RECENT AND FUTURE TRENDS AND CHALLENGES OF SOFTWARE TESTING

Bojana Koteska
Faculty of Computer Science and Engineering
Skopje, Macedonia

Bisera Dugalic
Faculty of Computer Science and Engineering
Skopje, Macedonia

ABSTRACT

In the modern information based world, software testing is an important element of software quality assurance. In order to satisfy new and more complex software systems requirements we need to improve the ways of software testing by adding software automation and different software methods. Better software testing helps increasing the software quality, decreasing testing time and development cycle time. In this paper we review the trends in software testing and we propose some methods for improving the software testing. The paper concentrates on recent and future trends in software testing, automation of software tests and new testing suggestions.

I. INTRODUCTION

All software and each change made in any piece of software, no matter how unimportant it may seem at first bring up the possibility of error. These potential errors raise the chance that the software will not accomplish its intended outcome. Software testing can be used in order to lower this risk. It can help to determine error issues and to prevent and resolve them.

Among the most important goals of testing is to see how the application behaves under normal load and in stress. Stress can be defined differently for distinct software behold on the software's requirements.

Regarding the fact that there is no method to accredit that software is free of error the process of testing can be used to reduce its number. If the process of testing is constantly included in the development process, it can be very useful in moving from one development phase to another. The software can be run in observed environment to check its performance in order to evaluate if the project is ready for the next phase.

Software testing depends on the approach applied in the process of testing. It is recommended that the testing process is implemented during the process of development since it can decrease the number of further errors in the software. Each phase of development is properly tested before the next one depending on it is developed. However in most cases, the process of software testing is only assigned at the end of development.

II. IMPORTANCE OF SOFTWARE TESTING EVOLUTION

Computer software is the main driving force in today rapid changing information world. Because of this, its reliability is one of the most important things in organization's software development. Making quality software products requires testing in all phases of software development: analysis, design, coding, testing and integration. [1]

Since it is very common for software systems to evolve, software testing must adapt to these changes rapidly. Software correction and adaptation to new platforms results in new systems' releases and software engineers must find a way to test them. The correlation between software evolution and software testing is often difficult to achieve. [2]

The process of software evolution requires tests that evolve at the same time. That makes the evolution more complex, difficult and time-intensive. On the other hand, the new software operations cannot be added without previously created tests. [2]

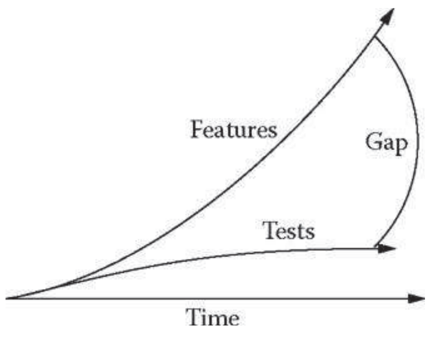

Figure 1: Gap between new features and tests.

Writing tests to understand the software before it is released is good approach for developing software systems.

III. RECENT TRENDS OF SOFTWARE TEST TECHNOLOGIES

Considering the fact that software testing does not stand alone but is a closely dependant on the developing process, it draws greatly on the development practices. The following test practices are considered to be among better when it comes to software testing. [13]

A. Functional Specifications

Functional Specification is meant to describe the external view of the matter usually indicating the options by which a utility could be requested. It can be very useful as basis for generating test oracle to help determining if the observed input/output is consistent with the specification. This practice brings the benefit of alongside generation of the tests and the development of the code. This can be very convenient from many aspects. Among important advantages is parallelism in execution process which helps in eliminating serialisation obstruction in development. This way developer can spare time because when the software code is finished, the tests are also ready to be run against the code. Another important feature is that it brings a level of clarity from the aspect of the designer and the architect that is necessary for the general completeness of the development process.

With formal specification the chance for errors is reduced because it excludes uncertainty. There is an enduring concern about getting a specification that agrees with the right customer requirements considering the possibility for stated requirements to change during the development process. The formal specification ought to be analyzed in order to find potential mistakes and consequences regarding the specification. Once this is finished, there is a higher certainty that the system is tested against its actual requirements. [16]

B. Formal Entry and Exit Criteria

The concept of this software testing method is that every step in the testing process, no matter whether that is observation or software design of functional test, it has a particular entry and a particular exit criteria. Although with the decrease of the emphasis development, the usability of the process of entry and exit criteria may have been out of currency but it still allows very careful management of the software development process. The entry and exit criteria are defined in the development process and later are observed by the management to gate the movement from one stage to another. It is considered as stand-alone testing and is most useful in Unit Testing. With each test based on technical design documents and built to carry out a certain task, a single module is validated with the assumption it will operate in a particular way or give particular result.

