27th Annual INCOSE International Symposium (IS 2017)
Adelaide, Australia, July 15-20, 2017

On the Use of the Terms Verification and Validation

Michael J. Ryan
Capability Systems Centre,
Canberra, ACT, Australia, 2600
m.ryan@adfa.edu.au

Louis S. Wheatcraft
Requirements Experts,
Aurora, CO
louw@reqexperts.com

Abstract. While the concepts of verification and validation are commonplace, in practice the terms are often used interchangeably and the context of their use is not made clear, resulting in the meaning of the concepts frequently being misunderstood. This paper addresses the use of the terms verification and validation and identifies their various meanings in terms of the context in which they are used, particularly as defined in the relevant standards and literature, as well as everyday usage. In particular, it is identified that both terms are ambiguous unless a modifier is included in front of the word, clearly indicating to which context the term is referring. This paper illustrates that failing to use a modifier—thus assuming the reader understands the context in which the terms are used—is the most common reason for ambiguity and misunderstanding. The paper presents a simple model of the system design process to allow the nature of verification and validation to be understood better within a given context. Definitions are then proposed in order to disambiguate the various uses of the terms.

Introduction

The terms verification and validation are used commonly, but the words are often used interchangeably, the context of their use is not made clear, and the terms are used to convey considerably different intent, resulting in the true meanings of the concepts often being misunderstood. This is true particularly when the terms are combined into the generic terms verification and validation (V&V); independent V&V (IV&V); or integration, verification and validation (IV&V).

The term “verification” is used commonly to refer to an activity associated in some way with requirements. These requirements can be on the system being developed or on the organization responsible for developing the system. Some refer to verification as an activity or systems engineering process an organization performs to make sure the designed and built system meets each of the requirements that drove its development.

The term “validation” has similar difficulties in that it is commonly used to refer to an activity in respect to stakeholder needs and expectations that relate to the original intent or expectations. Some consider “validation” to be an activity or process to ensure that a given outcome (requirement statements or requirement sets, design, or system) addresses the stakeholder needs. Others will use the term “validation” to refer to the systems engineering process of system validation by which the project shows that the system under development accomplishes its intended purpose in its operational environment.

Using the terms interchangeably with the assumed or implied meaning and context leads to confusion and misunderstanding. When referring to IV&V, the ambiguity in the general use of the terms verification and validation creates even greater difficulties, especially if an organization is being contracted to perform “IV&V”. First, the ‘I’ in IV&V is often used to mean ‘independent’ V&V in which an outside organization is called in to undertake ‘V&V’ as an activity. However, it is assumed what term each of the “Vs” refers to: “validation and verification” or “verification and validation”. Verification of what? Validation of what? Without a qualifying adjective, context is assumed and it is therefore not clear what ‘V&V’

refers to: requirements, design, or the system that has already been designed and built in accordance with the requirements.

Sometimes, however, the 'I' means 'integration' when referring to the right-hand side of the systems engineering 'Vee' model that depicts the processes of integration, verification, and validation. Clearly, by not being precise in the use of the terms and not indicating the context intended, confusion can result.

The confusion in terms does not just arise as a result of the 'loose' use of the terms. The nature of verification and validation means that the associated activities occur within each stage of the systems life cycle and particularly during development—for example, it is difficult to validate a requirement without doing some design and some analysis of ways to implement it, and a prototype may be necessary to validate user interface requirements (Boehm, 1984).

This paper addresses the use of the terms "verification" and "validation" and identifies their various meanings based on context, particularly in the relevant standards and literature, as well as in everyday usage. The paper illustrates that the meaning of the terms can be very different depending on the specific context, which is shown to be the most common reasons for ambiguity and misunderstanding. It is identified that both terms are ambiguous unless a modifier is included in front of the word clearly indicating the context to which the term refers, specifically: to requirements, the design, or to the system under development.

Common Definitions for Verification and Validation

We investigate in this section the various definitions of the terms in the relevant standards and literature.

