DEVELOPING A NEW TESTING MODEL

Boriss Misnevs

SWH Technology, Exigen Group
29 Eizensteina, Riga, LV-1079, Latvia
Ph: (+371)-70322726. Fax: (+371)-7032710. E-mail: borism@ exigengroup.lv

Dmitry Daineko

SWH Technology, Exigen Group
29 Eizensteina, Riga, LV-1079, Latvia
Ph: (+371)-70322781. Fax: (+371)-7032710. E-mail: dmitri_daineko@exigengroup.lv

Key words: Testing Model, Software Engineering, Requirements

Testing models implementation at software engineering practice is analyzed. A kind of test model classification is suggested. Life-cycle oriented testing models and formal methods oriented models are discussed as well as product type oriented models. A possibility of testing automation regarding the selected model is analyzed. Practical methodologies are investigated to define a suitable testing model. Requirements for the appropriate testing model selection are formulated.

Introduction

Testing is a critical component of the software development process. Organizations have not fully realized their potential for supporting the development of high quality software products. To address this issue we are building a new testing model to serve as a guide to organization focusing on test process improvement [1]. Of course, this new model must fit more general quality models, for example, see [2].

One of the most widely recognizing models for conducting software testing is the V-model, which tracks the development life cycle and associated testing tasks at each phase. V-model testing is a structured testing approach that can be used with any project management or system development methodology. The V-model emphasizes quality from the initial requirements stage through the final testing stage. It focuses on testing throughout the development life-cycle, early development of test requirements, and early detection of errors. Each major deliverable in the development process is assessed, verified, validated and tested.

Unfortunately, this model does not serve independent test groups that are not part of development, especially groups that represent the customer [3].

Week sides of the V-model were investigates by Brian Marick in [4]. He came to the conclusion of the V-model total fail. This conclusion was made because of the following V-model's features:

1. Ignores the fact that software is developed in a series of handoffs, where each handoff changes the behavior of the previous handoff.
2. Relies on the existence, accuracy, completeness and timeliness of development documentation,
3. Asserts a test is designed from a single document, with being modified by later or earlier documents
4. Asserts that tests derived from a single document are all executed together.

A need of a replacement model development is obvious. This replacement model must assume imperfect and changing information about the product. Testing a product is a learning process. Testers should talk to developers, users, marketers, technical writes, and anyone else who can give clues about better tests. Testing model must be adoptable.

This paper is an attempt to fined main features of a possible replacement model searching among the most interesting (from the author's point of view) existing testing models. We will not touch any general testing methodology like a black box or white box testing, but will concentrate our attention on the specific testing models, which are tried to be classified in the paper below.

Test models classification

A lot of existing testing models can be found in the world software engineering practice. They use completely different approach as for goals definition, as well as for testing practice applied.

Let us separate four main testing model groups to analyze in this paper: development life-cycle oriented, automation / maturity level oriented, product architecture oriented and formal approach oriented.

First group. Test models of the first group are organically concerned with development life cycle. For example, V-model is based on the classical waterfall life-cycle model.

Life cycle models describe the interrelationships between software development phases. The common life cycle models are the following:

- spiral model
- waterfall model
- throwaway prototyping model
- evolutionary prototyping model
- incremental/iterative development
- reusable software model
- automated software synthesis

Each of this development life cycles has testing activities, which can be represented by an appropriate testing model.
Most of modern software development is a chaotic activity, often characterized by the phrase "code and fix". A typical sign of such a system is a long test phase after the system is "feature complete". As a reaction to these practices, a new group of lightweight methodologies have appeared in the last few years. Extreme Programming (XP) and CRUM technologies could be used as examples. Two the most meaningful test process features of XP are following:

a) Writing unit tests before programming and keeping all of the tests running at all times.

b) Integrating and testing the whole system--several times a day.

Second group. Test models of the second group are intended to evaluation and support of testing process the SEI CMM like maturity. In most cases the testing process maturity is simply defined by testing automation level.

We can name the Automated Test Lifecycle Methodology (ATLM) as an example of a structured methodology [5], which is geared toward ensuring successful implementation of automated testing.

Another interesting model is Testing Maturity Model developed at Illinois Institute of Technology [6]. The TMM is intended to be used to identify the current testing capability state and initiate a testing improvement program. This model use the historical model provided in a key paper by Galperin and Hetzel [7]. This TMM model implement so called Extended/Modified V-model version to adopt the TTM to the real software development process. Software Testing Maturity model for automating testing was suggested by Krause in 1994. IBM has its own statistical concept known as Clean Room software engineering. Mercury Interactive also can be mentioned as one of leading software automation test tools provider.

