SOFTWARE QUALITY CONTROL WITH THE USAGE OF IDEAL AND TMMI MODELS

PETR BRIS, MICHAL FRANTIS, MONIKA KOLKOVA

Tomas Bata University in Zlin, Department of Industrial Engineering and Information Systems,Faculty of Management and Economics, TGM 5555, Zlin, Czech Republic

DOI: 10.17973/MMSJ.2015_12_201565
e-mail: bris@fame.utb.cz

Our work focuses on process of software testing with the regard on his control and quality securing. Theoretical part explains basic terminology used later in our work. It explains the terms such as quality, process control management or software. Later our work describes software testing in relation with methodology of software development, quality improvement of process testing with the usage of model TMMi and modelling of processes.

Project part consist of process analysis of current state of software testing process. Identified subprocesses are modelled and inserted into Process Cards. Identified imperfections are then subjects of solution proposal for improvement of software quality control testing process. Afterwares the proposal is implemented on the current state of software testing process with the goal to achieve second level of software testing process maturity according to model TMMi.

KEYWORDS
quality control, software testing, TMMi, modelling, competitiveness

1 INTRODUCTION

High quality software and quality of the software testing process are nowadays one of the biggest advantages in competitiveness for the companies working on the software development field. Accent on the speed, low cost and good quality requires new sophisticated tools and approaches, which can be applied on technological areas of cloud applications, mobile platforms, automation, etc.

Our work aims on software testing process, which is today as much important as software development itself. Even if non-professionals often perceive it only as a necessary tool for fault detecting. This false assumption operates with the idea that software testing serves only for finding the failures. But the aim of the whole process is actually prevent the failures caused by software defect. [Felderer 2014] The outcome of this approach is a sofware product of a good quality, which is supported with the software quality testing process itself. Ability to declare certain level of quality control management, or maturity of the software testing process, increases competitiveness in the current dynamically evolving software market.

The main focus of our work is to achieve the second level of maturity, or software testing process according to the model TMMi. This goal selected by authors aims on supporting companies, which want to achieve competitiveness advantages which allow them to provide software infrastructure on advanced software markets such as for example american markets (Hypothesis H1)

Declared goal is supported by solution proposal for improving quality control of software testing process, which will lead to decrease of the number of discrepancies and related additional costs. (Hypothesis H2)

Project part of the work is structured with the application of model IDEAL developed by institut SEI (Software Engineering Institute). First (Initial) and second (Diagnosing) phase of model are based on process analysis of the current state. By this step is completed the comparation of current and desired state, identification of imperfections and definition of reasons leading to the application of amendments. Third phase of the IDEAL model is called Estabilishing phase which aims on creating project workplan derived from the analytical part of the project. The forth phase is Acting phase, which presents actual implementation of solution proposal for the improvement of quality control of software testing process, which was created in the analytical part of the project. Fifth phase called Learning is focused on the process assessment and evaluation of the process testing sofware according the model TMMi.

2 SOFTWARE TESTING – LITERARY SUMMARY

For general public is a testing process perceived as marginal activity, which is done just because of sofware functionality validation. But it is far from the truth. Software testing is not only extremely important part of quality control process, but also leads to decreasing or complete elimination of additional costs. There is a rule for software testing that later the software problems are detected, bigger is the cost for their removal.

2.1 Definition of testing

We understand software testing as a process of controled launch of software product to find out whether it fulfils specified parameters and users implicit needs. As controled testing process we understand such process which is always executed with certain plan and is preceded by planning phase where the forms of test are proposed and evaluated. Testing focuses on examinigh a software product, while collecting information about its quality, which is understood as a degree of users' fulfillment and expectations. [Roudensky 2013]

Side-effect product of collection and analysis of information during the testing process is the identification of defects, which will be described in the separate chapter. Many older publications claim that the identification of defects is the main goal of testing proces. But the evolution of the software testing process went further and today is the testing process predominantly used as a tool for prevention of defects.

Testing is the process which consists of the sequence of activities where is necessary to determine when each of them needs to be carried out. At the same time there are defined inputs and outputs from the partial operations. Testing process has pre-defined responsibility of each element, which participates in any operation in the whole process. [Dolezel 2015]

2.2 Types of tests

All tests can be sorted according to following criterions:

- According to the level of knowledge of the software code,
- According to the method of test execution,
- According to the amount of quality of the software, which is tested
- According to the current phase of development of the software.

During the testing we can work with different amount of information and knowledge which is used. According to the level of the knowledge of the software code we have three different options:

- Black box approach
- White box approach
- Grey box approach

Black box approach tests reflect the fact, that we do not know the structure of the software code and its exact functionality. During testing we only work with the knowledge of inputs and required ourputs. We do not know how the software works. The complete opposite is the white box approach, where we know how exactly the computer code is working and we know all the details of its structure. The combination of the above approaches we get the approach called grey box testing. [Galin 2004]

Second known categorisation is the classification according to the method of test execution. We can talk about manual or automatic testing process. Manual testing is executed by the person called tester, who afterwards records the test outcomes. During manual testing the tester uses his knowledge-base, creativity and thought process. On the other hand automatic testing is executed by special software which increases the speed and effectivity of the executed tests. Automatic test is executed repeatedly, very fast and flawless. Automatic testing is a very usefull complementary tool to manual testing, but it cannot replace manual testing for the reasons mentioned above. [Roudensky 2013]

Next important classification of tests is based on the amont of quality of the software, which was defined by Hewelett-Pacard quality model FURPS. The shortcut is derived from the words decribing different aspects of software quality. They are as follows: Functionality, Usability, Reliability, Performance and Supportability.