The tests are performed in a test environment before the particular module is integrated into the system. When a bug is detected during the testing, depending on the proportion of the bug, it is decided if the module will be fixed before being approved. [17]

C. Functional Test-Variations

The point with this software testing approach is to make specific combination of input condition which will give a specific output. The process of making functional tests implies writing different variations of input and output in order to cover as much of the state as resolved important. The practice of writing the variations is considered to involve a factor of art, regarding the fact that there is no measure of coverage for functional tests. The essential part is to understand how to write the variations and gain coverage to sufficiently test the function.

According to previous experience, software may be tested by bringing together a group of familiar programmers, assigning their goals in broad terms, and then letting them differ on the way to these goals as their individual preparation, curiosity and biases demand. The basis of this approach is the persuasion that testing is an art. There is no initial mapping of the field to be analyzed and no precise assignment of field assignment. As a result to this, there is no way of prevent redundant analysis, no way of limiting analysis to fields with the higher probable payoffs and no curtain way to measure the analyzed field. Briefly, there is little manipulation over the testing process itself and few detached measures of test coverage. [18]

D. User Scenarios

The scenario is a story about someone trying to accomplish something with the product under test. Positive aspect of using user scenarios is that it tests the software in the way that is most likely to manifest on customers using it. Another important thing is that it reduces the complexity of writing test cases by moving to testing scenarios rather than features of application. Nevertheless, the process of developing user scenarios and using enough of them to get the necessary coverage at a functional level still presents a demanding task. The best practice when it comes to user scenarios is to capture methods of recording user scenarios and developing test cases according to them. Furthermore it could consider potential summary methods when characteristic failure scenarios happen.

Scenarios can also be appropriate to attach to documented software requirements, particularly requirements modeled with use cases because more complex tests are built up by designing a test that runs through a series of use cases. Also scenarios can be used to expose failures to deliver aimed advantages whether or not the company creates use cases or other requirements documentation. Scenario test provides a check on a benefit the program is supposed to deliver. Tests of separate units and mechanical combination tests of related units or their input variables are not designed to provide this kind of check. [19]

IV. AUTOMATION OF SOFTWARE TESTS

A. Automated test execution

The purpose of automated test execution is to minimize the quantity of manual work involved in test execution and to gain higher coverage with greater number of test cases. This way of testing has a big influence on the tools sets used for test execution and how the tests are designed. It is very convenient and very used in some segments of software testing but not in all of them. It needs to take advantage of what is already known and develop the practices for fields where it is not entirely performed.

Another important thing about this is that some tests cannot be carried out using manual testing efforts like memory leak detection, stress testing, high test coverage with a large amount of test data input etc. [20]

B. Automated test generation

Writing of the tests can be almost a third of testing task. Since the technology for automation has not advanced as expected, the automated test generation tools sometimes generate too large test set which reduces the gains of automation. Anyway, there are some good techniques and tools that are recognized as fine methods for the task. It is necessary to understand which of them are appropriate in what environments. [20]

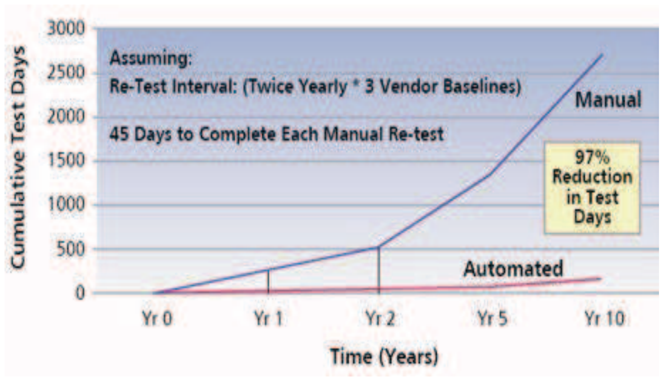

Figure 2: Example Automated Software Testing Savings over time

V. FUTURE OF SOFTWARE TESTING

A. Future trends in software development

Modern business logic requires new types of development approaches such as web based services and applications, wireless and mobile technologies, service oriented architecture, agile development methods and cloud computing. [3]

