Improvement in the V-Model

Ravi Shanker Yadav

Abstract—The V-model represents a software development process (also applicable to hardware development) which may be considered an extension of the waterfall model. Instead of moving down in a linear way, the process steps are bent upwards after the coding phase, to form the typical V shape. The V-Model demonstrates the relationship between each phase of the development life cycle and its associated phase of testing. The horizontal and vertical axes represents time or project completeness (left-to-right) and level of abstraction (coarsest-grain abstraction uppermost), respectively. This model depends on verification and validation phase Software Testing is the most important phase of the Software Development Life Cycle. On most software projects testing activities consume at least 30 percent of the project effort. On safety critical applications, software testing can consume between 50 to 80percent of project effort. Software testing is essential to ensure software quality. Schedule is always running tight during the software system development, thereafter reducing efforts of performing software testing management. In such a situation, improving software quality becomes an impossible mission It is our belief that software industry needs new approaches to promote software testing management. The article discussed the model that already existed, further excavates the parallelism between test stages and maintenance test stages and tries to propose a improved V model. This model makes the software testing pass through the each stage of software development cycle.

Index Terms— Acceptance, accuracy, compiling, debugging, model checking, performance, program validation, reliability, security, testing

1 Introduction

THE development process for a system is traditionally as a waterfall model where each step follows the next, as if in a waterfall. On current time everyone wants to a fully satisfied project in very few investments. During development and maintenance of such long lived software, requirements are analyzed, designed and code modules are developed, testing is planned and code is tested many times. Thus software development and maintenance services should ensure customer satisfaction. This calls for software developer to ensure the quality of development, implementation, Testing and as well as maintenance. Since the schedule of software Development is running tight, resulting in less effort for testing and maintenance. As testing directly links to quality of product, this demands that solution provider creates strong Testing and Maintenance Base for the technology solutions.

What should be done to enhance the software testing management? It does not imply that any of the steps in a process have to be completed, before next step starts, or that prior step will not have to be revisited later in development. It is just a useful model for seeing how each step works with each of the others.

We should have well techniques for testing supported by simple and clear model to be followed to avoid unnecessary ambiguity. This articles present two-dimensional approach for managing testing. Firstly we need a testing that incorporates testing into the entire software development life cycle. Secondly software testing management has to introduce the concept of software architecture to gradually enhance its software testing management. This paper discusses the simple V-model in detail and the improved mode .that is most important and most effective.

2 Software Testing Management

The software architecture is the key towards an efficient software testing management. We here briefly describe the architecture of the software in this section.

2.1 Software Architecture

The software architecture of a program or computing system is the structure or structures of the system, which comprise software elements, the externally visible properties of those elements, and their relationships among them.The term software architecture as defined by Jacobson is the set of models to be built, each having its characteristic or set of modeling notations and they presents conceptual view of the process adopted for software development. Software community is well familiar with requirement models such as Use case diagram, Object model which represent conceptual view of requirements and system to be built without implementation details.

The software architecture of a program or computing system is the structure or structures of the system, which comprise software components, the externally visible properties of those components and the relationships between them. The term also refers to documentation of a system's software architecture. Documenting software architecture facilitates communication between stakeholders, documents early decisions about high-level design, and allows reuse of design components and patterns between projects [1].The software system component consists of various elements of the system like Programs, System Utilities, System Services; one more definition is Architecture is the organizational structure of a system. Architecture can be recursively decomposed into parts that interact through interfaces, relationships that connect parts, and constraints for assembling parts. Parts that interact through interfaces include classes, components and subsystems.