Last important classification of the tests is according their time occurence in the process of software development. In this category we have the following types of test:

- Unit testing
- Integration testing
- System testing
- Acceptance testing.

[Roudensky 2013]

Acceptance tests are usually labelled by acronym UAT which stands for User Acceptance Testing. The goal of acceptance testing is to verify, whether the tested software meets the acceptance criterions of a customer. If this is true, then software is handed to the customer and one life phase of the software is over.

During unit, integration and system testing so called regressive testing is being proceed. The aim of regressive testing is to make sure that amended part of software won't affect the functionality of the other parts of the software. Two important types of regressive testing are smoke testing and sanity testing. Smoke testing deals with the aggregate of functionalities, whereas sanity testing is focused on the specific part.

2.3 Software Defect

By the software defect we understand the imperfection which resides in the computer code or data, which is most often caused by computer programmer. The origin of the defect differs. It may be the flaw in the programme code, false structure of the software, misunderstood requirements from customer, or it may even be a planned sabotage. Software defect is in default state inactive. Activation of the defect will cause the deviation of the software from its required state. This situation si called error of the software. If the error is visible to the user, we call it software failure. [Roudensky 2013]

Situations which lead to errors or defects we regard the following:

- Software is not doing what it should do according to its defined specifications
- Software is doing what is should not do according to its defined specifications
- Software is doing something which is not specified,
- Sofware is not doing something which is not specified, but presumably it should be doing it
- Sofware is not understandable, the work with it is uncomfortable, is very slow, etc.

If we wanted to show the proportion of each type of of activity for the occurance of defect, then on the first place there is insufficient understanding of customer's specifications. 6 defects out of 10 are caused because of specifications. Then follows the defects caused by wrong structure of the sofware, which should reflect the specifications. 3 defects out of 10 are caused by this part of the software development. [Patton 2002]

2. 4 Software quality control and assurance

Software Quality Control (SQC) is focused on outputs of individual processes, where is checked whether they are in accordance with the requirements and specifications. [Roudensky 2013] As outputs we understand partial or final products, which are for example represented by software documentation, programme code, but also by defects. Basic tools used in sofware quality control are as follows: revision, inspection, testing, simulation, verification, validation and audit. Even if validation and verification are mentioned here, they are also parts of testing process.

In reality the terms verification and validation are often mixed up, which is because of their similar conception and mutual dependency. According to the international standard ISO/IEC 12207:2008 verification is the process of authentication, whether certain subprocess of software development complies with the specific requirements and pre-defined conditions. We can distinguish between verification of requirements, verification of proposal, verification of source code, verification of integration and verification of documentation. In the above international norm we can also find a definition of validation. Validation is a process of confirmation, that the product being developed is correct and fulfills intended use. So during the software development process we can face the situation where the sofware is correctly developed and gets verification, however is not accepted by customer, therefore cannot be validated.

Software quality control belongs under category called Sofware Quality Assurance (SQA). Contrary to Software Quality Control, which is focused on the final product, is Software Quality Assurance orientated on the quality of the processes themselves. And it is namely testing process which is one of many processes used during software development, which allows us to achieve a good quality product. Again in the norm ISO/IEC 12207:2008 we can find the definition of Software Quality Assurance as a process which gives us sufficient guarantee, that all the processes included in the life-cycle of the software will meet required criteria. In the figure below we can see, that testing process is on the lowes level and belongs under software quality control and at the same time is part of the software quality assurance. [Fig. 1]

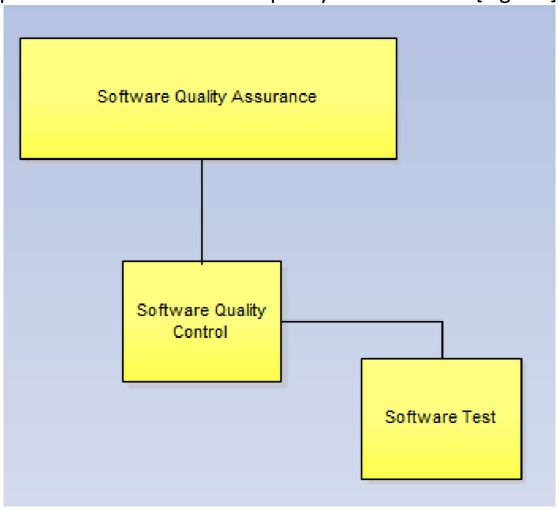


Figure. 1: Relationship between Software Quality Control and Assurance

2.5 Methods of increasing software quality

Aim of this chapter is to introduce methods of increasing software quality by way of refining of the test process. One of further mentioned methods will be later applied in the practical part of our work.