From IEEE-1012-2012 (IEEE Computer Society, 2012):

- *Verification:* An early version of IEEE-1012 (the 1984 version) defines verification as: "the process of evaluating a system or component to determine whether the products of a given phase satisfy the conditions imposed at the start of that phase". This was modified in IEEE 1012-2012 to be: "(A) The process of evaluating a system or component to determine whether the products of a given development phase satisfy the conditions imposed at the start of that phase. (B) The process of providing objective evidence that the system, software, or hardware and its associated products conform to requirements (e.g., for correctness, completeness, consistency, and accuracy) for all life cycle activities during each life cycle process (acquisition, supply, development, operation, and maintenance); satisfy standards, practices, and conventions during life cycle processes; and successfully complete each life cycle activity and satisfy all the criteria for initiating succeeding life cycle activities. Verification of interim work products is essential for proper understanding and assessment of the life cycle phase product(s). (IEEE Computer Society, 2012)
- *Validation:* An early version of IEEE-1012 (the 1984 version) defines validation as: "the process of evaluating a system or component during or at the end of the development process to determine whether it satisfies specified requirements". This was modified in IEEE 1012-2012 to be: "(A) The process of evaluating a system or component during or at the end of the development process to determine whether it satisfies specified requirements. (B)The process of providing evidence that the system, software, or hardware and its associated products satisfy requirements allocated to it at the end of each life cycle activity, solve the right problem (e.g., correctly model physical laws, implement business rules, and use the proper system assumptions), and satisfy intended use and user needs." (IEEE Computer Society, 2012)

ISO/IEC 15288 (2015) provides the following definitions of (system) validation and (system) verification:

- *Validation:* "confirmation, through the provision of objective evidence, that the requirements for a specific intended use or application have been fulfilled [ISO 9000: 2000] NOTE Validation in a system life cycle context is the set of activities ensuring and gaining confidence that a system is able to accomplish its intended use, goals and objectives." (ISO/IEC 15288, 2015)
- *Verification:* "confirmation, through the provision of objective evidence, that specified requirements have been fulfilled [ISO 9000: 2000] NOTE Verification in a system life cycle context is a set of activities that compares a product of the system life cycle against the required characteristics for that product. This may include, but is not limited to, specified requirements, design description and the system itself." (ISO/IEC 15288, 2015)

The INCOSE SE Handbook, v4, builds on the definitions of verification and validation used in ISO/IEC 1528 and adds the following notes:

- *"Verification is a set of activities that compares a system or system element against the required characteristics. This may include, but is not limited to, specified requirements, design description, and the system itself." "Verification ensures you built the system right."*
- *"Validation is the set of activities ensuring and gaining confidence that a system is able to accomplish its intended use, goals, and objectives (i.e., meet stakeholder requirements) in the intended operational environment." "Validation ensures you built the right system."*

ISO/IEC/IEEE 29148:2011 (2011) repeats the ISO/IEC 15288 definitions of (system) verification and (system) validation and provides the following definitions of requirements validation and requirements verification:

- *Requirements validation:* "confirmation by examination that requirements (individually and as a set) define the right system as intended by the stakeholders. NOTE Adapted from EIA 632:1998" (ISO/IEC FDIS 29148, 2011).
- *Requirements verification: "confirmation by examination that requirements (individually and as a set) are well formed NOTE 1 Adapted from EIA 632:1998 NOTE 2 This means that a requirement or a set of requirements has been reviewed to ensure the characteristics of good requirements are achieved." (ISO/IEC FDIS 29148, 2011)*

ANSI/EIA-632 does not define verification and validation as terms in their own right but defines 'end product verification' and 'end product validation' and 'requirements validation' (note that ANSI/EIA-632 defines the end product as "The portion of a system that performs the operational functions and is delivered to an acquirer.":

- *End product verification. "Confirmation by examination and provision of objective evidence that the specified requirements to which an end product is built, coded, or assembled have been fulfilled." (ANSI/EIA 632, 1998)*
- *End product validation. "Confirmation by examination and provision of objective evidence that the specific intended use of an end product (developed or purchased), or an aggregation of end products, is accomplished in an intended usage environment." NOTE "2. End product validation is used to demonstrate that the product developed or purchased satisfies the validated acquirer requirements in the context of its intended*

use. NOTE 3 Validation against other stakeholder requirements, generally, is not required." (ANSI/EIA 632, 1998)

- *Requirements validation. "Confirmation by examination that requirements (individually and as a set) are well formulated and are usable for intended use." (ANSI/EIA 632, 1998)*

In addition to the definitions in major standards, there are a number of definitions provided in the literature:

- The IEEE 1012-1984 (IEEE Computer Society, 1984) definitions are quoted by Leffingwell and Widrig (2000) and used by Rakitin (2001).
- Engel uses the IEEE 1012-2012 definition for verification but offers the following definition for validation: "The process of evaluating a system, to determine whether it satisfies the stakeholders of that system". (Engel, 2010)
- Verification: "Am I building the product right?" Validation: "Am I building the right product?" (Boehm, 1984)
- Verification: "The formal process of checking the designs, code, test plans and final software products against requirements." Validation: "The process of checking the results of each stage of the software life cycle to see if it has the correct relationship to results from the previous stage." (Davis, 1993)
- "Requirements validation is the process of certifying the requirements model for correctness against the user's intention" (Loucopoulos and Karakostas, 1995).
- Requirements validation is concerned with "… checking the requirements for omissions, conflicts and ambiguities and for ensuring that the requirements follow quality standards". (Sommerville, I., and P. Sawyer, 1997)
- Validation is a "… process carried out to demonstrate that one or more requirements are clearly understood and that it is possible to satisfy them through design work within the current technological state of art". (Grady, 1997) Grady (1997) disagrees that validation means that the right system is being or has been built, arguing that the meaning should be truncated to just be that the right system is being or going to be built—Grady (1997) refers to all post-design V&V work as forms of verification that requirements have been complied with.
- Dzida and Freitag (1998) relate verification to correctness and validation to appropriateness.
- As an example of how the 'adjective' changes the context somewhat, Pohl (2010) offers the following definition of validation in requirements engineering: "Validation denotes checking whether inputs, performed activities, and created outputs (requirements artefacts) of the requirements engineering core activities fulfil defined quality criteria."
- Validation of requirements is "… the process of creating a logical structure of specifications that is complete, actionable, and directly traceable to the highest-level system requirements" and verification is "… the complementary process that demonstrates complete compliance of the system and its subsystems with the previously validated specifications". (Marchant, 2010)
- As illustrated in Figure 1, Michael, et al. (2011) use the definitions that verification refers to activities that ensure the product is built correctly and validation refers to activities that ensure the right product is built in accordance with customer expectations.

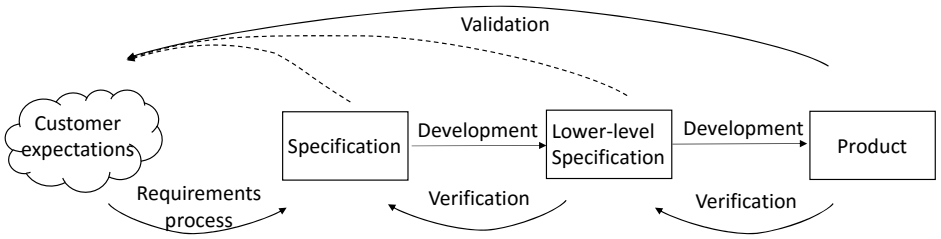

Figure 1. A continuous validation and verification process—from Michael et al (2011).

- As illustrated in Figure 2, taken from the INCOSE SE HB, 2015, requirements, design, and the final product are the subject of verification activities at each level to ensure each meets the requirements that drove the development of the work product.

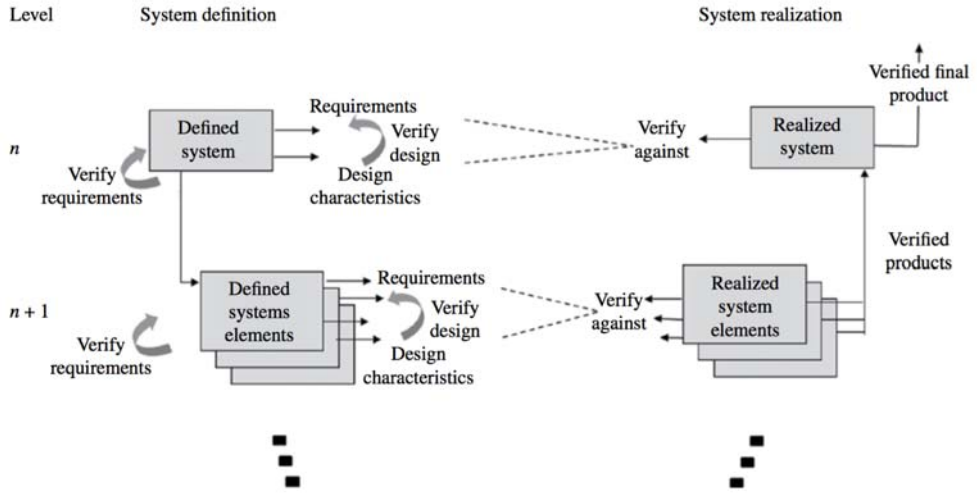

FIGURE 4.15 Verification level per level. Reprinted with permission from Alain Faisandier. All other rights reserved.

