Vol. 3 No. 2 October 2015 ISSN: 2321 – 788X

SOFTWARE VERIFICATION & VALIDATION STRATEGY AND IMPLEMENTATION

Ramkumar Soundarapandian
Manager - Ecommerce, Automation (AI) & Cloud Technology
Capgemini Technology Services India Limited

Abstract

Verification and validation in software development are the necessary processes through which software products are made to meet requirements and behave as expected. While the process of validation confirms that the final product meets users' expectations and functions in the manner expected of it, verification makes sure that the program is developed rightly based on the design and technical specifications. This includes the different strategies and techniques of V&V, such as inspections, code reviews, UAT, static and dynamic testing, and inspections. When properly executed, V&V suppresses risks, quality is superior, and dependability and security are guaranteed well before software release. The study closely examines the two important roles of validation and verification in software engineering. While verification would use methods like code reviews and static testing to ensure the software conforms to specifications, validation involves user acceptability and dynamic testing to test if the user requirements are present in the software. The study puts forth how integrating the two approaches will lead to improvement in software quality and reduction of risk.

Key words: *Verification and Validation (V&V), User Acceptance Testing (UAT), Software, Dynamic Testing, Inspection*

1. Introduction

Software verification and validation are important processes to ensure a software product meets its requirements for its intended purpose. Verification-software is produced correctly according to design and technical requirements in mind-is one of the major components of this strategy. The other major component involves validation that ensures the completed product meets user demands and acts as expected when exposed to the real world. These are implemented through code reviews, inspections, static and dynamic testing, and UAT, among other activities. The aim is to detect and fix defects as early as possible in the development life cycle, reduce risks, enhance quality, ensure reliability and safety, and then deploy the system.

Figure 1 Verification and Validation

Software validation and verification have also been acknowledged as major areas within software engineering that contribute much to software quality. It is also taught within computer engineering curricula and in university computer science software courses. A verification process will involve proving that the program is developed according to particular specifications, behaves as expected, and also complete. Alongside the correctness of the behavior, the aim is to achieve proper fulfilment of consistency and accuracy of program translations. The process used in determining the correctness of the software at all phases of development is known as validation.

This has set a strong framework in developing high-quality software. The comprehensive system engineering approach used in verification and validation as an effective standard has been one of the keys to ensuring that software quality has been developed at every level of the software life cycle. Consequently, it requires that in software engineering, the program is first validated at each stage of the development lifecycle and then validated again in case it is moved. Verification and validation of the software can be defined as the process of software comparison with its needs. In fact, each project should confirm and approve the software that it is developing. The end result of verification and validation is that the program will be used appropriately. Many software problems have been very costly, embarrassing, and sometimes even hazardous. It has been realized that verification of accuracy and improvement of the quality of the overall program before it is ever used is the best defense against such problems. The correctness of a software program has been one major factor in software development. The most prominent methods employed to do so have been SQA, Verification & Validation, and Testing.

Verification and validation involve the extensive testing of the program and analysis in order to make sure that all the requirements are being met out of it, and what level of dependability and quality there is present in it. Verification and Validation has been considered a system engineering topic for assessing software within a system. It involves the use of a structured method to test the software against all system functions as well as against the user, hardware, and software interfaces, whereas system engineering does not. Moreover, they have been complementing one another rather than replacing one another. Sufficient verification and validation may guarantee the quality of the program as well as the processes used in the development and testing of the software, which will ultimately enable achieving the business objectives. Merely providing appropriate verification without any validation, on the other hand, might not guarantee the best software solutions. However, a combination of the two is needed.

These are the two important facets of software quality control. Additionally, the verification offers responses to queries about whether software has been developed in an appropriate manner, and the validating offers responses whether software has been generated in an adequate manner. Precision is indicated by verification, and value associated with the finished or final product is indicated by validation. Verification and validation are important stages that are used in many processes across a variety of industries. Verification and validation's primary goal has been to provide evaluations of the software's capacity to meet both user and software requirements. While validation demonstrates ensuring that the program meets client expectations, verification assesses whether the program complies with its standards. The verification and validation processes offer an assessment of the software products and the processes involved in the software life cycle. This type of evaluation demonstrates whether the software and system requirements have been accurate, full, consistent, testable, and precise.

