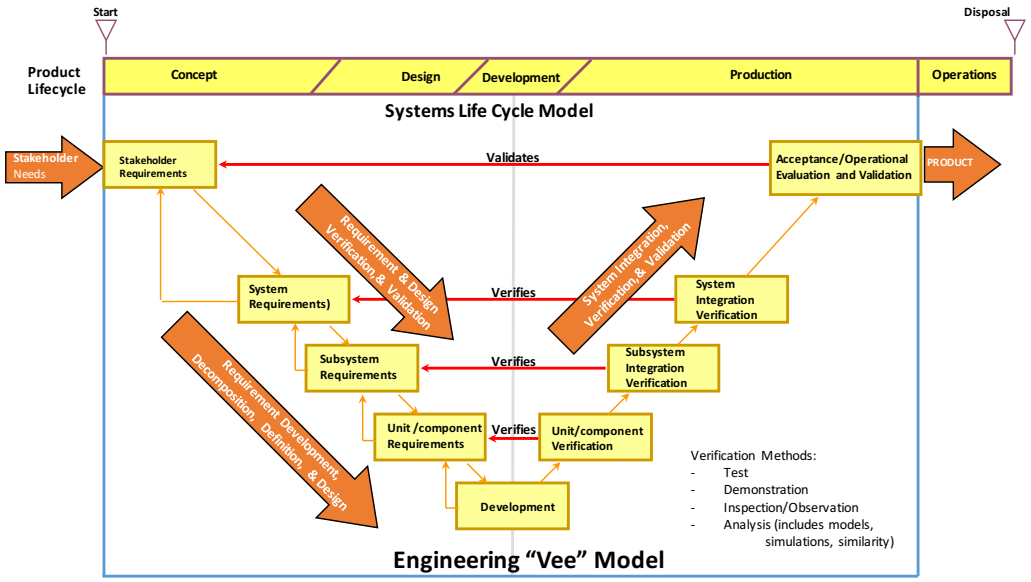

Figure 2: Systems Engineer "V" Model

As shown in Figure 2, Systems engineering starts with the concept stage where stakeholder needs, expectations, and requirements are elicited, documented, and baselined. This is part of defining the scope of the system to be developed. Then the stakeholder needs, expectations and requirements are transformed into a set of system requirements which are baselined via *requirements verification* and *requirements validation* as discussed above. Once the system requirements are baselined, design results in a system architecture in which the subsystems are defined. This design is baselined via the *design verification* and *design validation* process discussed above.

For each subsystem, the above cycle of stakeholder subsystem needs, expectations, and stakeholder requirements are defined and transformed into subsystem requirements, which are baselined via *requirements verification* and *requirements validation* as discussed above. Once the subsystem requirements are baselined, design results in a subsystem architecture in which the units/components are defined. This design is baselined via the *design verification* and *design validation* process discussed above. This process continues until the organization makes a buy, build, code, or reuse decision.

System verification and *system validation* occur bottom-up as shown on the right side of the SE "V" as shown in Figure 2.

Once all the components that make up the subsystems are developed, *unit/component verification* and *unit/component validation* take place as described above. Once these activities are complete, the units/components are integrated together and then the resulting subsystems are verified and validated as described above. Once the *subsystem verification* and *subsystem validation* activities are complete the subsystems are integrated together and then *system verification* activities are completed.

Following *system verification* activities, *system validation* activities are performed. This could be done in the form of acceptance and/or operation evaluation and validation activities. In the end, proof will be documented that can be evaluated by the customer to determine *system validation* activities have been completed successfully, stakeholder needs have been met, and the system will operate as intended in its operational environment.

Following the customer evaluation, the system can be accepted and ownership transferred to the customer.

The focus of the rest of this paper is on *system verification* and *system validation* planning – address the right side of the Systems Engineering "V". (For more insight into *requirement verification* and *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 in its operational environment. 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 *system verification* and *system 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 *system verification* and *system validation* activities from the beginning of the project and continuing to address *system verification* and *system 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 *system verification* and *system validation* planning from the beginning of your project starting with scope definition and continuing throughout the requirement definition, management, and design processes.

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

Impact of requirement defects on system verification and system 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 when performing *system verification* and *system validation* activities.

As shown in Figure 3[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 product 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 3, *system verification* and *system validation* activities occur during development and acceptance testing as well as during initial operations.