2.2 Part of Software and Maintenance tests

1. Software testing is more than just error detection; Testing software is operating the software under controlled conditions, to (1) verify that it behaves "as specified"; (2) to detect errors, and (3) to validate that what has been specified is what the user actually wanted.
2. Verification is the checking or testing of items, including software, for conformance and consistency by evaluating the results against pre-specified requirements. [Verification: Are we building the system right?]
3. Error Detection: Testing should intentionally attempt to make things go wrong to determine if things happen when they shouldn't or things don't happen when they should.
4. Validation looks at the system correctness – i.e. is the process of checking that what has been specified is what the user actually wanted. [Validation: Are we building the right system?]
5. Software Testing as defined by Pressman is the process of executing a program or system with the intent of finding errors. [Myers79] [2] Or, it involves any activity aimed at evaluating an attribute or capability of a program or system and determining that it meets its required results. [Hetzel88] [3] The results are observed or recorded, and an evaluation is made of some aspect of the system or program. A good test case is a one that has a high probability of finding an as-yet undiscovered error and a successful test is the one that uncovers an as-yet undiscovered error. On the other hand the maintenance of the software can account for over 60 percent of all effort expended by a development organization and the percentage continues to rise as more software is produced [Han93][12]. Software maintenance is the modification of a software product after delivery to correct faults, to improve performance or other attributes or to adapt the product to a modified environment [10]. Thus as described by Pressman only 20percent of all maintenance work is fixing errors and remaining 80 percent is spent adapting existing systems to changes in their external environment, making enhancements requested by the users and reengineering an application for future use. For carrying out the software maintenance, certain software tests needs to be performed in order to enhance the maintainability and performance of the software. Thus software testing and software maintenance tests work together in achieving a good quality, highly reliable and efficient software. These strategies contribute towards the software testing management. Thus this paper defines different categories of software tests based on IEEE standard glossary of Software Engineering Terminology [4, 5, 6, 7, 8] and various categories of software maintenance tests. Their definitions are summarized as shown as following; In other words, validation checks to see if we are building what the customer wants/needs, and verification checks to see if we are building that system correctly. Both verification and validation are necessary, but different components of any testing activity. The definition of testing according to the ANSI/IEEE 1059 standard is that testing is the process of analyzing a software item to detect the differences between existing and required conditions (that is defects/errors/bugs) and to evaluate the features of the software item. Remember: The purpose of testing is verification, validation and error detection in order to find problems – and the purpose of finding those problems is to get them fixed.

4 PART OF SOFTWARE TESTS

1. **Unit Testing**: Starting from the bottom the first test level is "Component Test", sometimes called unit testing. It involves checking that each feature specified in the "Component Design" has been implemented in the component.it involves checking that each feature specified in the "component design"has been implemented in the component. It focuses on each component individually, ensuring that if functions properly as a unit. It heavily uses white box testing techniques, exercising specific paths in a module's control structure to ensure complete coverage and maximum error detection.

2. **Integration Testing**: It is most important test because here system tests with integrative method. It addresses assembling and integration of components to from a complete software package. It uses black box testing techniques to address issues related to dual problems of verification and program construction.
3. System Testing: Once the entire system has been built then it has to be tested against the "System Specification" to check if it delivers the features required. It is still developer focused, although specialist developers known as system testers are normally employed to do it.
In essence the system test is not about checking the individual parts of the design, but about checking the system as a whole .In effect it is one giant component.

System testing can involve a number of specialist types of test to see if the entire functional and non-functional requirements have been met. In addition to functional requirements these may include the following type of testing for the non-functional requirements:

: PERFORMANCE-Are the performance criteria met?
: VOLUME- Can large volumes of information be handled?
: STRESS- Can peak volumes of information be handled?
: DOCUMENTATION- Is the documentation usable for the system?
: ROBUSTNESS- Does the system remain stable under adverse circumstances?
There are many others, the needs for which are dictated by how the system is supposed to perform.

Testing conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements. It requires no knowledge of the inner design of the code or logic.

4. **Acceptance Testing:** Acceptance Testing checks the system against the "User Requirements". It is similar to systems testing that the whole system is checked but the important difference is the change in focus:
: Systems testing checks that the system that was specified has been delivered.
: Acceptance testing checks that the system delivers what was requested.
The customer and not the developer should always do acceptance testing. The customer knows what is required from the system to achieve value in the business and is the only person qualified to make that judgment. The forms of the tests may follow those in system testing, but at all times they are informed by the business needs.
Testing to verify a product meets customer specified requirements customer usually does this type of testing on a product that is developed externally.

