Andreas Spillner

Dr. Spillner is working as Professor at the Hochschule Bremen (University of Applied Sciences) where he is responsible for software engineering and real time systems. Dr. Spillner has over 20 years experience in software development and teaching. He has been involved with development projects for government and also research projects in collaboration with industry. He has a degree in computer science from the Technical University of Berlin and received his PhD on the dynamic integration testing of modular systems from the University of Bremen in 1990. He has also worked as Research Assistant and as Associate Professor at the University of Bremen.

Dr. Spillner is currently the chairman of the German Computer Society Special Interest Group for the Testing, Analysis and Verification of Software Systems (GI-TAV). His research interests include the validation of software, test methods (especially for large software systems), the testing of object oriented software systems, and software development process models.

He was speaker (and co-speaker) at many national and international conferences (some examples):
_ EuroSTAR´98, Munich, Germany, 1998
(IQUIP-Award for best contribution to EuroSTAR´98)
_ Conquest 2000, 2000, Nuremberg, Germany, (Invited Speaker)
_ Quality Week 2000, San Francisco, USA, 2000 (together with Prof. Dr. U. Breymann)
_ EuroSTAR´2000, Copenhagen, Denmark, 2000
_ EuroSTAR´2001, Stockholm, Sweden, 2001 (together with Dr. K. Vosseberg)

He was member of different program committees (some examples):
_ International Software Quality Week Europe, Brussels, Belgium, 1998
_ EuroSTAR´98, Munich, Germany, 1998
_ TEST 2000 - The European Software Testing Congress, London, England, 2000
_ EuroSTAR 2001, Stockholm, Sweden, 2001
****

The W-MODEL – Strengthening the Bond Between Development and Test

Andreas Spillner
University of Applied Sciences Bremen
Germany

Abstract

In software development, 30 to 40% of all software activities are testing related. That is why it is critical to launch test activities at the beginning of the project rather than after coding is completed. While new software development models such as the Rational Unified Process and eXtreme Programming (XP) continue to be popular with practitioners, the V-model has gained particularly wide acceptance. Based on the V-model, this paper describes a model that shows how the tasks for testing relate to the tasks in the development model. This testing model – the W-model – further clarifies the priority of the tasks and the dependence between the development and testing activities. Though as simple as the V-model, the W-model makes the importance of testing and the ordering of the individual testing activities clear. It also clarifies that testing and debugging are not the same thing.

Keywords: software development process models, testing, quality assurance

1. Introduction

In the last 20 years a whole range of software development models have been presented and discussed. The objective of these considerations was always to develop a model that on the one hand comes close to reality i.e. that is suitable in practice and on the other hand serves to give structure to the process of software development.

Most of the models have some deficiencies with regard to test activities. Testing starts after the coding phase at which point the implementation appears to be complete. The tight link between test, debug, change and re-test is not usually emphasised in such development models and the handling of this process is unclear. The W-model introduced here aims to overcome such problems and to improve the status of testing in the development life cycle.

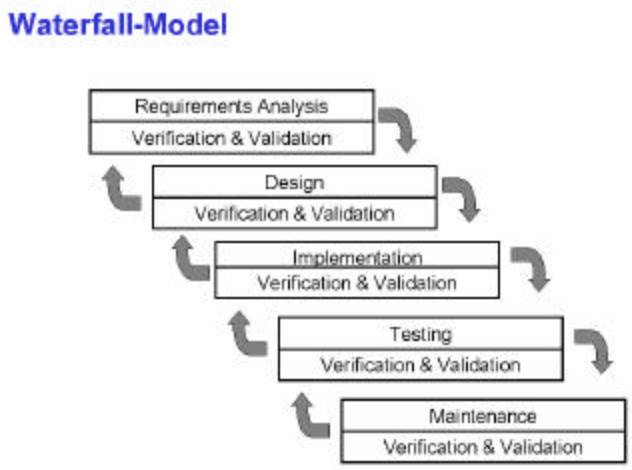
The paper is structured as follows: a brief overview of existing development models is presented in the next section. This is followed by the description of the W-model and the activities in each stage of the development and test process. Section 4 describes two new development models. The final section presents a short summary and some conclusions.