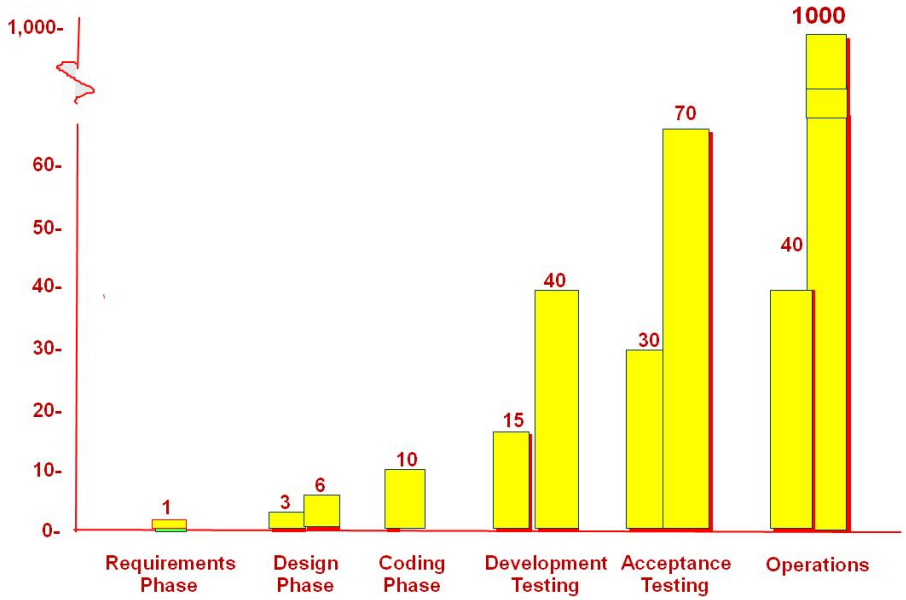

Figure 3: Cost to fix requirement defects

As shown in Figure 3[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 product 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 3, *system verification* and *system validation* activities occur during development and acceptance testing as well as during initial operations.

Note: Project managers often don't understand these concepts nor the resources needed to accomplish the activities associated with the product lifecycle phases. I have seen cases where the cost to do system verification and system validation activities ended up costing between 50%-70% of the total development cost for a new system. [Hooks 2001]. *The higher percentages occur due to having a poor set of requirements and thus the resulting rework. Project managers who only budget for maybe 20% of the development costs for system verification and system validation are in for a rude awaking and are setting their project up for failure.*

Early identification of defective requirements prevents re-work and results in significant cost savings. Conversely, allowing these defects to go undetected until *system verification* and *system validation* will result in significant cost and schedule risk.

Considering your *system verification* and *system validation* strategy early in the development lifecycle provides a foundation for estimating your cost and schedule for these activities.

Resist the temptation to reduce your *system verification* or *system validation* activities when you are over budget or have schedule problems. As shown in Figure 3, 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 do *requirement verification* and *requirement validation* and baselining your requirements before design can result in *requirement verification* and *requirement validation* AND *design verification* and *design validation* activities being performed together, during what was supposed to be just *design verification* and *design validation* activities. This could result in a design that meets the requirements (passes *design verification*) but if they are the wrong or defective requirements, the design will fail *design validation*.

Even worse, failing do *requirement verification* and *requirement validation* and baseline before design and build can result in *requirement verification* and *requirement validation* AND *design verification* and *design validation* activities being performed at the same time your focus should be on *system verification* and *system validation*. What happens if you verify that your system meets its requirements, but the requirement set is defective with ambiguous, missing, or incorrect requirements? *Design verification* and *design validation* may not expose these defects.

Because the set of requirements is defective, they don't correctly and completely communicate stakeholder needs and expectations resulting in your system failing to pass *system validation*. If your system fails *system validation*, it will not meet the stakeholder needs expectations and will not accomplish its intended purpose in its operational enviornment. 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 requirement verification and validation, design verification and validation, and system verification and 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.

Note: The concept of "system validation" discussed above is part of the formal product development lifecycle addressed in most textbooks. What most people don't recognize is that there is also an "informal" system validation that takes place after a product is released for use. This informal system validation is communicated to the developing organization is various forms in addition to product failures during operations which leads to costly warranty work. These informal system validation actions include, returns, negative customer comments and reviews, recalls, declining reputation, decreased sales, sales projections not being met, etc. All of these things have a direct impact on profits and mission success. In today's world of limited funding, profit margins are getting tighter and tighter. Companies cannot afford to fail informal system validation any more than they can afford to fail formal system validation.