4 PART OF SOFTWARE MAINTENANCE

1. **Set Testing**: A test case is a set of Conditions or variables under which a tester will determine whether an application of software system meets its specifications at the unit level.

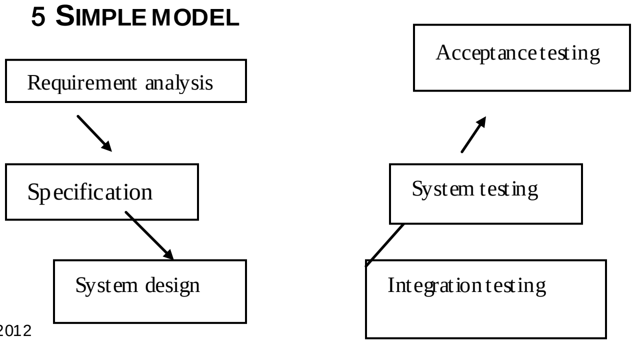
2. **Regression Testing**: With modern systems one person's system, become somebody else's component. It follows that all the above types of testing could be repeated at many levels in order to deliver the final value to the business. In fact every time a system is altered.It is a technique that detects spurious errors caused by software modifications or corrections.

3. **Compliance Testing**: It is use for security purpose that's played a specific role in the software development. Evaluates the presence and appropriate functioning of the security of the application to ensure the integrity and confidentiality of the data.

4. **Simulation Testing**: Acceptance Testing checks the system against the "User Requirements". It is similar to systems testing that the whole system is checked but the important difference is the change in focus:
: Systems testing checks that the system that was specified has been delivered.
: Acceptance testing checks that the system delivers what was requested.
The customer and not the developer should always do acceptance testing. The customer knows what is required from the system to achieve value in the business and is the only person qualified to make that judgment. The forms of the tests may follow those in system testing, but at all times they are informed by the business needs.
The testing and/or simulation of system assets in the physical and operational environment in which they are expected to perform.

5. **Release Test**: Even if a system meets all its requirements, there is still a case to be answered that it will benefit the business. The linking of" business case" to Release testing is looser than the others, but is still important.
Release testing is about seeing if the new or changed system will work in the existing business environment. Mainly this means the technical environment, and checks concerns such as:
: Does it affect any other systems running on the hardware?
: Is it compatible with other systems?
: Does it have acceptable performance under load?
These tests are usually run the by the computer operation team in a business. The answers to their questions could have significant impact if new computer hardware should be required, and adversely affect the "BUSINESS CASE".
It would appear obvious that the operations team should be involved right from the start of a project to give their opinion of the impact a new system may have. They could then make sure the "BUSINESS CASE" is relatively sound, at least from the capital expenditure, and ongoing running costs aspects. However in practice many operations teams only find out about project just weeks before it is supposed to go live, which can result in major problems.

5 SIMPLE MODEL



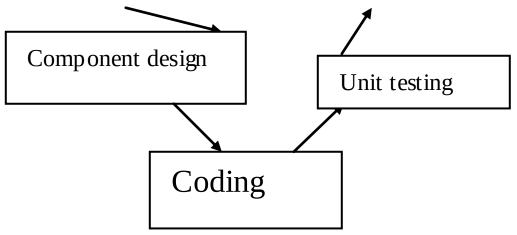


Figure 1

The V-Model is system development model designed to simplify the understanding of the complexity associated with developing in system engineering. it is used to define a uniform procedure for product or project development.The V-model is a software development process which can be presumed to be the extension of the waterfall model. It was the first proposed by Paul Rook [11] in the late 1980s and is still in use today. The V-Model demonstrates the relationships between each phase of the development life cycle and its associated phase of testing. Instead of moving down in a linear way, the process steps are bent upwards after the coding phase, to form the typical V shape. The V-model deploys a well-structured method in which each phase can be implemented by the detailed documentation of the previous phase. Testing activities like test designing start at the beginning of the project well before coding and therefore saves a huge amount of the project time. The purpose of V model is to Improve efficiency and effectiveness of software development and reflect the relationship between test activities and development activities as shown in Figure 1. This model is very simple and is perhaps the most traditional model followed for management of software tests.