1.1. The Role of Verification & Validation in Software System Development Life Cycle

The role of verification and validation for every product should be evaluated on a project-by-project basis. Complexity, limitations, and product criticality will all have an impact on this kind of evaluation. Verification and validation function's general goal has been to make sure the product satisfies user needs. As a result, every need and specification for a product should be focused on a distinct verification and validation task. The methodologies of Verification and Validation determined will zero in on practical as well as performance segments connected with particulars and prerequisites concerning security, viability, movability, wellbeing, convenience, as well as workableness. Although these aspects are highly significant for many systems, they won't be covered here in order to limit the scope of such a module. The goal of validation is also to "build the right system," just as verification is to "build the system right." As a result, validation looks at procedures to make sure the system is making the right decisions, while verification looks at issues like verifying that the system's knowledge has been sufficiently specified. Validation and verification are crucial to the development and deployment of case-based systems. There could be mistakes in the case representations if the system isn't validated. If the system hasn't undergone validation, it may not be producing decisions with the necessary level of quality.

This section will cover verification and validation in the software development life cycle. Specifically, what is checked in relation to every stage of the life cycle and who is accomplishing this checking. Additionally, the methods used to carry out the checking. Additionally, the software verification and validation process makes use of testing, reviews, and analytical techniques to ascertain whether the software system and its intermediate products meet the requirements. These requirements cover both functional skills and quality traits. Verification and validation's primary responsibility has also been to manage project risks by keeping an eye out for and identifying faults made during the development and maintenance phases. The Verification & Validation contractor has been tasked with finding mistakes as early as feasible and tracking progress toward resolution because it is impractical to detect and address every error early in a project's life cycle. Additionally, verification based on review or non-executable ways guarantees that the system (people, hardware, software, and documentation) complies with organizational processes and standards. Moreover, the process of physical validation verifies that the system is functioning according to a plan by carrying out a number of tests that may be evaluated and monitored.

1.2. Verification And Validation in Software Engineering

Verification

Verification is the most common way of ensuring software fills in as planned and is sans blunder. It is the system to ensure the created item is right or not. It verifies whether the made item fulfils our guidelines. Static testing, to lay it out plainly, is verification.

Static Testing

Verification testing, commonly referred to as static testing, is the process of determining whether or not we are creating the correct product and whether or not our program satisfies the needs of the consumer. These are a few of the tasks that go into verification.

- Inspections
- Reviews
- Walkthroughs
- Desk-checking

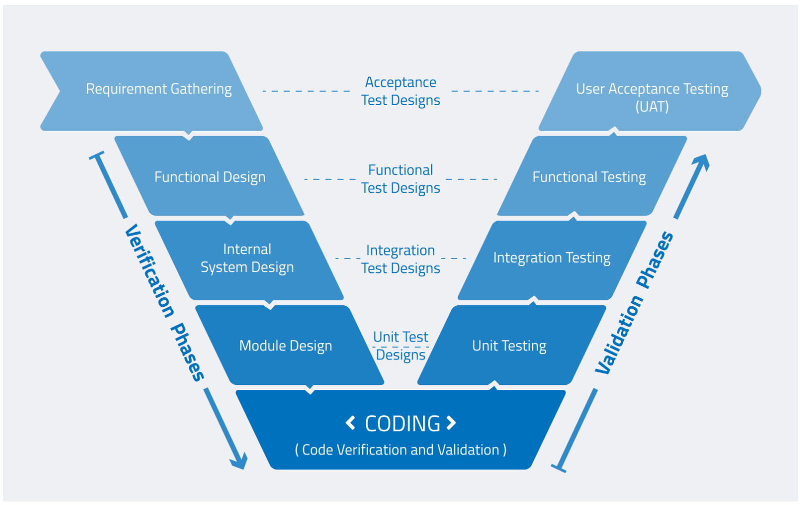

Figure 2 Verification and Validation Model

Validation

Validation is the method involved with deciding if the software item fulfils exclusive expectations, or on the other hand assuming it is adequate. The strategy confirms the item's validation, or that the item we are building is the right one. It's a certification of the genuine and expected things. Dynamic testing is the normal term utilized for validation.

Dynamic Testing

Validation testing, sometimes referred to as dynamic testing, involves determining whether or not we have produced the product correctly and learning about the client's business needs. These are a few of the tasks that come with validation.

- Black Box Testing
- White Box Testing
- Unit Testing
- Integration Testing

2. LITERATURE REVIEW

Beyer, D. (2013) outlines the second edition of this extensive assessment of completely automatic software program verifiers, the 2nd International Competition on Software Verification (SV-COMP 2013). The presented findings, as well as their availability and participation, constitute the state-of-the-art in automatic software verification as of 2012. Reachability and memory security are among the characteristics of whole numbers, stack information structures, digit vector operations, and simultaneousness that are uncovered by the 2 315 C projects that make up the benchmark set of verification errands. Yet again the opposition is set up as a TACAS satellite occasion.