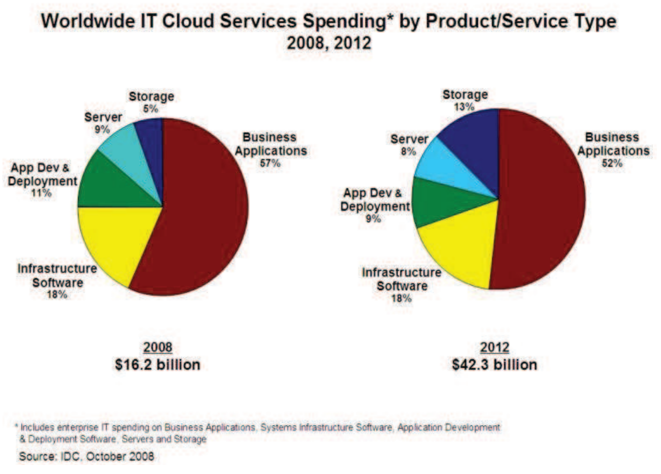

Figure 3: Worldwide IT Cloud Services Spending* by Product/Service Type 2008, 2012.

B. Future trends in software testing

New trends in the software industry suggest new trends in software testing. These trends must follow the rapid system, application development and new development strategies and technologies.

When the organization wants to develop quality software the testing should start from analysis and design phases. Stating testing must be “robotized”. It means that all testing techniques should be a part of the analysis and design automated checking tools.

The automation combined with early detection test simulators, just-in-time test cases, self-testing and self-monitoring will result in better defect and error founding in early stages of software development lifecycle. [3][4]

1) Agile software testing

Agile software testing follows the principles of agile software development. It means that agile software methodologies are based on incremental and iterative development, where everyone in the team works together towards a single goal.

Agile software product can be represented as:

$$\text{Product}_{sw} = \int_{Sprint1}^{Sprintn} TW(TDD^{CI} + PP + \sum R)\,. \quad (1)$$

In (1) *SW* means software, *TW* is a teamwork, *TDD* is a Team Driven Development, *CI* is a continuous integration, *PP* is a Pair Programming and *R* is refactoring.

Agile testing is about future testing on newly developed code so agile testers must adapt to fast changes in testing and rapid development. Testing during different stages is not only responsibility of testers, but also on developers and users.

The story testing (testing against user stories) and the story-level regression testing (regression test that is build up from the story testing from the previous iteration) are the main responsibilities of the testers. They are also involved in business analysis and story writing to ensure that all stories are testable.

Test automation is the main driving force in agile testing. Automated builds and regression testing in all development phases all rely upon automation and story-level regression testing and acceptance testing are automated in most of the agile projects.

Agile testing requires the team to develop a set of metrics and consistent analyzed data. The testing benefits are: short time for defect fixes and only few new defects.

Size metrics in agile testing can be defined as a number of regression test steps that run at each of the iterations because test size is more correlated with complexity and only features that have full regression testing at each of the iterations are counted as delivery product size.

Agile testing has a simple rule to fix every defect as soon as it is discovered. This is required because the released version to the customers in each of the iterations should not contain known bugs. Fixing bugs early helps developers to work on clean and stable code that makes new development faster. [5][6]

2) *Cloud software testing*

Cloud testing is primary intended for dynamic, distributed and component-based application that are flexible and more connected to real-time data. Cloud testing is a new approach where cloud computing environments simulate real-world user traffic. It is based on software-as-a-service (SaaS) model.

This online service provides daily operation and testing support across the internet and it can be accessed through web-based browsers and servers. [7][8][9]

Let *Ti* be the set of clouds needed for testing the software products *Pi. T* can be defined with this equation.

$$T_{Pi} = \begin{cases} \left(sm_{i,} \;\; tc_{i1}, \;\; tc_{i2}, \ldots, tc_{iy}\right), \\ tc_{i1} = \left(ts_{i(11)}, \;\; ts_{i(12)}, \ldots, ts_{i(1k_{i1})}\right), \\ tc_{i2} = \left(ts_{i(21)}, \;\; ts_{i(22)}, \ldots, ts_{i(2k_{i2})}\right), \\ \quad\ldots \\ \quad\ldots \\ tc_{iy} = \left(ts_{i(y1)}, \;\; ts_{i(y2)}, \ldots, ts_{i(yk_{iy})}\right) \end{cases} \tag{2}$$