2. Software Development Process Models

In the following, several models are presented. The models that are widely known are not completely described. Instead, the emphasis lies on the consideration of the test activities in the different models.

2.1 Waterfall-Model

One of the first models for software development is the so-called waterfall-model by B. W. Boehm [1]. The individual phases i.e. activities, that were defined here are to be found in nearly all models proposed since. In this it was set out that each of the activities in the software development must be completed before the next can begin. A return in the development process was only possible to an immediate previous phase.



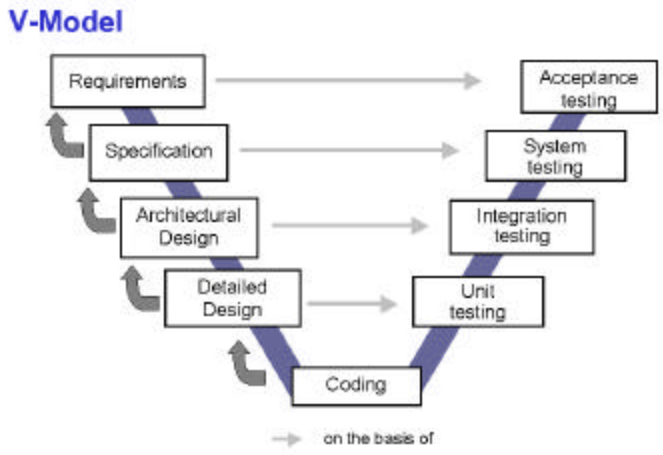
In the waterfall-model, testing directly follows the implementation. By this model it was suggested that activities for testing could first be started after the implementation. Preparatory tasks for the testing were not clear. A further disadvantage is that testing, as the last activity before release, could be relatively easily shortened or omitted altogether. This, in practice, is unfortunately all too common.

In this model, the expense of the removal of faults and defects found is only recognisable through a return to the implementation phase.

2.2 Spiral-Model

In the spiral-model (B. W. Boehm [2]) a cyclical and prototyping view of software development was shown. Tests were explicitly mentioned (risk analysis, validation of requirements and of the development) and the test phase was divided into stages. The test activities included module, integration and acceptance tests. However, in this model the testing also follows the coding. The exception to this is that the test plan should be constructed after the design of the system. The spiral-model also identifies no activities associated with the removal of defects.

2.3 V-Model



Many of the process models currently used can be more generally connected by the V-model where the “V” describes the graphical arrangement of the individual phases. The “V” is also a synonym for verification and validation.

The model is very simple and easy to understand. By the ordering of activities in time sequence and with abstraction levels the connection between development and test activities becomes clear. Oppositely laying activities complement one another i.e. serve as a base for test activities. So, for example, the system test is carried out on the basis of the results specification phase. The coarse view of the model gives the impression

that the test activities first start after the implementation. However, in the description of the individual activities the preparatory work is usually listed. So, for example, the test plan and test strategy should be worked out immediately after the definition of the requirements. Nevertheless it can contribute very well to the structuring of the software development process.

The disadvantage of the model is the coarse division into constructive work (including the implementation) on the left-hand side of the “V” and the more destructive tasks on the right-hand side. Here also the impression may develop that, after the implementation phase, a ready product can be delivered. A planned-in removal of defects and regression test is not given.