Keremoglu, M. E. (2011) introduces CPAchecker, a framework and tool designed to make adding new verification components simple. Each abstract domain implements the customizable program analysis (CPA) interface along with the appropriate operations. The reachability study of any combination of currently available CPAs can be carried out by adjusting the core algorithm. We want to speed up the process of translating a verification notion into actual experimental results in software verification, which now requires a significant amount of work. With this adaptable and simple-to-extend platform, we believe that researchers will find it straightforward and productive to implement novel verification ideas and methods, and that it will advance the discipline by simplifying the execution of real-world experiments. The Java program can be used as an Eclipse plug-in or as a command-line tool. CPAs are implemented for a number of abstract domains by CPAchecker. We compare our current tool with existing state-of-the-art model checkers and assess its effectiveness on software-verification benchmarks from the literature. CPAchecker is a publicly accessible, open-source toolbox.

Griggio, A. (2012) provide the results of IC3's initial inquiry in the context of software verification. Our initial step is to generalize it from SAT to Satisfiability Modulo Theories (SMT), which allows us to analyze programs directly after they have been encoded as symbolic transition systems. Secondly, we modify the "linear" search method of IC3 to a tree-like search in order to take advantage of the Control-Flow Graph (CFG) of the program under analysis. Third, we place this method within the framework of lazy abstraction using interpolants, and when appropriate, we optimize it by employing interpolants taken from proofs.

Hsieh, W. H. (2010) make use of the information scientist's created knowledge domain visualization and intellectual structure construction methodologies to make it easier to comprehend the central ideas of this research area. Based on the analysis, it is possible to separate the research themes related to software validation into two main categories: the first group addresses systems that involve both hardware and software issues, such as parallel and real-time systems, while the second group addresses issues related to model checking, such as automata theory and temporal logic.

Raza, B. (2010) Assess the cost-effectiveness of the European Cooperation for Space Standardization (ECSS) standards now in use, look for ways to streamline the process without sacrificing quality, and determine whether their verification and validation (V&V) operations can be made more efficient. This paper reports on the results of two industrial case studies of European space industry companies applying ECSS standards in various V&V activities. The sections that follow in this article put the emphasis on the case companies themselves and how their use of the ECSS standards affects their operation and ultimately how their V&V work can be more streamlined.

3. IMPLEMENTATION OF VERIFICATION AND VALIDATION

The following section describes how to plan for the implementation of V&V in the research project, including some very specific details on selected tools, resources allotted, processes incorporated, and continuously updated documentation. A methodical, systematic strategy will have to be established in order to ensure that the study findings are correct and can be relied upon.

3.1. Setting Up the Environment

3.1.1. Tool Selection

A successful execution of a V&V activity in this research study is much dependent on proper tool selection. The procedure includes the following:

- **Identification of Requirements:** study of the particular needs of research which may be formal verification techniques, dynamic testing, or static code analysis.
- **Comparative Analysis:** The investigation and comparison of these tools concerning their features, user experience, and integrative potential. Example: Unit testing would use JUnit while static analysis would make use of SonarQube-type tools.
- **Conducting Trials:** putting experimental versions of chosen tools into use to ensure they are appropriate for the study. This aids in comprehending how they function in the context of research.
- **Cost Evaluation:** This includes financial consideration-training, licenses-to see whether this will fall within the budget set for the research.

3.1.2. Environment Configuration

Configuration of the environment plays an important role in the successful execution of V&V:

- **Tool Installation:** Installation of the tool selected on required platforms, like test and development environment.
- **Integration:** Set up tools to work with the existing systems. For instance, hook up the testing framework to the study's CI server.
- **Customization:** modifying, for instance, the parameters of tools or creating rules for code analysis, in order to meet some specific research requirements.
- **Setup Validation:** This includes preliminary testing designed to provide confidence that the tools and environment are set up correctly and working as expected.

3.2. Resource Allocation

3.2.1. Team Roles and Responsibilities

Clearly, defining the roles and responsibilities clears the way for effective handling of the V&V activities:

- **Verification Engineers:** Will be responsible to execute static analysis, code reviews, and adherence to research standards
- **Validation Engineers:** Responsible for executing validation tests in order to verify whether the study results meet the required standards and specifications.