The risks of cost overruns and schedule slips associated with failing to develop defect free requirements and failing to plan ahead for *system verification* and *system validation* activities

are mitigated by adopting good requirement development practices and planning ahead for system verification and system validation activities during the scope definition and requirement development and management lifecycle phases.

The importance of thinking ahead to system verification and system validation

Unfortunately, there are all too 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 System Verification and System 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 needs, expectations, and requirements? In the final analysis, system validation is based on making sure the product addresses the stakeholder needs, expectations, and requirements 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; drivers and constraints, and concepts for 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 (interfaces).

Thinking ahead to *system verification* and *system validation* begins during the scope definition phase. During this phase, in addition to defining stakeholder expectations, the project is developing drafts of their Program/Project Management Plan (PMP), Systems Engineering Management Plan (SEMP), and the Master Integration, Verification, and Validation (MIVV) Plan. Schedules and budgets are being formulated and the Work Breakdown Structure (WBS) is being defined. Because *system verification* and *system 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 and resulting technical requirements

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 in its operational enviorment.

Requirement validation addresses how well the requirements communicate the stakeholder needs and 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 *system verification* and *system validation* perspective, the stakeholders that will have anything to do with any of the varication 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 *system verification* and *system validation* activities will result in sufficient "proof" such that they will approve and accept the product. During scope definition, a preliminary draft of your MIVV Plan should be developed.

- *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. *System verification* and *system validation* activities must be completed successfully to show "proof" that the system is compliant with the identified drivers and constraints. Failing to show compliance, will result in the system failing certification for use.

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 *system verification* and *system*

validation planning must include producing the data required by the regulatory agencies to "prove" compliance with their requirements.

What if you are developing a medical product? What system verification and system 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 and constraints. Compliance is expected and failure to address relevant drivers and constraints can result in failure during *system verification* and s*ystem validation* and product rejection by the customer *even if the customer did not explicitly communicate these 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 associated with the requirements that implement your MOPs and KPPs is low, they are also high-risk requirements and must be traced closely by project management. Planning for system 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 system concepts that address the needs and expectations of all the stakeholders and address each of the product's lifecycles -- for both nominal and off-nominal conditions. Scenarios and system concepts for system verification and system validation need to be defined and documented so you have the information needed to develop your WBS, budget, schedule, and draft MIVV plan. 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 system verification and system validation activities.

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

Your system may need specific features dedicated to support your system verification and system validation activities especially when using test or demonstration as methods for system verification and system validation. Additional interfaces may be needed to give you access to data that is needed for system verification and system validation, but not normal operations. This could result in the need for additional connectors, test points, 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 models needed for system verification and system validation can be very expensive to develop and take up a large portion of the project's resources. In addition, you will need to validate the model. To do this, you may have to develop a simulation of your system. This simulation will be 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 system 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 or flight 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 system verification and system validation possible in some cases and reduce costs in other cases. Addressing the equipment and facilities needed to perform system verification and system 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 verification and system validation activities. Planning for system verification and system validation early

permits the execution of concurrent activities that will allow you to complete your system 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 external systems over which you have little or 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 make sure (verify) your system meets 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 AND is protected from outside threats across your interfaces.

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

Addressing System Verification during the Requirement Definition Phase

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

During the scope definition phase discussed above, I discussed system verification and system validation from the 10,000-foot level. Basic concepts and scenarios for completing your system verification and system 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 MIVV Plan.

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. Once your requirements have stabilized, you also need to start defining the activities needed to verify your

system meets its requirements. Let's look at the characteristics of good requirements from a system 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. Remember that integration, system verification, and system verification costs can be 50% or more of your development budget. You don't want to spend time and effort proving the system meets requirements that were not 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. We recommend you take a "zero-based" approach to defining your requirement set. By this we mean that you define the minimum set that is necessary and sufficient to meet the stakeholder needs and expectations. Then if anyone wants to add additional requirements, they have to "fight" to get the additional requirement(s) added to the set. Insist that each requirement that is submitted has rationale documented with the requirement that clearly communicates the need for that requirement.

To test for to see if a requirement is needed, 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. Insist that each requirement that is submitted has the recommended verification method documented with the requirement.

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/achievable*

A third mandatory characteristic of a requirement is that it must be attainable/achievable given the existing budget, schedule, and level of technical maturity. A requirement that is not attainable will fail system 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). If the requirement is a performance value, you need to do an assessment as to the maturity of the technology needed to meet the requirement. If the current maturity is low, the requirement represents a risk to your project and you need a plan to mitigate that risk and mature the technology. This could represent both a cost and schedule risk as well.