3. The W-Model

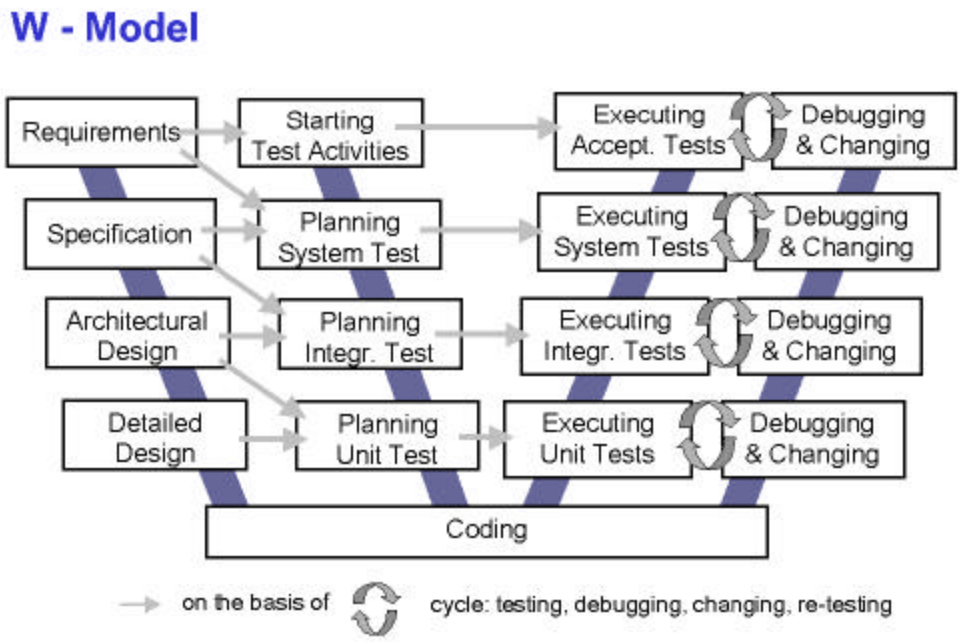
From the view of testing, all of the models presented previously are deficient in various ways:

- the test activities first start after the implementation
- the connection between the various test stages and the basis for the test is not clear
- the tight link between test, debug and change tasks during the test phase is not clear

In the following, the W-model is presented. This is based on the general V-model and the disadvantages previously mentioned are removed [3].

3.1 Idea

The test process usually receives too little attention in the models presented and usually appears as an unattractive task to be carried out after coding. In order to place testing on an equal footing, a second “V” dedicated to testing is integrated into the model. Both ‘V”s together give the “W” of the W-model.



It is already clear in the early development phases which testing activities must be carried out. In these it is not only the planning and management tasks that have to be carried out but also, for example, as the system is divided into components the corresponding test cases for checking the components interfaces can also be developed. When the tasks that a particular component should undertake are clear then the test cases to check these tasks lie relatively open to hand. Should these considerations be first placed at the integration test phase (as the V-model suggests) then a considerable increase in cost is necessary - the detailed knowledge of the tasks of the component must first be relearned.

No model clarifies the cycles between testing with defect discovery, debugging with defect localisation and the implementation of changes to remove defects together with re-testing. The

necessarily tight interaction between testing and the changes in implementation are clarified on the right-hand side of the W-model. This side contains not only the “destructive” test activities as in the V-model but also the “constructive” change activities that are carried out as a result of the discovery of faults and defects.

3.1 Start of the Test Activities
Review of the Requirements / Planning and Preparing Acceptance Test

At the beginning of the project the test activities must start. These first activities are:

- Fixing of test strategy and test concept
  - risk analysis
  - determine criticality
  - expense of testing
  - test intensity
- Draw up the test plan
- Organize the test team
- Training of the test team - If necessary
- Establish monitoring and reporting
- Provide required hardware resources (PC, data base, …)
- Provide required software resources (software version, test tools, …)

The activities include the foundations for a manageable and high-quality test process. A test strategy is determined after a risk evaluation, a cost estimate and test plan are developed and progress monitoring and reporting are established. During the development process all plans must be updated and completed and all decisions must be checked for validity.

In a mature development process reviews and inspections are carried out through the whole process [4]. The review of the requirement document answers questions like: Are all customer´s requirements fulfilled? Are the requirements complete and consistent? and so on. It is a *look back* to fix problems before going on in development. But just as important is a *look forward*. Ask questions like: Are the requirements testable? Are they testable with defensible expenditure? If the answer is no, then there will be problems to implement these requirements. If you have no idea how to test some requirements then it is likely that you have no idea how to implement these requirements.