Figure 2. Verification performed incrementally at each level—from INCOSE SE HB, 2015.

- *In CMMI® for Development, Version 1.3 (CMMI® (Capability Maturity Model® Integration Software Engineering Institute (SEI)) CMMI® refers to verification and validation as process areas.*

  "Verification: The purpose of Verification is to ensure that selected work products meet their specified requirements. Verification includes verification of the product and intermediate work products against all selected requirements, including customer, product, and product component requirements. Verification is inherently an incremental process because it occurs throughout the development of the product and work products, beginning with verification of requirements, progressing through the verification of evolving work products, and culminating in the verification of the completed product.

  Validation: The purpose of Validation is to demonstrate that a product or product component fulfills its intended use when placed in its intended environment. Validation activities can be applied to all aspects of the product in any of its intended environments, such as operation, training, manufacturing, maintenance, and support services. The work products (e.g., requirements, designs, prototypes) should be selected on the basis of which are the best predictors of how well the product and product component will

satisfy end user needs and thus validation is performed early (concept/exploration phases) and incrementally throughout the product lifecycle (including transition to operations and sustainment).

Validation demonstrates that the product, as provided, will fulfill its intended use; whereas, verification addresses whether the work product properly reflects the specified requirements. In other words, verification ensures that "you built it right"; whereas, validation ensures that "you built the right thing." Validation activities use approaches similar to verification (e.g., test, analysis, inspection, demonstration, simulation). Often, the end users and other relevant stakeholders are involved in the validation activities. Both validation and verification activities often run concurrently and can use portions of the same environment."

- *From the Project Management Institute (PMI), Project Management Book of Knowledge (PMBOK), verification and validation are defined as:*

  "Verification: The evaluation of whether or not ac product, service, or system complies with a regulation, requirement, specification, or imposed condition.

  Validation: The assurance that a product, service, or system meets the needs of the customer and other identified stakeholders. The process of formalizing acceptance of the completed project deliverables." Validation "increases the chance of the final project, service, or result acceptance by validating each deliverable." Validation "assumes verified deliverable, i.e, "completed project deliverables that have been checked and confirmed for correctness through the Control Quality process."

  It is interesting to note that, from a process standpoint, the PMBOK addresses verification and validation as part of "Quality". Where Quality is defined as: "The degree to which a set of inherent characteristics fulfills requirements." The Control Quality process is defined as: "The process of monitoring and recording results of executing the quality activities to assess performance and recommend changes." The Control Quality process includes: "*Validating* that project deliverables and work meet the requirements specified by key stakeholders necessary for final acceptance and *verifying* that the delivered output will meet the requirements." "Control Quality is primarily concerned with the correctness of the deliverables and meeting the quality requirements specified for the deliverables."

- *In PMI's Requirements Management – A practice Guide, they refer specifically to requirement verification and requirement validation:*

  "Requirement verification: the process of reviewing requirements to ensure the requirements are constructed properly and are error free." Per the requirement verification process, requirements are compared to a set of requirement quality characteristics which serve as a guideline for writing high quality requirements.

  Requirement validation: the process used to evaluate that all requirements accurately reflect the intent of the stakeholder, thereby ensuring requirements meet stakeholder expectations."

From the above treatment of the terms verification and validation in relevant standards and literature on systems engineering, software engineering and project management, while there are some differences, there is also a common pattern. While the generic use of the words has common interpretations, without including a modifier as to what you are verifying or validating and making it clear whether or not you are referring to an activity or process, the terms can easily be misunderstood. By including the modifier, the 'context' of the word usage is clearer.

Proposed Definitions for Verification and Validation

In this section, we propose definitions for the terms "verification" and "validation" in the context of what the terms are referring to: requirements, design, or the system.