The basic V Model development process is divided into Understanding of user's requirements, performing requirements analysis, designing the initial outline, designing advanced detailed and describing required tests in the basic development process as Shown in figure 1 from left to right on each other counter parts. Software testing is too important to leave to the end of the project, and the V-Model of testing incorporates testing into the entire software development life cycle. In a diagram as in Figure 1 of the V-Model, the V proceeds down and then up, from left to right depicting the basic sequence of development and testing activities. The model highlights the existence of different levels of testing and depicts the way each relates to a different development phase.

Like any model, the V-Model has detractors and arguably has deficiencies and alternatives but it clearly illustrates that testing can and should start at the very beginning of the project. In the requirements gathering stage the requirements are gathered, verify and validated in order to justify the project. The business requirements are also used to guide the user acceptance testing. The model illustrates how each subsequent phase should verify and validate work done in the previous phase, and how work done during development is used to guide the individual testing phases. This interconnectedness lets us identify important errors, omissions, and other problems before they can do serious harm.

6 Modified Model

V model is the most representative model for traditional software testing management. The purpose of the V model is to improve efficiency and effectiveness of software development and reflect the relationship between test activities, development activities and Maintenance activities. Once the system has been made functional and all activities have been performed, if it is not maintained properly, all the development and testing efforts shall go in vain. Thus in this section of the paper we propose a new improved V model called as the Advanced V model that reflects the relationship between the development activities, test activities and maintenance activities in order to achieve a highly efficient and reliable system.

7 Improved V-Model

It is basically based on testing and maintenance that is more effective in this model. Software testing is described as a continuous improvement process that must be integrated in into an application maintenance methodology. The term software maintenance usually refers to changes that must be made to software after they have been delivered to the customer or user. The definition of software maintenance by IEEE [1993] [9] is "The modification of a software product after delivery to correct faults, to improve performance or other attributes, or to adapt the product to a modified environment.
Software testing and software maintenance are the most important phases of software development life cycle that go hand in hand to obtain reliable software. The improved V model of testing incorporates testing and maintenance activities into the entire software development life cycle."
In the diagram of improved model , it proceeds down and then up, in first half it move down and second half its move upward direction. From left to right depicting the basic sequence of development, testing and maintenance activities. The model highlights the existence of different levels of testing with respect to their maintenance activities tests and depicts the way each relates to different development phase activities. The testing commences together with the initial phase of development of the project. In the requirements gathering stage the requirements are gathered, analyzed, verified and validated in order to justify the Project. The business requirements at

the same time also guide to the acceptance testing. Once the acceptance testing is done the error free product needs to be deployed as per the satisfaction of the customer The improved V Model development process is divided into understanding of the business case user's requirements, performing requirements analysis, specification, designing the initial and detailed outline and laying out the program specifications. Then the required tests are described in the basic development process as shown in improved model from left to right on each other counterparts Along with the maintenance tests being carried out for each of the test activity as shown in the figure. Once the activities of the development phase starts simultaneously the activities of the testing phase and maintenance phase commence. Ever imagined Software being deployed without carrying out these testing activities? No would be the prompt reply. So with the unit testing, the modules programmed are tested and test cases (set tests) are designed. Then moving on to the next level is integration testing where individual modules are integrated and tested for functionality. But this is incomplete without regression testing as the updated changes are then reflected. System testing describes the testing of the system as a whole. Along with it we need to do the security testing in order to check the systems compliance to various security threats. In the modern era where technology is moving with the speed of light, the need to deploy security measures has increased. Thus security threats like unauthorized access, user permission grant needs to be checked at each phase of development activity and testing activity. Now we need to a release test in this test even system meets all its requirements there is still a case to be answered that it will benefit the business. The linking of business case to release testing is looser then the others, but is still important. Release testing is about seeing if the new or changed system will work in the existing business environment mainly this means the technical environment and checks concerns such as:

: Does it affect any other systems running on the hardware?
: Is it compatible with other systems?
: Does it have acceptable performance under load?