In (2) $sm_{i,}$ means service manager of the product *Pi* belongs to the Application Services, and H is the number of products that have to be tested, (0 < i ≤ H).

tc_{ij} is one of the testing cloud of *Ti* , while $ts_{i(ju)}$ is one of the testing services of tc_{ij} and 0 < j ≤ y , 0 < u < k_{ij}-1. y is the number of clouds and number of testing clouds for product *Pi.* k_{ij} is the total number of testing services available in the testing clouds "j" of the *Pi.*

Because the scalar type test environment is provided, $T_i \cap T_j = \emptyset$, where 0< $i_1, i_2 \leq H$ and $i_1 \neq i_2$. It means that at any specific service duration, Testing Cloud or a Testing product cannot be shared by more than one product.

Three main component of this system are: Service Manager ($\{sm_{1,} sm_{2,} \ldots, sm_z\}$), Testing clouds ($\{tc_{1,} tc_{2,\ldots,} tc_x\}$) and clones of Testing Services ($\{ts_{11}, \; ts_{12}, \ldots, ts_{1k_1)}\}$, … , $\{ts_{x1}, \; ts_{x2}, \ldots, ts_{xk_x)}\}$) that each runs on different machines on the network.

The Service manager is responsible for tracking and coordination of core activities and each cloud is responsible for providing a specific output standard that every task must conform to. [8]

Cloud online service has some benefits such as: minor investments in installing and maintaining testing environments for customers, online market for testing providers and customers and software testing as an online service that can be made for about ten working days that leads to saving time and money. [7][9]

3) *Search-based software testing*

Search-based software testing is a meta-heuristic optimizing search method where the testing task is automated and the data are generated automatically.

The simplest algorithm is random search where input data is generated randomly until the test requirements are satisfied. Random search is very poor method when the input domain is big because it provides only few inputs compared with the size of the input domain.

Better way for meta-heuristic searching is a problem-specific fitness function and it needs to be defined different for every new problem. The inputs closer to the test requirements are signed with higher fitness value. High climbing is an algorithm that utilizes this function and it starts at random point in the search space and continues with the point with higher fitness value in the space neighbourhood. [10][11]

The main problem with search-based software testing is handling the execution environment where the software under testing exists. For example, the network file system or operation system interactions may not be handled.

For solving the "flag" problem, when a branch predict consists of Boolean value the "flag" and has only two branch distance values: true and false, there is "Testability Transformation" method. It produces a temporary version of the software program that is being tested and it removes the unwanted features for Searched-Based test data generation. For example, a Boolean variable can be replaced with a condition that leads to making the flag value true. [10][11]

Testing Transformation usually improves the reliability of test data generation. When there is a program with complex data, then search is performed with both transformed and untransformed program's versions. There is no limit to the number of transformed versions so it can be used for improving the reliability of the program. [10][11]

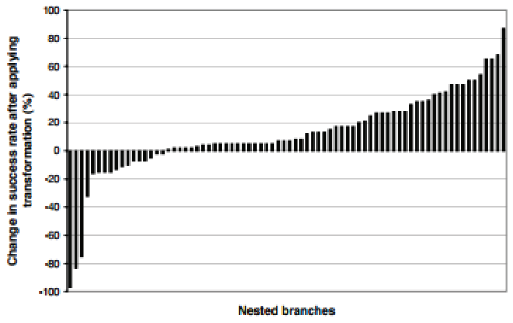

Figure 4: Results with testability transformation. The success of test data generation improves the success of test data generation

VI. EDUCATION OF TEST ENGINEERS AND TEST DESIGNERS

Today, the need of professional software testers is huge. However, there is no real support from universities and colleges in realising courses for professional testers.

ITSS (IT skill standard) has a “testing skill” as an item in its function skills required for IT specialist, but the testing category has no job for testing specialists.

Since test engineers are important category in embeded software testing, ETSS (Embedded software skill standard) listed “test engineer” as a job category.

Japan Electronics College offered a Software Test Design course for becoming a test specialist. This is 2-year course and it provides skills such as testing environment development skill, test management skill, performance evolution skill, quality management and case studies. Required skills for this course include computer knowledge in general, programming and network skills. Results show that many companies are interested in graduate students who finished this course.