Separately Six Sigma Solutions must be noted. Six Sigma uses comprehensive test automation approach based on statistical methods [8].

Third group. Third group contains testing models, which use special product architectural features to implement the testing process. For example, we may name a lot so called Object-Oriented (OO) models for software testing.

A good example of OO testing tools set is Software Testworks products: STW/Regression, STW/Advisor and STW/Coverage. Implementation of OO model (based on UML) for testing is used by Rational products TestStudio, Test RealTime and ClearQuest.

Forth group. To this group belong different testing models, which use quite formal approach for test process implementation. They can use different statistical models for test case development and planning or formal languages for specification description and test generation.

As an example, we can mention NIST project of Software Testing by Statistical Methods. Software testing by Statistical Methods is explicitly described in [9]. Another NIST project, leaded by P.E. Black was devoted to Automatic Test Generation from Formal Specifications [10].

To study any of existing testing models we can find that each of them could be definitely assign to one of the mentioned above group, but in the same time each of this models has more or less visible features from the rest groups (e.g. see IBM GOTCHA-TCBeans Software Testing Methodology, Rational “ICED T” model or SDT Dotted-U Model). We suggest that a new testing model must have clearly defined characteristics, which can allow this model classification regarding all four named groups.

“The fifth” element of testing model

One more dimension of the suggested testing model must be discussed in this paper. It is the “fifth” element – testing personnel. This dimension of the model must help organizations successfully address their critical people issues regarding testing process and maintain an appropriate testing organization structure.

We separated testing personnel as a different type of the model element (it is not a” dimension”) from all previously discussed. It is done because the human factor is closely linked with all “four” discussed dimensions and cannot be considered as independent (or orthogonal).

The software testing process has evolved considerably and has reached the point where it is a discipline requiring trained professionals. To succeed today, an organization must be adequately staffed with skilled software testing professionals who get proper support from management [14].

Testing process, and of course, testing team should be independent, unbiased, and organized for the fair sharing of the recognition and rewards for contributions made to product quality.

Typical test process implementation's risks are listed below (this list can be used as a checklist when planning organizational modifications):

- Test independence, formality, and bias are weakened or eliminated.
- People in testing do not participate in reward programs.
- Testing becomes understaffed.
- Testing becomes improperly staffed with too many junior individuals.
- Testing is leaded by inexperienced managers.
- There is no leverage of test knowledge, training, tools, and process.
- Testing lacks the ability to stop the shipment of poor quality products.
- There is a lack of focused continuous improvement.
- Management lacks the bandwidth to manage testing groups.
- The quality focus is not emphasized.
- Test managers become demoralized with the lack of career growth.

Let us review possible organizational solutions to be addressed individual competencies improvement, effective testing teams' development, and high performance motivation. Let us clarify the five typical approaches to testing organization. In practice it may reflects the evolution of the development organization maturity.

Approach 1. Testing is each person's responsibility

There is a group of product developers whose primary responsibility is to build a product. The problem with this approach is that it violates a basic assumption – testing must be done independently.

Approach 2. Testing is each development unit's responsibility

There is a group of product developers, within the group there are product developers assigned to testing each other's code. In realty, these employees will take time to test other's people's code as time and skills permit. They will usually not get very good at learning the special skills, tools, and methods of testing while they are simultaneously responsible for developing product.

Approach 3. Dedicated test unit in development

Test team that is part of the development organization. This approach usually puts the test group under the second line product development manager. Possible issue of this approach may be the following: does the development manager meet the test management requirements? As organizations grow, more people are hired into the test team, and multiple test groups are needed. A new issue arises. Should the multiple test groups be centralized or decentralized?

Approach 4. Independent test unit within organization

Central test organization exists within and serves a product development division. This approach adds more opportunities for a test manager/director to significantly impact the organization. Disadvantages of this approach are following: potential lack of teamwork at individual/project level, possible lack of consistent test methodologies across organizations with vice president direction.

To reduce above-mentioned disadvantages we can add a new test unit – it is a test technology group. This test technology group is responsible for: leading and managing testing process and testing productivity improvement efforts. The group drives and coordinates testing training programs; coordinates planning and implementation of testing tool programs; documents test process, standards, policies, and guidelines as necessary; recognizes and leverages best practices within testing groups; recommends consensus for, and implements key testing measurements.