- *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].) In addition, you can get a copy of the "INCOSE Guide to Writing Requirements, 2015".

- *Requirement is documented at the level where it will be verified.*

Proper flow down (allocation) of requirements to system architectural levels is very important to effective system verification and system validation. All requirements must be documented at the level where they will be implemented and verified.

As shown in Figure 2, the SE "V" model, requirement definition and design are top-down activities during which requirements for the system as a whole are defined and then decomposed into sub-systems defined during the design activity while allocating requirements to each subsequent level of the architecture, descending down the left side of the "V".

Once the system has been decomposed into the various units/components that make up the system; integration, system verification and system validation activities begin, ascending up the right side of the "V". During integration, lower level units/components are integrated together to form the subsystems. Prior to integration into subsystems, the lower level units/components are verified and validated and compatibility is assessed to ensure these lower level entities meet the requirements that drove their design and will successfully combine into the subsystems at defined interfaces. Once the subsystems have been formed, the process repeats.

Prior to integration into the system, the subsystems are verified and validated and compatibility is assessed to ensure the subsystems meet the requirements that drove their design and will successfully combine into the system at the defined interfaces.

Once, the subsystems have been integrated together to form the system, system verification activities are performed to ensure the integrated systems meets the requirements that drove the system design.

Once the entire system has been verified, the system is ready for system 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.

- *Requirement is 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. Once allocated, the next level will derive children requirements, that when implemented at that level, will contribute to the parent requirement being met.

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.) This, in turn, could result in the system failing either system verification or system validation.

- *Requirement addresses an internal interface*

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 documented 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 system verification. (The inclusion of a reference to where the interface is defined is a common part 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.)

- *Requirement is traceable to a parent or source*

Traceability is the concept that all requirements need to be linked (traced) to their parent or source. Tracing to a parent or sources is also a method to determine if a requirement is needed. When there is no trace to a parent or source, 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 system verification and system 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 or source requirement. Assessment of whether or not the children requirements linked (traced) to the parent or source requirement are necessary and sufficient to meet the intent of the parent to is needed 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 V model, integration is a bottoms-up process. When integrating at one level, you need to make sure system verification and system validation are complete at the previous level. Successful system verification at the lower level is a prerequisite to successful system verification at the next level. This is especially true when it comes to interfaces.

> *A key point is that before verifying the subsystem or system meets a requirement, if that requirement has children requirements, verification that the previous level meets the children requirements must have been completed before verifying the parent requirement has been met. Proof that the children requirements have been met is needed (a prerequisite) before verifying that the parent has been met. Allocation and traceability allow this to happen.*

When planning system verification at one level, always include a step in your process to look at the system verification documentation at the previous level to make sure it has been successfully completed before you do system verification at your current level.

Planning your system verification and system validation activities

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

Thinking about system verification and system validation while developing your requirements

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

Addressing system 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 modeling.) "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 verification by Test includes some analysis activities and Analysis may include some testing activities. The best practice is to state a single primary method for verification. *Note: Make sure you understand the difference between test, inspection, and analysis as activities, vs Test, Inspection, and Analysis as methods of system verification. The concepts are completely different, yet often this difference is often not understood.*

   Risk is a major consideration when choosing the system verification method. This is a very important issue from a project management standpoint. System 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 integrated and 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 nor cost effective to develop.

All of these considerations need to be addressed when deciding on the system 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.

The rationale for why you are using a specific verification method needs to be included in your plan for each requirement.

Because choosing the system verification and system validation method is really a cost, schedule, and risk decision, the Project Manager and Lead Systems Engineer need to be involved in the decision as to which system verification method will be used and which activities will be included as part of the system verification method agreed to.

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

Develop a scenario or operational concept to determine which activities (test, inspection, analysis) 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 your system verification and system validation activities. Where will the system verification or system 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 system verification or system validation activity? Who must observe the test or demonstration activity, sign off that the activity was performed per procedure, perform the analysis of the data resulting from the system verification or system validation activity, or inspect the system? What training or certifications do the personnel need to have to participate in their assigned role? What safety considerations are there? What resources are needed to complete this activity? How long will it take? How much will it cost?

The system verification and system validation (aka "test") engineers will ask these questions. Project managers need to know the resources needed, the cost, and schedule needs to accomplish the system verification or system 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 system verification or system validation 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, management, and design are an ongoing activities. 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 system verification and system validation activities be performed?"

   As part of your system verification and system validation planning and concept development efforts, consider timing and develop a "when" strategy. Remember integration, system verification, and system validation are bottoms-up process activities. When and where in your product integration lifecycle will you verify the requirement has been met?