As stated earlier and illustrated by the lack of agreement in their definitions, while the terms "verification" and "validation" are commonly used, the true meaning of the concepts represented in each are often misunderstood and the terms are often used interchangeably without making clear the context in which they are used - resulting in ambiguity. 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 3.

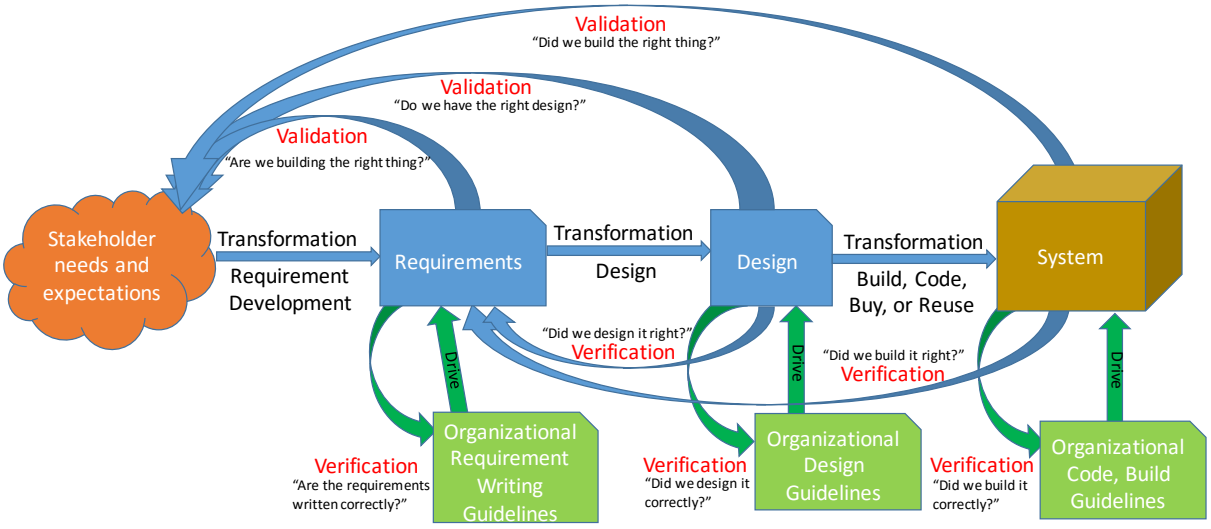

Figure 3. Verification and validation are the processes of confirming that systems engineering artifacts generated during the transformation processes are acceptable.

The concepts of verification and validation are very different depending on the modifier. When using these terms, it should be clear as to which context is intended.

Requirement Verification and Validation Defined

Context example: As shown in Figure 3, a requirement set results from a formal transformation of stakeholder needs and expectations. Correspondingly, design is a result of formal transformation of the requirement set in to an agree-to design, and a system is a formal transformation of the design into that system.

The process of creating a requirement set involves:

- analyzing stakeholder needs and expectations to obtain the necessary elements to be included in the requirement set;
- selecting a format for the requirement expression and an organization of the requirement set;
- identifying the characteristics of the desired result against the organizational guidelines and rules by which the requirement statements and requirement set is to be written, and
- transforming the stakeholder needs and expectations into a set of requirements that

unambiguously communicates these stakeholder needs and expectations to the design organization.

In this context, **Requirement Verification** confirms, by inspection, that the requirements contain the necessary elements and possess the characteristics of a well-formed requirement, and that the requirement set conforms to the rules set forth in the organization's requirement development guidelines. **Requirement Validation** confirms, by inspection and analysis, that the resulting requirement set meets the intent of the stakeholder needs from which the requirements and requirement set were decomposed or derived. Thus, the requirement statements and the requirement set are confirmed by both verification and validation activities.

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

- *Requirement Verification*: 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, guidelines, rules, and checklists?".
- *Requirement Validation*: confirmation that the requirements and requirement set 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 requirements and requirement set is communicating. "Does the requirements and requirements set clearly and correctly communicate the stakeholder expectations and needs?" "Are we doing the right things?" or "Are we building the right thing [as defined by the requirement set]?"

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

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" as defined later.

Design Verification and Validation Defined

Figure 3 also carries these concepts forward to design and realization of the system under development.