At this stage of the development process all the knowledge for the acceptance tests is available and to hand. So this is the best place for doing all the planning and preparing for acceptance testing. For example one can

- Establish priorities of the tests depending on criticality
- Specify (functional and non-functional) test cases
- Specify and - if possible - provide the required infra-structure
- …

At this stage all of the acceptance test preparation is finished and can be achieved.

3.2 Review of the Specification / Planning and Preparing System Test

In the review meeting of the specification documents ask questions like: Is the specification testable? Are they testable with defensible expenditure? Only these kinds of specifications can be realistically implemented and be used for the next steps in the development process. There must be a re-work of the specifications if the answers to the questions are no.

Here all the knowledge for the system tests is available and to hand. Tasks in planning and preparing for system testing include:

- Establishing priorities of the tests depending on criticality
- Specifying (functional / non-functional) system test cases
- Defining and establishing the required infra-structure
- …

As with the acceptance test preparation, all of the system test preparation is finished at this early development stage.

3.3 Review of the Architectural Design | Detailed Design
Planning and Preparing Integration Test | Unit Test

During the review of the architectural design one can look forward and ask questions like: What is about the testability of the design? Are the components and interfaces testable? Are they testable with defensible expenditure? If the components are too expensive to test a re-work of the architectural design has to be done before going further in the development process.

Also at this stage all the knowledge for integration testing is available. All preparation, like specifying control flow and data flow integration test cases, can be achieved.

All accordingly activities of the review of the architectural design and the integration tests can be done here at the level of unit tests.

3.4 Executing Tests / Debugging and Changing

After coding is done the execution of the prepared tests can be started beginning with the unit tests. Checking the test results you have to decide if errors are found. After reporting the errors debugging is started. Finding the reasons for the errors, the defect localisation, and deciding if and how the problem should be fixed. This job is not part of testing and it has to be done by developers not by testers, but with close co-operation of both groups. After changing re-testing starts. The tester has to check that the error has been fixed and that no new problems have arisen. Often it is a test-debug-change-regression-re-test cycle.

The execution of tests, debugging, changing and re-testing activities have to take place at all stages from unit test to acceptance test.

3.5 Advantages of the W-Model

In the W-model the importance of the tests and the ordering of the individual activities for testing are clear. Parallel to the development process, in a tighter sense, a further process - the test process - is carried out. This is not first started after the development is complete.

The strict division between constructive tasks on the left-hand side and the more destructive tasks on the right-hand side that exists in the V-model is done away with. In the W-model it is clear that such a division between tasks is not sensible and that a closer co-operation between development and testing activities must exist. From the project outset onwards the testers and the developers are entrusted with tasks and are seen as an equal-rights partnership. During the test phase, the developer is responsible for the removal of defects and the correction of the implementation. The early collaboration and the tight co-operation between the two groups can often in practice avoid *conflict* meetings.

The W-model becomes closer to practice when the test expenditure is given 40% and more. The model clearly emphasises the fact that testing is more than just construction, execution and evaluation of test cases.

3.6 Disadvantages of the W-Model

Models simplify the real facts. In practice there are more relations between the different parts of a development process. However, there is a need for a simple model if all people involved in a project are to accept it. This is also a reason why the simple V-model so frequently used in practice.