- **Test Managers:** V&V activity planning and execution are in line with the goals and timing of the research.
- **Developers:** Cooperate with V&V teams with the focus on problem rectification as identified through the processes of validation and verification.
- **QA Specialists:** Ensure, in the duration of the research work, that quality standards are observed and adhered to, and also procedures related to V&V are observed.

3.2.2. Training and Skill Development

The following are some of the key enablers in the successful application of V&V: good training and skill development.

- **Tool Training:** This is quite important in helping the team fully realize the full features and functionalities of the selected tools.
- **Process Training:** Train the group about V&V approaches and best practices that apply to the study overall, specific steps and methods.
- **Continuing Development:** Webinars, workshops, and courses for ongoing learning opportunities will keep the team current about state-of-the-art V&V procedures.
- **Knowledge Sharing:** Encourage the team to share knowledge and experiences in order to further improve team performance or in finding solutions to joint problems.

3.3. Process Integration

3.3.1. Integration with Development Processes

Integrating V&V processes with research activities ensures consistency and effectiveness:

- **Early Integration**: Start V&V work early in the research cycle to find problems early and fix them. The integrity of the research is preserved by this proactive strategy.
- **Continuous Integration**: By automating testing and feedback processes, V&V may be integrated into the CI/CD pipeline to ensure ongoing validation of research findings.
- **Feedback Mechanisms**: Provide researchers with quick feedback through well-defined communication routes, which will enable expeditious remediation of issues that are discovered.

3.3.2. Coordination with Other Teams

The effective coordination with other research teams enhances the process of V&V.

- **Researcher collaboration:** This would involve working with the researchers to understand what they're trying to achieve and how this work of V&V can complement their objectives.
- **Coordination of Project Management:** The integration of V&V into overall research objectives through coordination with project timelines and milestones.
- **Timing of Stakeholder Engagement:** Engage the project sponsor and subject matter experts as stakeholders at the appropriate juncture in the project to ensure that the V&V undertakings meet their needs and expectations.

3.4. Documentation

3.4.1. Verification and Validation Plans

The documentation of V&V plans is influential in tracking and managing the activities of V&V.

- **Verification Plan:** Elaborate on a plan that describes the research tools, techniques, and the processes of verification. The plan should list the tasks with their objectives accordingly.
- **Validation Plan:** Based on the above objectives and requirements, the research will design a validation plan in detail, presenting test methodologies, test cases, and acceptance criteria.
- **Schedule:** An estimate of the timeline for conducting the V&V tasks, including the identification of major milestones and dates that will help in tracking to assure timely completion.

3.4.2. Reports and Records

Good recordkeeping will contribute to ensuring that all activities involving V&V are transparent and accountable:

- **Test Reports:** Document the test case results, defect logs, and solutions owing to the testing activities. Such papers provide insight into the quality and reliability of research output.
- **Verification Records:** Record the verification activities to control the compliance of research measures, including code review summaries and static analysis results.
- **Log issue:** Record and log all the issues identified through V&V. The issues include severity information, status, and resolution plans that help in early problem resolution and effective problem management regarding the research issue.
- **Audit Trails:** Provide for full audit trails of all V&V activities to enable the standardization and reproducibility of research.

4. Best Practices for the Verification and Validation Process

This section discusses the best practices used to ensure that the V&V is successful and the project is a success. The comprehensive document review, impact analysis of the iterative build, strategy and documentation for the V&V activities in the current build, improved and enhanced testing methodology, testing procedure review, and comprehensive report production are the tasks that need to be completed. The ensuing paragraphs will provide descriptions of each best practice.

- **Document Review:** One of the first things to do after the design team releases the build for the V&V activities is document review. Comprehensive information about the requirements, design, and code implementation is provided by the document review. The test procedures are established in accordance with the review.
- **Impact Analysis:** In the case of iterative builds, the impact of modifications to the test method approach is examined by reviewing the new build and comparing it to the prior version.
- **Test Strategy and Documentation:** The test strategy for the test procedure is developed, updated, or modified with the aid of the effect analysis on the build. During the build activity, the effect analysis and plan are recorded for future reference.