These tests are usually run the by the computer operation team in a business. The answers to their questions could have significant impact if new computer hardware should be required, and adversely affect the "BUSINESS CASE".
It would appear obvious that the operations team should be involved right from the start of a project to give their opinion of the impact a new system may have. They could then make sure the "BUSINESS CASE" is relatively sound, at least from the capital expenditure, and ongoing running costs aspects. However in practice many operations teams only find out about project just weeks before it is supposed to go live, which can result in major problems.Then once the user is satisfied after conducting the alpha and beta tests of acceptance test activity the software or the product is deployed at the customers place. Thus a continuous interaction of the development activities, testing activities and maintenance test activities completes the software development life cycle of the product thereby efficiently carrying out the software test management activities. It's played a very important role in software development.

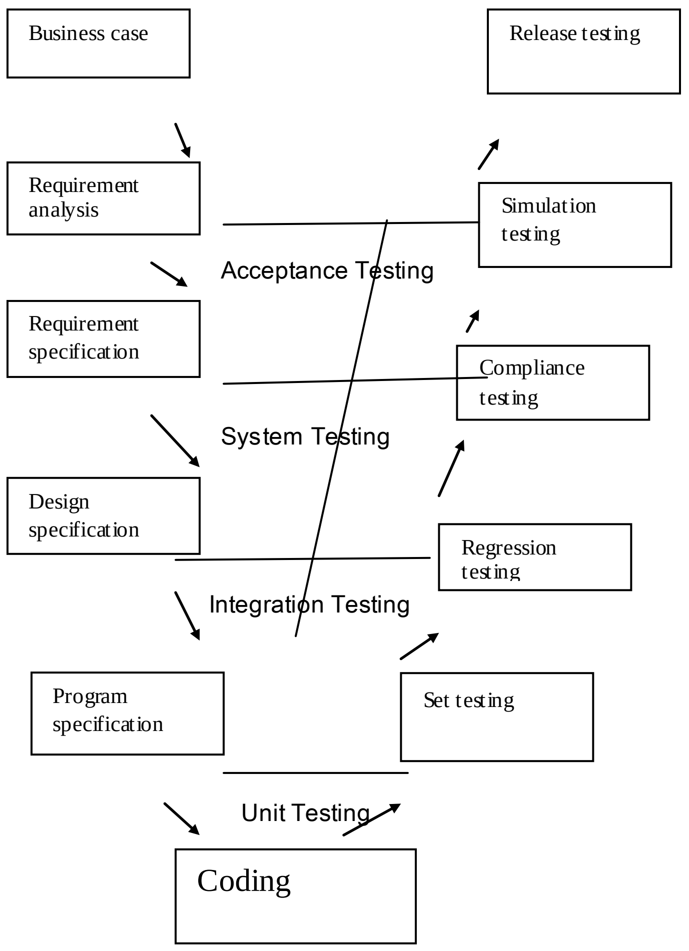


Figure 2

8 ARCHITECTURE OF IMPROVED V-MODEL

Architecture gives the structural description of the components and thereby helping us to understand the components in a modified and better way. This section of the model will describe the architecture of the component for software test management.

8.1 Management of Improved V Model

In this improved model section construct a software test management structure of component from a structural point of view. The structure of the components of software testing management and software maintenance tests are the basic elements, and they compose of software testing management structure.

The necessary and beneficial structural components are analyzed from software development, testing and maintenance point of view. That is, we need to identify all the builders and destroyers of the software testing management such as customers and the role of the users are as follows:

1. Business manager: A business manager is a provider of the business case that will be depended on the business environment and also related to the user. It's very important.
2. Project Manager: A project manager is a facilitator. Means provider .The project manager is the one who is responsible for making decisions in such a way that risk is controlled and uncertainty minimized. Every decision made by him should ideally be directly benefit the project. He must possess a combination of skills including the ability to ask penetrating questions, identify unstated assumptions, and resolve personnel conflicts along with more systematic management skills.
3. Software Development Manager: Leads a team of Programmers. Development Manager is responsible for leading the software development team in support of the software development life cycle process, change management, development environments and production releases. He will provide overall supervision and technical guidance to the development team in understanding requirements, preparing high level and low-level designs, coding and building the software.
4. Software Architect: An architect acts as a technically savvy business owner. He deals with the interactions of systems, whether between components written in different languages at different times and at different locations, or between components of the same software system that use the same coding language.
Architects deal with the interactions of systems, whether between components written in different languages at different times and at different locations, or between components of the same software's system that use the same coding language.
5. Software Developer: A software developer is a person concerned with facets of the software development process wider than design and coding, a somewhat broader scope of computer programming or a specialty of project managing including some aspects of software product management.
6. Testing head: The testing head compare all tests and also use release test according to business environment and business case then instructed the releasing project. Its played most important role.
7. Test Manager: Test managers really serve two different customers, their testers and corporate management. For the testers, he helps develop product test strategies, and provides test expertise to the testing group. For management, he gathers product information so that corporate management can decide when the product is ready for implementation.
8. Test Leader: Technical leader acts as in interface between the test manager and the testing team. He is responsible for the completion of the testing as per the designed time frame.
9. Test Designer: Test designer is responsible for developing test strategies and test plans. He provides an assessment on the overall status of the testing program. He stays well informed and connected with the industry and the current trends in the technologies available.
10. Software Tester: Software tester is responsible for carrying out software testing using various strategies of testing. He builds up the test cases and test plans for the project.
11. Quality Manager: Quality Manager works towards customizing software development processes. He is responsible for creating and implementing a quality management program plan for the entire organization and works towards process improvement.
12. Quality Assurance Engineer: Software quality assurance engineer deals with the location of the defect and mechanisms to prevent defects.
13. Quality Control Engineer: Software quality control engineer looks after the set of activities designed to evaluate the quality of developed software.
14. Quality Guarantee Engineer: Software quality guarantee engineer is responsible for maintaining software quality. He is responsible for tackling and solving all software problems.
15. Quality head. this is responsible for releasing the project .In final step the project head release the project according to business environment. This is most important in the software releasing.

8.2 Management Structure service

The services of these fifteen components obtained for the software testing management and maintenance are as follows:

Project Management Plan

1. Business case manager manages the business case and provide the environments for the project.

2. Project Manager manages the project and provides the project management plan.
3. Software Development Manager is responsible for all issues regarding system development.
4. Software Designer provides the system designing services.
5. Software Developer provides the program design and development service. Test Management Plan
6. Testing head manage the release test.
7. Test Manager provides the acceptance testing service and test management plan.
8. Test Leader provides the system testing service.
9. Test Designer provides the Integration testing service.
10. Software tester provides the unit testing service.
Quality Management Plan
11. Quality head manages the release testing in final process
12. Quality Manager provides the simulation testing service and quality management plan.
13. Quality Assurance Engineer provides the security testing service.
14. Quality Control Engineer provides the Regression testing service.
15. Quality Guarantee Engineer provides the test case service.

CONCLUSION

Here propose prepare an improved V model describing that for efficient software testing management along with the development and testing process improved the maintenance process is also equally important. Thus we have integrated these processes for efficient software testing management. We have achieved what should be done, why should be done and how it should be done in software testing management at all the phases of the software development. Maintaining the software before and after testing helps improving the quality of the software to a large extent. I think it will be efficient model that provides guidance for the planning and realization of projects. The following objectives are intended to be achieved by a project execution:

Minimization of Project Risks: The V-Model improves project transparency and project control by specifying standardized approaches and describing the corresponding results and responsible roles. It permits an early recognition of planning deviations and risks and improves process management, thus reducing the project risk.

Improvement and Guarantee of Quality: As a standardized process model, the V-Model ensures that the results to be provided are complete and have the desired quality. Defined interim results can be checked at an early stage. Uniform product contents will improve readability, understand ability and verifiability.

Reduction of Total Cost over the Entire Project and System Life Cycle: The effort for the development, production, operation and maintenance of a system can be calculated, estimated and controlled in a transparent manner by applying a standardized process model. The results obtained are uniform and easily retraced. This reduces the acquirers dependency on the supplier and the effort for subsequent activities and projects.