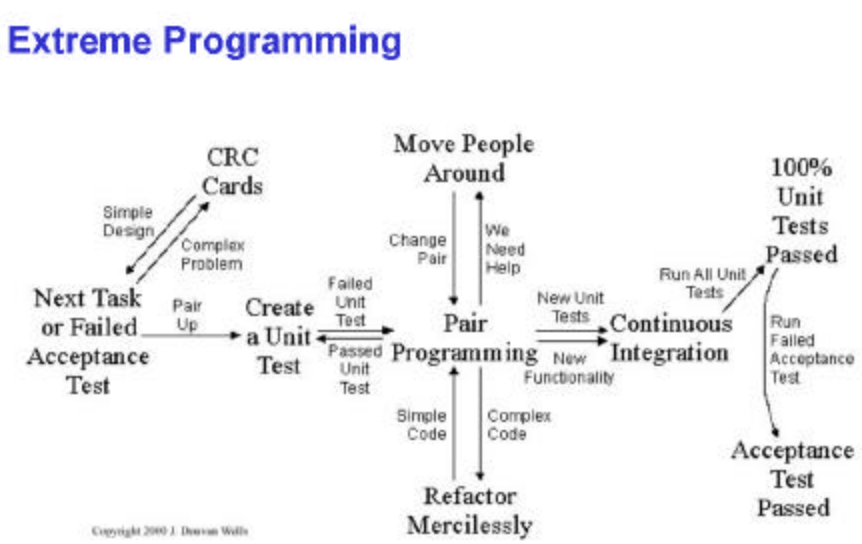
The models of software development presented do not clarify the expenditure needed for resources that need to be assigned to the individual activities. Also in the W-model it appears that the different activities have an equal requirement for resources (time, personnel, etc.) In practice this is certainly not the case. In each project the most important aspects may vary and so therefore the resource allocation is unlikely to be equal across activities. For highly critical applications the test activities certainly have higher weighting or at least equal weighting with other activities.

4. Two New Software Development Models

In the following, two models are presented. The models are not completely described. The consideration of the test activities in the two models is discussed shortly.

4.1 Extreme Programming

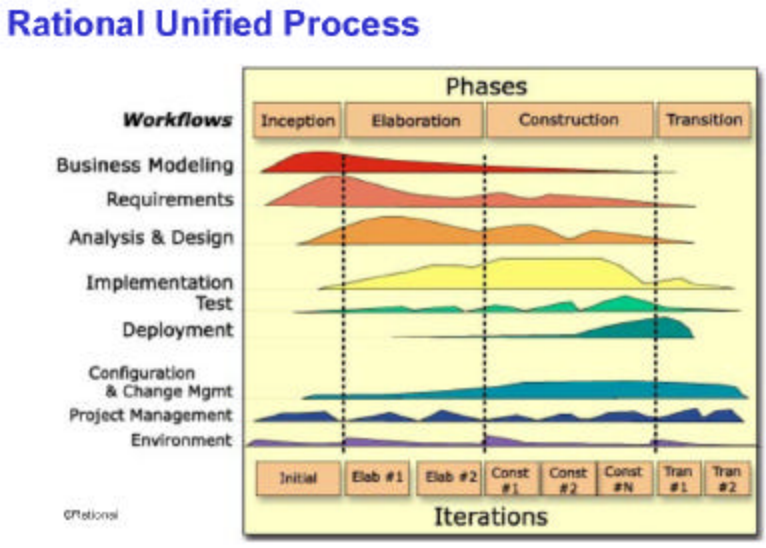
A further model of software development is currently frequently discussed: Extreme Programming [5]. Taking a simplistic view of the model one could say that extreme programming does not use specifications. The test cases initially defined are used as a description of the requirements. These are then used after the implementation to help check the (sub-) product.



Jeffries writes in [6]: “To be sure that new features work, write Unit Tests for every feature. Write them before you release the code, preferably before you even write it. Save all the unit tests for the whole system... Whenever Extreme Programmers release any code at all, every unit test in the entire system must be running at 100 percent!”

The idea in this excerpt from Extreme Programming can also be found in the W-model: the left part of the “W” can simply be omitted. This then leaves just the testing activities as tasks up to the point of implementation. The requirements for the system to be developed are then extracted from the specified test cases.

4.2 Rational Unified Process



The Rational Unified Process (RUP [7]) is based on the UML (Unified Modelling Language). The UML can be used for analysis and design in an object oriented software development. The development is managed in iterations, as in extreme programming.

RUP defines little concrete hints on the dependencies between development and test. Testing starts early and uses the documents of the

early development activities but it is not exactly described when and how the test activities are involved.

"However, testing is primarily employed when each build (as an implementation result) is integrated and system tested ([7, p. 296])". It seems that the main activities of testing begin after implementation is done.