The most important thing for test engineers is to understand the product and also global organization perspective and strategy. Since resources in software testing are one of the main factors for evaluation the quality risks of products, it will be necessary to include these kinds of courses in universities and graduate schools. [12]

VII. SUGGESTIONS AND NEW TESTING SOFTWARE IDEAS

Regarding the fact that software testing is a mental activity and software can be experimented, in the future the existing automation tools can have an artificial intelligence, i.e. robots can help the software testing.

Future testers must be more technically professional to develop test stubs and drivers that interact with more programming languages that will work with more complex data sets and will be integrated with agile teams. Tools must be highly complex to support the testing of these data sets.

Testers in the future must be professionally educated and accredited with computer science degree and even with some additional courses. They must be concentrated on adding value to the business software development lifecycle.

Software testers will operate in position where the combination of their knowledge, creativity and ability for decisions about testing criteria will be the main reason for company success.

The role of professional testers will become more essential. Testing and quality assurance will take part of modern technologies to satisfy the goals regarding the new products quality on the market rapidly and at minimal risk. This kind of system interaction will lead to minimizing the risks and increasing the benefits of business processes in the companies that is also the main reason for their existence.

REFERENCES

[1] Mark N. Frolick and Brian D.Janz, ” Cycle Time Reduction in Software Testing”, The University of Memphis.

[2] Tom Mans and Serge Demeyer, “ Software evolution”. Verlag Berlin Heidelberg, 2008.

[3] AppLabs, “Future of software testing”, 2008.

[4] Nick Jenkins, “A Software Testing Primer, An introduction to software testing”, Creative Commons, 2008.

[5] The Magazine for Professional Testers, Testing experience- Agile Testing, Germany, 2009.

[6] David Talby, Arie Keren, Orit Hazzan and Yael Dubinsky, “Agile Software Testing in a Large-Scale Projects, IEEE Computer Society, 2006.

[7] Leah Muthoni Riungu, Ossi Taipale, Kari Smolander, “Software Testing as an Online Service: Observations from Practice”, IEEE, 2010.

[8] T. Vengattaraman, P. Dhavachelvan, R. Baskaran, “A Model of Cloud Based Application Environment for Software Testing”, IJCSIS, 2010.

[9] Jerry Gao, Xiaoying Bai, Wei-Tek Tsai, “Cloud Testing- Issues, Challenges, Needs and Practice”, an international Journal (SeiJ), Vol. 1, no. 1, 2011.

[10] Phil McMinn, “Search-Based Software Testing: Past, Present and Future”, University of Sheffield, Department of Computer Science Regent Court, 211 Portobello, Sheffield, S1 4DP, UK, 2009.

[11] Giuliano Antoniol, “Search Based Software Testing for Software Security: Breaking Code to make it Safer”, SOCCER Laboratory – DGIGL ´Ecole Polytechnique de Montr´eal, Qu´ebec, Canada

[12] Toshiaki Kurokawa, Masato Shinagawa, “Technical Trends and Challenges of Software Testing”, Quarterly Review No. 29, 2008.

[13]IBM Research Technical Report RC 21457 Log 96856 4/26/99 “Software testing Best Practices” Ram Chillarege, Center for software engineering IBM Research

[14] Toshiaki Kurokawa, Masato Shinagawa, “Technical Trends and Challenges of Software Testing”, April, 2008

[15] Corina S. Pasareanu, Willem Visser, “A survey of new trends in symbolic execution for software testing and analysis”, 30 August 2009

[16] R. M. Hierons, K. Bogdanov, J. P. Bowen, R. Cleaveland, J. Derrick, J. Dick, M. Gheorghe, M. Harman, K. Kapoor, P. Krause, G. Luettgen, A. J. H. Simons, S. Vilkomir, M. R.Woodward, and H. Zedan, “Using Formal Specications to Support Testing”

[17] John E. Bentley, Wachovia Bank, Charlotte N, “Software Testing Fundamentals—Concepts, Roles, and Terminology”

[18] W. R. Elmendorf “Evaluation of the Functional Testing of Control Programs”

[19] Cem Kaner, Florida Tech, “An Introduction to Scenario Testing”, June 2003

[20] Bernie Gauf, Elfriede Dustin, “The Case for Automated Software Testing”