   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 (time consuming) to trace the problem to its source for a complex system. Once you find the source of the problem, it will be more 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 components meet their requiremens. Alternatively, you may be able to accept the risk and wait to do system verification at a higher level of the architecture - if possible. Another approach is to use a less costly and time consuming method of system 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 in its operational environment (validation). Even though every part, component, or subsystem 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 preferred method of verification. *(Using Analysis by similarity is very risky!)* 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 expectations for the evidence needed and level of confidence in this evidence 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 in your system development lifecycle, putting your project at risk.

Addressing system verification, system validation, certification, and acceptance in Vendor/Supplier Agreements and Statements of Work

If an end-item is to be outsourced to a vendor/supplier, the requirements for each end-item are typically documented in a System Requirement Document (SRD) or a Technical Data Package (TDP). Along with this will be a Statement of Work (SOW) that contains requirements on the vendor/supplier concerning design, development, system verification, system validation, and acceptance. The SOW contains process and workmanship requirements levied on the vendor/supplier on the tasks they need to perform as part of the contract to fabricate, test, transport, and install a specific end-item. The SRD contains requirements intended to communicate to the design team the stakeholder needs and expectations in an unambiguous and clear technical language, which can be though of as the "design-to" requirements. 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 SRD or end-item requirements may be contained directly in the SOW. The method of how the system will be verified that it meets all of these requirements needs to be defined as part of the vendor/supplier agreement process.

Key deliverables specified in the SOW need to include an allocation matrix that shows how the vendor/supplier allocates your requirements to the parts the vendor/supplier is responsible for providing as well as a traceability matrix showing how the vendor's/supplier's derived requirements trace to your parent requirements. In addition, you need to address what proof

you will need from the vendor/supplier as to how they have met your requirements and what is needed for the customer to accept the system from the vendor/supplier.

Together, the SOW and SRD or TDP results in a vendor/supplier agreement which is a contract between the customer and vendor/supplier. This contract needs to clearly state your expectations for the system 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 system verification and system validation? Missing, vague, or open-ended system verification and validation plans can overrun the supplier's budget or leave you, the customer, without confidence in the system. Addressing system verification and system validation in vendor/supplier agreements protects all parties in the contract.

The documentation that provides proof that these activities have been successfully completed needs to be clearly listed as contract deliverables. This documentation is used as part of your overall system verification and validation activities. You can accept the supplier's documentation as proof, or for critical items, you may want to perform your own system verification and system 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" and is treated as a valid method for system verification and system validation.

If you are a vendor/supplier and your customers will be performing their own acceptance activities, assess your system verification and validation activities 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 in the operational environment as advertised or as described in your supplier agreement/contract. In either case, both organizations need to ensure their acceptance plan reflects the actual requirements and does not introduce previously unstated requirements. [Hooks 2001]

Tools to help with your system verification and system validation planning

This section addresses some of the key tools that are used to help with system verification and system 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 system verification and system 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 system verification and system validation philosophy for the project or program.

- Master Integration Verification and Validation (MIVV) Plan: Expands and implements the system verification and system validation philosophy of the project or program as defined in the PMP and SEMP. The MIVV Plan defines the detailed processes and approach that will be used for system verification and system validation. The information in the MIVV Plan is used to manage and plan system verification and system validation activities. The MIVV Plan further defines the integration, system verification, and system 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 each requirement, and needs to be captured and documented with the requirement. You can also include the level, lifecycle phase, and success criteria for the verification activities. This information will be used to create the Requirements Verification Matrix.

- Requirements Verification Matrix (RVM): A matrix that lists each requirement, verification method (Test, Inspection, Demonstration, Analysis), responsible organization, level (component, subsystem, system), phase (design, manufacture, integration), location or facility, and often includes a short description of the success criteria for successful verification. The RVM is often included as part of the SRD/SRS. If the information needed to create the RVM is included as part of the requirement attributes, then your Requirement Management Tool (RMT) will be able to generate the RVM for you.

- System Integration, Verification and Validation (SIVV) Plan: Implements the project's MIVV Plan system verification and validation activities for a given component, subsystem, or system as documented in the SRD/SRS and associated RVM. The SIVV Plan 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 MIVV Plan 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 RVM?" 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 a 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.) Using a separate VRD to document this information allows you to baseline your requirement set and minimizes changes to your SRD as the information in the VRD matures.