5. Conclusion

In this paper, an idea is presented that may be seen as a stimulus for the discussion about the importance of testing. In the W-model, testing is consistently shown as a separate process that has a very tight interconnection with development activities. This testing model further clarifies the priority of the tasks and the dependence between the development and testing activities. Though as simple as the V-model, the W-model makes the ordering of the individual testing activities clear. It clarifies that testing and debugging are not the same thing.

It is not a complete new model; it is an extension of the well known and spread used V-model. The W-model shall replace the V-model. The W-model is already used in training in a successful German software development company. It is also the recommended model of a German consulting company.

If with the W-model the importance of testing becomes clearer and the test activities start earlier, then this can be seen as a further step towards improvement in the quality of software development.

References

[1] Boehm, B.W.: *Software Engineering Economics*.
Englewood Cliff: Prentice Hall, 1981

[2] Boehm, B.W.: *A Spiral Model of Software Development and Enhancement*.
IEEE Computer, May 1988, pp. 61-72

[3] Spillner, A: *From V-Model to W-Model – Establishing the Whole Test Process.*,
Proceedings Conquest - 4th Conference on Quality Engineering in Software Technology, 2000

[4] Gilb, T.; Graham, D.: *Software Inspections*.
Addison-Wesley, 1993

[5] Beck, K.: *Extreme Programming Explained: Embrace Change*.
Addison Wesley, 1999

[6] Jeffries, R.E.: *eXtreme Testing.*
Software Testing & Quality Engineering, March/April 1999, pp. 23-26

[7] Jacobson, I.; Booch, G.; Rumbaugh, J.: *The Unified Software Developement Process.*
Addison-Wesley Longman, Amsterdam, 1999

Helpful links (checked March 2002)

Extreme Programming
http://www.extremeprogramming.org/ (© J. Donovan Wells)
http://www.xProgramming.com/ (© Ronald E Jeffries)

Rational Unified Process
http://www.rational.com/products/rup/ (© Rational)

Unified Modelling Language
http://www.rational.com/products/uml/ (© Rational)

Biography

Dr. Spillner is working as Professor at the Hochschule Bremen (University of Applied Sciences) where he is responsible for software engineering and real time systems. Dr. Spillner has over 20 years experience in software development and teaching. He has been involved with development projects for government and also research projects in collaboration with industry. He has a degree in computer science from the Technical University of Berlin and received his PhD on the dynamic integration testing of modular systems from the University of Bremen in 1990. He has also worked as Research Assistant and as Associate Professor at the University of Bremen.

Dr. Spillner is currently the chairman of the German Computer Society Special Interest Group for the Testing, Analysis and Verification of Software Systems (GI-TAV). His research interests include the validation of software, test methods (especially for large software systems), the testing of object oriented software systems, and software development process models.

He was speaker (and co-speaker) at many national and international conferences (some examples):

- EuroSTAR´98, Munich, Germany, 1998
  (IQUIP-Award for best contribution to EuroSTAR´98)
- Conquest 2000, 2000, Nuremberg, Germany, (Invited Speaker)
- Quality Week 2000, San Francisco, USA, 2000 (together with Prof. Dr. U. Breymann)
- EuroSTAR´2000, Copenhagen, Denmark, 2000
- EuroSTAR´2001, Stockholm, Sweden, 2001 (together with Dr. K. Vosseberg)

He was member of different program committees (some examples):

- International Software Quality Week Europe, Brussels, Belgium, 1998
- EuroSTAR´98, Munich, Germany, 1998
- TEST 2000 - The European Software Testing Congress, London, England, 2000
- EuroSTAR 2001, Stockholm, Sweden, 2001

Contact

Prof. Dr. Andreas Spillner
Hochschule Bremen
University of Applied Sciences
Flughafenallee 10
D - 28199 Bremen
Germany

Phone / Fax: ++49 421 5905 -5467 / -5412
Email: spillner@informatik.hs-bremen.de
URL: http://www.fbe.hs-bremen.de/spillner/