Once the requirements set is baselined, the requirements are transformed into a design of the system. Most organizations have a set of "design guidelines" or "golden rules" that guide the design process. These 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 system design is complete, there is usually a gate review or series of reviews 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 requirement set 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 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 the gate reviews, the design team "pushes back" on requirements that proved difficult to meet or were deemed not feasible for reasons of cost, schedule, and/or technology. This results in proposed changes to the requirements, which are submitted to the configuration control authority for the system for approval. When this happens, not only do the requirements need to be changed, but also the stakeholder expectations and needs from which the requirement were 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 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 in terms of a product life cycle context.

- *Design Verification*: 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 requirement set?" "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? 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?"

System Verification and Validation Defined

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, quality control, and branding requirements for 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 process represents a set of activities (test, demonstration, inspection, analysis) that cumulate with one or more gate reviews associated with the qualification and 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 system reflects the baselined requirements have been meet. From a contracting perspective, the baselined requirements are a type of contract and are legally binding.

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 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 in terms of a product life cycle context.

- *System Verification*: a process done after design and build or coding, ensuring 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*: a process that occurs after *system verification* that confirms 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 are directly related to the contractual obligation concept for a requirement statement and set of requirements. It is through these process 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 4.

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. Next, 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 earlier. Once the system requirements are baselined, design results in a system architecture in which the subsystems are defined. The design at this level is baselined via the *design verification* and *design validation* as discussed earlier.

For each subsystem, the above cycle is repeated with the definition of stakeholder subsystem needs, expectations, and stakeholder requirements, and then transformed into subsystem requirements, which are baselined via *requirements verification* and *requirements validation*. Once the subsystem requirements are baselined, design results in a subsystem architecture in which the units/components are defined. This design is baselined via *design verificatio*n and

design validation. This process repeats until the organization makes a buy, build, code, or reuse decision.

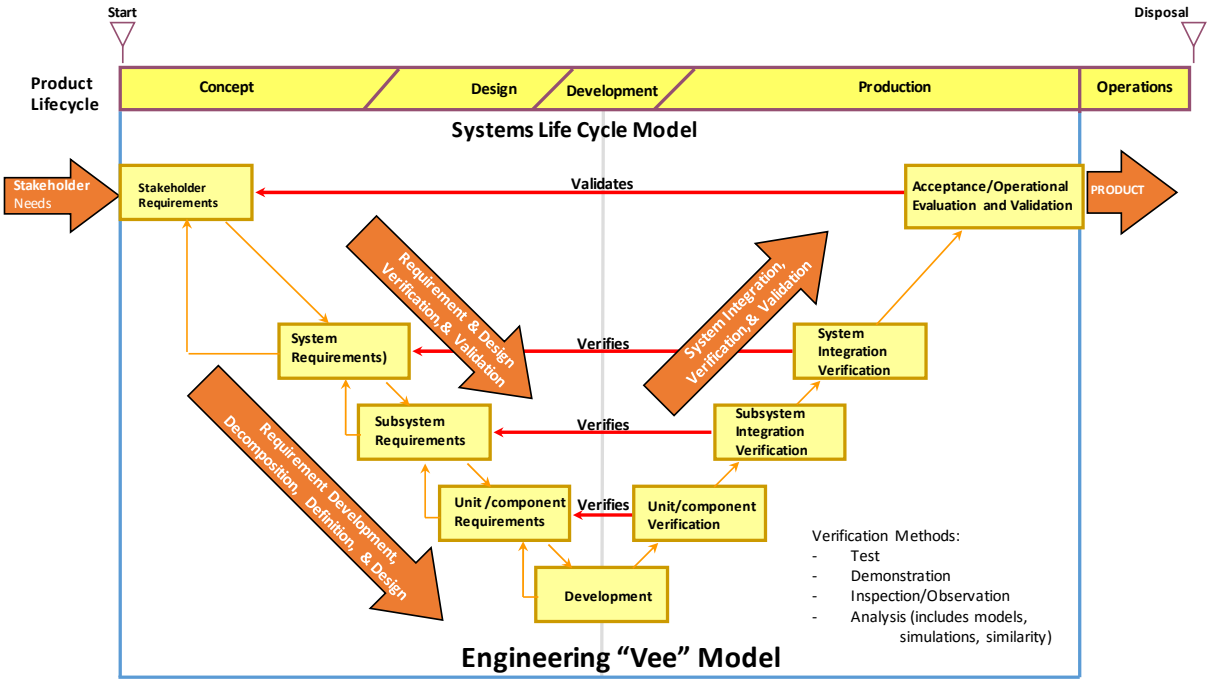

Figure 4: Verification and Validation and the Systems Engineering "V" Model

System integration, *system verificatio*n, and *system validatio*n (IV&V) occur bottom-up as shown on the right side of the SE "V" as shown in Figure 4.