Approach 5. Testing outsourcing

There is other approach to testing – it is outsourcing test efforts. That can provide critical support in the following ways: field support/consulting, personnel, technology, hardware, and facilities. There are five scenarios in which it is appropriate to outsource either some or all of the QA/test functions: (1) outsourcing all of the SQA/test functions, (2) outsourcing to augment your current QA effort by providing specialized skills and/or capabilities, (3) outsourcing to add resources, (4) outsourcing for testing in support of existing products, and (5) outsourcing to get an independent view. Each scenario has different project management challenges, requirements, time frames, deliverables, and expectations.

A set of recommended tasks is listed below to help a software organization with testing group maturing:

- provide the ability for rapid decision making;
- enhance teamwork, especially between product development and testing development;
- provide for an independent, formal, unbiased, strong, properly staffed and rewarded, test organization;
- help to coordinate the balance of testing and quality responsibilities;
- assist with test management requirements;
- provide ownership for test technology;
- leverage the capabilities of available resources, particularly people;
- positively impact morale and career path of employees (including managers)?

Several world wide known models could be mentioned as an example of people oriented model. One of them is Microsoft Readiness Framework Process Model. [13] This model supports an organization's readiness to adopt a new technology. This includes the readiness of staff (IT development, IT operations, management, and users) and the readiness of the organization's processes, that is, the "people and process readiness." This framework also supports gaining sufficient sponsorship, commitment, and involvement for the technology change to ensure success. Another widely recognized model is People CMM by SEI.

As example of the outsourcing test organizations we can point out such organizations as VeriTest, Mercury Interactive, QA Labs Inc., eTesting Labs, Metamor Software Solutions, and etc.

Requirements for a “good” testing model

Let us say that an alternative “good” testing model must be at least “four dimensional”. Testing personnel may be considered separately as the “fifth” model element. It does not mean that we must mass four or more existing models together. It does mean that we have to tune and adjust with each other all four dimensions of the new model. Only a model balanced in this way could be considered as an acceptable for the modern software development process.

While every attempt is made to focus it to testing, we know, that testing does not stand alone. It is intimately dependent on the software development activity and therefore draws heavily on the development practice, as discussed just above. But finally, testing is a separate process activity – the final arbiter of validity before the user assesses its merit. So it is reasonable to develop a testing model from the testing primary goal, not from the other development requirements and practices.
Requirements for a 'good" testing model may be the following:

1. A useful model must allow testers to consider the possible savings of deferent test execution.
2. A model must allow to executed test derived from a description of a component before the component is fully assembled
3. A model must allow individual tests to be designed using information combined from various sources
4. A model must allow tests to be redesigned as new sources of information appear.
5. A test model should force a testing reaction to every code handoff in the project
6. A test model must explicitly encourage the use of source of information other then project documentation during test design.
7. A test model must include feedback loops so that test design takes into account what's learned by running tests.

To make this very general requirement more practical let us list particular testing activities, which can benefit our "good" testing model in the case of its implementation. This model, from our point of view, must contain all possible activities, which can be used in a software verification process.

Review and Inspection. Software inspection, invented by Fagan in the mid 70's at IBM, remains one of the most efficient methods of debugging code.

Formal Entry and Exit Criteria. The idea is that every process step (handoff), be it inspection functional test, or software design, has a precise entry and precise exit criteria. This model called ETVX was also invented by IBM.

Variations of Functional Test. A variation refers to a specific combination of input conditions to yield a specific output condition. The best practice involves understanding how to write variations and gain coverage, which is adequate to thoroughly test function.

Multi-platform Testing. Testing on multiple platforms has become a necessity for most products. Therefore techniques to do this better, both in development and testing, are essential.

Automated Test Execution. The automated test execution has a significant impact on both the test tool set for test execution and also the way tests are designed. Integral to automated test environment is the test oracle that verifies current operation and logs failure with diagnosis information.

Usability Testing. For a large number of products it is believed that usability becomes the final arbiter to quality. Usability testing needs to not only assess how usable a product is but also provide feedback on methods to improve the user experience and thereby gain a positive quality image.

"Requirements" for Test Planning. Requirements management and its transition to produce test plan is an important step. Sometimes it is planed to test what ought to be developed as opposed to what is needed in the market.

Automated Test Generation. Test case writing is a prime candidate for saving through automation. On other hand, there do exist a few techniques and tools that have been recognized as good method for automatically generating test cases.