Improvement of Communication between all Stakeholders: The standardized and uniform description of all relevant elements and terms is the basis for the mutual understanding between all stakeholders. Thus, the frictional loss between user, acquirer, supplier and developer is reduced.

References:

Marshall Anthony, Student Software Architect, Fairleigh Dickinson University

Semyon Axelrod and Mike Regan, GMAC/RFC

Dewayne E. Perry and Alexander L. Wolf. "Foundations for the Study of Software Architecture". ACM SIGSOFT Software Engineering Notes, 17:4, October 1992:

[1] Bass, Len; Paul Clements, Rick Kazman (2003). Software

Architecture In Practice, Second Edition. Boston: Addison-Wesley. pp. 21–24. ISBN 0-321-15495-9.
[2] [Myers79] Myers, Glenford J., The art of software testing, Publication info: New York: Wiley, c1979. ISBN: 0471043281 Physical description: xi, 177 p. .
[3] [Hetzel88] Hetzel, William C., The Complete Guide to Software Testing, 2nd ed. Publication info: Wellesley, Mass. : QED Information Sciences, 1988. ISBN: 0894352423.Physical description: ix, 280 p..
[4] Sommerville, Ian, Software Engineering, 6th ed., Addison Wesley, 2000
[5] IEEE Standard for Software Verification and Validation Plans (Reaff.1992). IEEE Std 1012-1986.
[6] IEEE Standard for Software Unit Testing. IEEE Std 1008-1987.
[7] IEEE Standard Glossary of Software Engineering Terminology. IEEE Std 610.12-1990.
[8] IEEE Standard for Software Test Documentation. IEEE Std 829-1998.
[9] IEEE. 1993. IEEE Standard for Software Maintenance. IEEE Std 1219-1993. Institute of Electrical and Electronics Engineers, inc., New York, NY.
[10] Gopalswamy Ramesh; Ramesh Bhattiprolu (2006). Software maintenance : effective practices for geographically distributed environments. New Delhi: Tata McGraw-Hill. ISBN 9780070483453.
[11] Rook, Paul, E. Rook, "Controlling software projects", IEEE Software Engineering Journal, 1(1), 1986, pp. 7-16.
[12] [Han93] Hannus, Jouko, Prosessijohtaminen. Gummerus Kirjapaino, Jyväskylä, 1993.
http://members.tripod.com/bazman/index.html

^ a b c d Clarus Concept of Operations. Publication No. FHWA-JPO-05-072, Federal Highway Administration (FHWA), 2005

^ "Systems Engineering for Intelligent Transportation Systems". US Dept. of Transportation. p. 10. Retrieved 2007-06-09.

^ a b c d e f g h Forsberg, K., Mooz, H., Cotterman, H. Visualizing Project Management, 3rd edition, John Wiley and Sons, New York, NY, 2005. Pages 108-116, 242-248, 341-360.

^ a b c d e 3.3 to International Council On Systems Engineering (INCOSE), Systems Engineering Handbook Version 3.1, August 2007, pages 3.8^ Forsberg, K., Mooz, H. (1998). System Engineering for Faster, Cheaper, Better. Center of Systems Management. Archived from the original on 2003-0420.^ "The SE VEE". SEOR, George Mason University. Retrieved 2007-05-26.^ a b c d e Forsberg, K. and Mooz, H., "The Relationship of SystemsEngineering to the Project Cycle," National Council On Systems Engineering (NCOSE), October 1991

Garlan and Perry, guest editorial to the IEEE Transactions on Software Engineering, April 1995:

Dewayne E. Perry and Alexander L. Wolf. "Foundations for the Study of Software Architecture". ACM SIGSOFT Software Engineering Notes, 17:4, October 1992:

Bass, Clements, and Kazman. Software Architecture in Practice, Addison-Wesley 1997:

Bass, Clements, and Kazman. Software Architecture in Practice 2nd ed, Addison-Wesley 2003:

Bass, Clements, and Kazman. Software Architecture in Practice, Addison-Wesley 1997:

Wikipedia and Encyclopedia