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. Once the *subsystem verification* and *subsystem validation* activities are complete, the subsystems are integrated together and then *system verification* activities are completed. In the end, proof will be documented that can be evaluated by the customer to determine *system verification* activities have been completed successfully showing that the stakeholder requirements have been met (both organizational/people requirements as well as the technical requirements).

Following *system verification* activities, *system validation* activities are performed. This could be done in the form of acceptance and/or operations evaluation and validation activities. In the end, proof will be documented that can be evaluated by the customer to determine whether *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.

Conclusions

In general, *verification* refers to the basics (structure) of the item (requirements, design, system) 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 coding or manufacturing of the system. *Validation* goes beyond the basics (structure) to how well the item communicates or addresses

stakeholder needs and expectations while operating in the intended operational environment.

As shown in this paper, while these terms are commonly used, the true meaning of the concepts represented in each are often misunderstood and the terms are often used interchangeably without making clear the context in which they are used - resulting in ambiguity. This paper addressed the use of the terms verification and validation and presented their various meanings, particularly as stated in the relevant standards and literature, as well as everyday usage. In particular, it was identified that both terms are very ambiguous as to their true meaning unless a modifier is included in front of the word clearly indicating what the context in which term is referring to, specifically: requirements, the design, or to the system under development. The meaning is also ambiguous unless it is clear whether the terms are used to refer to an activity as part of a process or to an outcome of that process.

To help resolve these ambiguities, the authors provided illustrations and descriptions showing how the concepts of verification and validation are used throughout the product development lifecycle. Based on this description, the point was clearly shown that the meaning of these terms changes depending on context and where you are in the system development lifecycle. Specifically, it was shown that by including a modifier with each term that clearly indicates the context in which the terms are used, the reasons for ambiguity and misunderstanding are removed. The authors provided definitions of the terms in context to what they were referring to (requirements, design, or system) to disambiguate the use of each term.

A final conclusion can be made concerning the overall concepts of verification and validation. As a systems engineering best practice, the activities of verification and validation need to be done continuously throughout the system development lifecycles as shown in Figure 3. First the focus is on the requirements, then the focus shifts to the design, and finally the focus shifts to system verification and validation. With the increased use of language-based modeling in systems engineering, the systems engineer has the ability to do continuous verification and validation as the model is build and matured. This feature of language-based modeling allows changes to be evaluated as they are made to identify any inconsistencies caused by the change as well as to verify and validate the system when the change is made.

References

ANSI/EIA-632-1998, 1999. EIA Standard—*Processes for Engineering a System*, Arlington, V.A.: Electronic Industries Association.

ANSI/PMI 99-001-2013, 2013, *A Guide to the Project Management Body of Knowledge* (PMBOK Guide), Fifth Edition, PMI.

Boehm, B.W., 1984. "Verifying and Validating Software Requirements and Design Specifications", *IEEE Software*, Vol. 1, No. 1, pp. 75–88.

Davis, A.M., 1993, *Software Requirements: Objects, Functions and States* (Revised Edition), Engelwood Cliffs, NJ: Prentice Hall International.

Dzida, W. and R. Freitag, 1998, "Making Use of Scenarios for Validating Analysis and Design", *IEEE Transactions on Software Engineering*, Vol. 24, No. 12, pp. 1182–1196.

Engel, A., 2010, *Verification, Validation, and Testing of Engineered Systems*, Hoboken, NJ (US): Wiley.

Grady, J.O., 1997, *System Validation and Verificatio*n, Boca Raton, FL: CRC Press.

IEEE Computer Society, IEEE-STD-1012-2012 - I*EEE Standard for System and Software Verification and Validation*, New York: IEEE Computer Society, 2012.

INCOSE-TP-2003-002-04, 2015, *Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities*. Version 4, Edited by Walden, David D., et al.

ISO/IEC 15288, ISO/IEC 15288:2015, 2015, *Systems Engineering—System Life Cycle Processes*.

ISO/IEC/IEEE 29148:2011, ISO/IEC/IEEE 29148 *Systems and Software Engineering—Life Cycle Processes—Requirements Engineering*.