- 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 system 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 procedures and an integrated schedule for completing all the defined tasks involved in verification, validation, acceptance, certification, 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 SIVV Plan. 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 system verification or validation activities defined in the TDS. This is the document actually used to accomplish the system verification or system 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 the system meets a requirement or set of requirements as defined in the VRDSs or validated as defined in the MIVV Plan and SIVV Plan.

  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 system verification or system validation activity standpoint, TPSs are prepared based on the information contained in the SIVV Plan, VRDSs, and TDS that the TPS is being prepared to satisfy. All TPS line items involving a system verification or system validation activity for a specific requirement must include a mandatory inspection point (MIP) that includes a Quality Assurance/Quality Engineer (QA/QE) stamp.

Wrap up and Parting Thoughts

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

We started with a discussion on the terms verification and validation and made the point that these terms are ambiguous unless a modifier is used (requirement, design, or system) that clearly communicates the context in which the term refers.

The emphasis of the rest of this paper was to present best practices and tools and proven methods project teams can use to address planning for system verification and system 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 system verification and system validation starting at the beginning of the system lifecycle from a risk mitigation standpoint and the importance of project management ensuring that requirement development, management, and design processes address system verification and system validation activity risks, cost, and schedule.

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

- During scope definition, involve those personnel associated with system verification and system validation with the development of a set of scenarios or operational concepts used to successfully complete needed system verification and system validation activities. This knowledge is documented in the PMP, SEMP, and MIVV Plan.
- Address system verification and system validation early in the system development lifecycle 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 system 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 (PIVV plan, RVM, VRDS, TDS, and TPS) to facilitate planning for and completing system verification and system validation activities.
- Ensure project management is involved in key decisions associated with balancing the risks against cost and schedule when selecting system verification and system validation methods and then determining at what level, phase, facility or location, the system will be verified that it meets its requirements it was designed to.

Addressing system verification and system 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 15 years, Lou has worked for Compliance Automation (dba Requirements Experts), where he has conducted over 160 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.

Lou also maintains a blog on requirement development and management topics. You can access his blog articles at http://www.reqexperts.com/blog

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

Appendix A: Verification Requirement Definition Sheet (VRDS)

[Program or Project Name] [System/Subsystem/Component Name] **Verification Requirement Definition Sheet**			[Project Logo]
[Requirement Number]			
Verification Requirement Number: VR [Rqmt Number]		**Parent Requirement(s):** [Rqmt Number(s)]	
Requirement Text:			
System 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]:			
System 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 children requirement VRDSs:** xxxxxxxxxxx			
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 and the defined success criteria has been achieved. **Events preceding the System Verification Activity:** xxxxxxxxx **Estimated Duration of the System Verification Activity:** [TBS]			
Closure Of the VR [Rqmt Number]			
Date Closed: mm/dd/yyyy		[ ] **Safety Critical (Y/N)**	
xxxx Quality Engineer:	**xxxx System PM:**	**xxxx Chief Engineer:**	**Customer Systems 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 system verification requirements as tailored for the system verification activities to prove the system meets the requirements that drove the system's design.

- **VRDS Title**

The title contains the name of the Component/Subsystem/System the requirement concerns

- **Requirement the System Verification is addressing**

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 method to obtain the measure of the success criterion. Supporting activities 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 activities will be inherent in any verification method, so it is preferred to focus on the final method that generates the measure of the success criterion, even when multiple activities are involved.) *(Remember that the methods and activities, may share the same names, but the concepts are totally different.)*
- **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 (proof) needed to verify the system meets the 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 (which activities) will be used 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 activities 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 focus of the rationale statement includes an explanation and justification for "why" the specific verification method, success criterion, measure, constraints, conditions, and assumptions have been selected. 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.

- **Verification Implementation**

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

- **Lifecycle Phase**

  The Verification Phase states in which system integration lifecycle phase the system will be verified to have met the requirement.
- **Applicable Documents**

  Any documents dealing with the system verification for 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 whose verification activities must be completed and signed off before the system 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 to prove 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 system verification. This includes verification that children requirements have been met, 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 system verification for this requirement. In cases where one activity will involve the system verification for multiple requirements, state the time to complete that activity.

- **Closure of the Verification Requirement**

  This section is completed once the system verification activity has been completed and the closure data and closure documentation has been completed and documented in this section of the form. 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 activity results (proof the success criteria has been met) and their confirmation that system meets the stated requirement.

Appendix B: Task Definition Sheet (TDS)

[Program or Project Name] [System/subsystem/component Name] 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. Include VRDSs that are addressed by 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.