Teaming Testers with Developers. Coupling of testers with developers improves both the test cases and code that is developed. An extreme case of this practice is Microsoft, where every developer is shadowed with a tester.

Code Coverage. There are a host of metrics: statement, branches, and data that are implied by the term code coverage. Today, there exist several tools that assist in this measurement and additionally provide guidance on covering elements not yet exercised.

Automated Environment Generator. Tools that can automatically set up environments, run test cases, record the results, and then automatically reconfigure to a new environment, have high value.

Test models and project reality

We have to understand clearly that not tester, not test team leader and not project manager are intended to develop a kind of "testing model" within his current project responsibility and implement it in his work. The real project situation, especially for the dedicated testing team, usually is based on the existing local testing practice, already used tools (or absence of testing tools) and fuzzy testing process requirements. Product development life cycle is not formally defined in project documentation as well. Usually one test team is used to test completely different products from different project teams concurrently using the same test equipment. In this situation really no chance to introduce any new testing model or practice exists.

Practice shows us that simple one or two key persons transfer cannot implement a technology movement from "good" projects to "week" one. Additional resources for testing process improvements have to be dedicated. A targeted testing model must be developed in advance.

The right solution is a Testing process improvement project sponsored by company's executives (see Approach 4 above). This suggested by SEI IDEAL process project group (external to testing team) might perform a kind of analysis to understand what testing model is used de facto and investigate requirements for a testing process changes.

At this stage of testing process improvement the "right" testing modal could be developed and adopted to the real product and project demands. For example, if XP is recognized as the main product development approach, the appropriate XP testing model's features must be requested.

Unwarranted attempt, for example, to use the V-model testing in the case of XP development will lead to problems with test process organization and cause the product quality reduction. In most cases a test case developer intuitively is searching for a good project design documentation, which will never exist. In opposite, only clearly recognized and managed XP testing model may force code developers document unit tests for using them as the replacement of detail design documentation.

Conclusion

Existing testing models are briefly discussed. A kind of classification of the existing testing models is suggested to be used for a new model development. Main features of a new alternative testing

model are sketched. New model must be at least "four dimensional" regarding the suggested classification. Testing personnel is considered separately as the "fifth" model element. Desirable best testing practices are listed. Possible problems in the new testing model development are characterized.

Discussed approach and made recommendation could direct software development organizations in their own testing model selection and implementation.

References

1. G. Urtans, B. Misnevs. *Quality System IT solutions – experience and recommendations*. 3rd International Conference on Total Quality, Riga. 1999. pp. 82-87.
2. B. Misnevs *UML models of Software Quality System*, The International Federation of Operational Research Societies Special Conference "Organizational Structures, Management, Simulation of Business Sector and Systems", Kaunas, Lithuania, 1998. pp.178 -180.
3. L.G. Hayes. *The Testing Piramid*, WorkSoft Inc., 2001.
4. Marick. *New Models for Test Development*. Reliable Software Technolgies, 1999.
5. E. Dustin. Automated Software Testing. Addison-Wesley, 2000.- 573 p.
6. I. Burnstein, T. Suwannasart, C.R. Carlson. *Developing a Test Maturity Model: Part 1*. Illinois Institute of Technology, 1996.
7. D. Galperin, B. Hetzel. *The Growth of Software Testing*. CACM, Vol. 31, No. 6, 1988, pp. 687-695.
8. W. Forrest. *Implementing Six Sigma Smarter Solutions Using Statistical Methods*. Wiley and Sons, New York, NY, 1999.
9. D. Banks. W. Dashiell, L. Gallagher etc. *Software Testing by Statistical Methods*, National Institute of Standards and Technlogy, 1998. 31 p.
10. P.E. Ammann, P.E. Black. *A Specification-Based Coverage Metrics to Evaluate Test Sets.* World Scientific Publishing, Singapore, 2000.
11. C. Kaner, J. Falk, H.Q. Nguyen *Testing Computer Software*, Wiley Computer Publishing, 1999 – 479 p.
12. William E.Perry, Randall W.Rice *Surviving the Challenges of Software Testing. A people-oriented approach*, Dorset House; ISBN: 0932633382, 216 p.
13. How to Maximize Your IT Investment: The Microsoft Readiness Framework Process Model.
14. Kit Edward, *Software testing in the real world*, Addison-Wesley, ISBN: 0-201-87756-2, 252 p.