Kotonya, G., and I. Sommerville, 2000, *Requirements Engineering: Processes and Techniques*, Chichester, England: John Wiley & Sons.

Leffingwell, D., and D. Widrig, 2000, *Managing Software Requirements: A Unified Approach*, Reading, MA: Addison-Wesley.

Loucopoulos, P., and V. Karakostas, 1995, *System Requirements Engineering*, New York: McGraw-Hill.

Marchant, A.B., 2010, "Obstacles to the Flow of Requirements Verification", *Systems Engineering*, Vol. 13, No. 1.

Michael, J.B., D. Drusinsky, T.W. Otani, and M-T. Shing, 2011, "Verification and Validation for Trustworthy Software Systems", *IEEE Software*, November/December 2011, pp. 86–92.

MIL-STD-499B, 1994, *Military Standard—Systems Engineering*—Draft, Washington D.C.: United States of America Department of Defence.

Pohl, K., 2010, *Requirements Engineering: Fundamentals, Principles, and Techniques*, Heidelberg: Springer-Verlag.

PMI, 2016, *Requirements Management – A Practice Guide*, PMI.

Pohl, K., and C. Rupp, 2011, *Requirements Engineering Fundamentals*, Santa Barbara CA: Rocky Nook.

Rakitin, S.R., 2001, *Software Verification and Validation for Practitioners and Managers*, Norwood, MA: Artech House.

Robertson S. and J. Robertson, 2013, *Mastering the Requirements Proces*s, Harlow, England: Addison-Wesley.

Ryan, M.J., Wheatcraft, L.S.; Dick, J.; and Zinni, R., 2015. *"An Improved Taxonomy for Definitions Associated with a Requirement Expression"*, *Systems Engineering and Test and Evaluation Conference SETE 2014*, Adelaide, Australia, 28-30 April 2014 and INCOSE IS 2015, Seattle, WA 12-17 July 2015.

Software Engineering Institute (SEI), 2010, *CMMI® for Development*, Version 1.3 CMU/SEI-2010-TR-033 ESC-TR-2010-033, November 2010.

Wheatcraft, L.S., 2012, *Thinking Ahead to Verification and Validation*, presented at NASA's PM Challenge 2012, February 2012, Orlando, Florida.

Biographies

Dr. Mike Ryan is the Director of the Capability Systems Centre, University of New South Wales, Canberra, at the Australian Defence Force Academy. He holds Bachelor, Masters, and Doctor of Philosophy degrees in electrical engineering as well as a Graduate Diploma in Management Studies. For the first seventeen years of his career he held a number of communications engineering, systems engineering, project management, and management positions. Since joining UNSW in 1998, he lectures and regularly consults in a range of subjects including communications and information systems, systems engineering, requirements engineering, and project management. He is the conference chair of two annual international conferences, he is the editor-in-chief of the Journal of Battlefield Technology, and is chair of the Requirements Working Group in the International Council on Systems Engineering (INCOSE). He is the author or co-author of eleven books, three book chapters, and over 160 technical papers and reports.

Lou Wheatcraft is a senior instructor/consultant for Requirements Experts (RE) who educates organizations on the importance of writing good requirements and helps them implement Requirement Development and Management (RD&M) processes based on industry best practices. Lou has taught over 180 requirement seminars over the last 15 years. Lou works with both government and industry clients. Lou has spoken at Project Management Institute (PMI) chapter meetings, INCOSE conferences and chapter meetings. Lou has had published and presented a multitude of papers on requirement RD&M topics for NASA's PM Challenge, INCOSE, INCOSE INSIGHT Magazine, and Crosstalk Magazine. Lou is a member of INCOSE, co-chair of the INCOSE Requirements Working Group, a member of PMI, the Software Engineering Institute (SEI), the World Futures Society, and the National Honor Society of Pi Alpha Alpha. Lou has a BS degree in Electrical Engineering from Oklahoma State University, an MA degree in Computer Information Systems from the University of Houston – Clear Lake, an MS degree in Environmental Management from the University of Houston – Clear Lake, and has completed the course work for an MS degree in Studies of the Future from the University of Houston – Clear Lake. Lou is the primary contributor to RE's blog on requirements best practices. The blog can be assessed at: http://www.reqexperts.com/blog.