- **V&V activities for the iterative build:** The different tasks involved in the iterative construction are also documented in the text. Depending on the functionality that a given build implements, not all of the activities will be relevant for that build.
- **Enhanced and improved methodologies:** Static and dynamic software testing is one of the testing methodologies. Static, dynamic, and random testing are all included in hardware software integration testing. Hardware Software Integration testing can be done in a non-real time manner depending on the functionality.
- **Review of the Testing Procedure:** This is done to ensure that the test protocol is accurate and competitive prior to the tests being run.
- **Execution and Report Generation:** This is the final task that needs to be completed. The findings are examined, and test reports are produced with details on the requirements that were met, any restrictions, and the observations made throughout the testing, depending on the pass-fail criteria.

These activities are carried out for the V&V process activity in a sequential way ensuring the effectiveness of the process and capturing the bug early in the process.

5. Conclusion

In software engineering, software verification and validation are crucial procedures that guarantee a product satisfies requirements and accomplishes goals. While validation provides assurance that the final product will be able to meet user expectations for its intended operation in real-life settings, verification checks whether software is being developed rightly to meet its set design and technical requirements. This study specifies that the verification process must be embedded into the process of software engineering along with validation. While validation ensures that software meets user needs and expectations, verification ensures that software is developed against its specifications. While incorporating these strategies, the organizations are greatly able to increase customer satisfaction, reduce chances of occurrences leading to errors, and hence enhance quality. Basically, a well-balanced approach toward both validation and verification will be needed as a means of generating reliable and effective software solutions.

References

1. Cimatti, A., & Griggio, A. (2012). Software model checking via IC3. In Computer Aided Verification: 24th International Conference, CAV 2012, Berkeley, CA, USA, July 7-13, 2012 Proceedings 24 (pp. 277-293). Springer Berlin Heidelberg.
2. Beyer, D., & Keremoglu, M. E. (2011). CPAchecker: A tool for configurable software verification. In Computer Aided Verification: 23rd International Conference, CAV 2011, Snowbird, UT, USA, July 14-20, 2011. Proceedings 23 (pp. 184-190). Springer Berlin Heidelberg.
3. Beyer, D. (2013). Second Competition on Software Verification: (Summary of SV-COMP 2013). In Tools and Algorithms for the Construction and Analysis of Systems: 19th International

Conference, TACAS 2013, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2013, Rome, Italy, March 16-24, 2013. Proceedings 19 (pp. 594-609). Springer Berlin Heidelberg.

4. Bérard, B., Bidoit, M., Finkel, A., Laroussinie, F., Petit, A., Petrucci, L., & Schnoebelen, P. (2013). Systems and software verification: model-checking techniques and tools. Springer Science & Business Media.
5. Bjarnason, E., Runeson, P., Borg, M., Unterkalmsteiner, M., Engström, E., Regnell, B., ... & Feldt, R. (2014). Challenges and practices in aligning requirements with verification and validation: a case study of six companies. Empirical software engineering, 19, 1809-1855.
6. Oberkampf, W. L., & Roy, C. J. (2010). Verification and validation in scientific computing. Cambridge university press.
7. Demirbilek, Z., & Rosati, J. (2011). Verification and Validation of the Coastal Modeling System. Report 1: Summary Report.
8. Bérard, B., Bidoit, M., Finkel, A., Laroussinie, F., Petit, A., Petrucci, L., & Schnoebelen, P. (2013). Systems and software verification: model-checking techniques and tools. Springer Science & Business Media.
9. Henzinger, T. A., Jhala, R., Majumdar, R., & Sutre, G. (2003). Software verification with BLAST. In Model Checking Software: 10th International SPIN Workshop Portland, OR, USA, May 9–10, 2003 Proceedings 10 (pp. 235-239). Springer Berlin Heidelberg.
10. Söderberg, C. A., Lambert, W., Kjellström, S., Wiegandt, A., Wulff, R. P., Månsson, C., ... & Emanuelsson, C. (2012). Detection of crosslinks within and between proteins by LC-MALDI-TOFTOF and the software FINDX to reduce the MSMS-data to acquire for validation. PLoS One, 7(6), e38927.
11. Feldt, R., Torkar, R., Ahmad, E., & Raza, B. (2010, April). Challenges with software verification and validation activities in the space industry. In 2010 third international conference on software testing, verification and validation (pp. 225-234). IEEE.
12. Chen, T. T., & Hsieh, W. H. (2010, December). Uncovering the main research themes of software validation. In 2010 International Conference on Computational Intelligence and Software Engineering (pp. 1-6). IEEE.
13. Sargent, R. G. (2010, December). Verification and validation of simulation models. In Proceedings of the 2010 winter simulation conference (pp. 166-183). IEEE.