Software quality increasing represents a tool for improving quality of final software product and for reduction of costs or enhancment of all other processes in the whole software life-cycle. We can distinguish two basic approaches, which deal with software quality increasing :

1. Process maturity approach.
2. Agile approach [Sommerville 2011]

First approach is focused on process maturity. This approach aims on increasing project and process management and introduces optimised sequences. Process maturity approach is applied by set of developed models and metrics. One of the best known models used to determine and increase software maturity, organisation and capability is integral model CMMi created by the Institute of sofware engineering and process increasing. [Galin 2004]

Model CMMi is designed for testing only in a part and doesn't cover test activities in complex scale. For this reason in practical part of our work we use another model TMMi, which puts more focus on software test processes than CMMi model. Agile approach is connected with incremental development, which represents the opposite of rigorous software development introduced in previous chapter of our work. Primary characteristic of this approach is reiterative development, reduction of expenses in software process and quick responses on different requirements of customer. [Sommerville 2011]

2. 6 Increasing quality of process testing with the use of TMMi model

Key document defining TMMi model which includes requirements on testing processes is Test Maturity Model integration (TMMi) in the recent release called Release 1.0. Another important document is TMMi Assessment Method Application Requirements (TAMAR) which describes requirements for methods of process testing assessment in its most recent version called Release 1.0. Unfortunately the company did not published the assessment method itself.

TAMAR document offers two views on the assessment approach. Assessment can be executed in formal or informal way. Aim of the formal way to obtain certification TMMi Foundation which proves achieved maturity level. Such certificate as well guarantees compliance with the norm ISO/IEC 15504-2. In order to get such a certification, company needs to have accredited assessment method or they can use external company with necessary accredited assessment method. [Goslin 2012]

3 USED METHODS

Method is structured using the IDEAL model developed by SI Institute (Software Engineering Institute). Initial and Diagnosing phase of the model is based on actual state analysis of processes. This way a comparison of actual and required state is performed including errors identification, defining the reasons as well. All such steps lead to the application of changes.

Third phase of Ideal model named Establishing is aimed to form project working plan and continue the analytical part of the project. Fourth phase, Acting, is about the proposed solution implementation regarding quality management of software testing processes, which was developed during the analytical part of the project. Fifth phase,Learning, is about project results assessment using the evaluation of the testing, according to the TMMi model.

The analytical part of the project such method of analysis will be used, which will divide process of software testing into partial components. Technique used for data collection will be again the process analysis.

Next part of the work will use the synthesis method with the inductive approach, where the findings acquired during analytical part will be used to propose the solution of better processes of software quality testing. Applied analytical technique during this part of the project will be the assessment of processes knowledge according to TMMi model.

4 ANALYSIS OF CURRENT SOFTWARE TESTING IMPERFECTIONS

Actual state of software testing can be sorted into four categories as follows:

1. Test Planning
2. Test Development
3. Test Implementation
4. Test Execution

4.1 Test Planning

Test Planning Process can be divided into four sub-processes:

- defining goals and extent of testing
- analyzing risks of testing
- defining approach toward testing
- other tasks

Principal task of software testing is the prevention of errors.

Extent of testing is based on plan of the project, where the time, expenses and source limits are defined. Using those initial information's the Test Manager is completing the list of tested and untested parts of software and its attributes including the reasons why certain part is not included in testing.

Visible problem of this task is that the priorities of each position on the list of tests extent are not defined. Priorities than shall be based on needs of interested parties and identified product and project risks, which are the output of analysis sub-process risks. Another distinct problem is that testing metrics are not defined. Unless basic values of measuring are not defined, sufficient testing management cannot be performed and software quality management as well.

4.2 Test Development

Test Development process shall define the ways and means of software testing procedure. As an entry to this process serve a sub-process of development studies analysis, Release Study. Content of Release Study is formed by partial specifications by users, defined within the second stage of software development, named Specification. Using the development studies the conditions for testing are defined. They are later mapped by Test Case procedure and Testing Scenario.

The main tasks in developing the tests is the plan of testing events and scenarios. Output of such assignment are artifacts or documentation named Testing Events and Testing Scenarios.

After analysis no other serious imperfections, other than missing metrics were received.

4.3 Test Implementation

Goal of Test Implementation is to prepare testing data and testing environment for testing. As a visible problem the missing audit of database environment transfer among the AP employees where the proper condition of testing environment was not verified. Such task could be performed using smoke tests, verifying basic functionality of testing environment.

Testing data are for partial testing tasks and testing scenarios always prepared manually. This is done by preparing the real data, where the sensitive or personal data are replaced by fictive ones. Other possibility is to write completely new data, but this is more time-consuming and real data structure is not preserved.

Output products of Test Implementation process are Testing Data, Test Timetable and Testing Set. Testing Data are based on prioritizing and organizing Testing Events and Testing Scenarios. During the Test Implementation Phase a final analysis closing is again manifested, that metrics are not bound to the process, thus not allowing the conceptual process management.

4.4 Test Execution

Test Execution is the crucial sub-process of all the testing process. The aim of this sub-process is to perform manual tests according to the developed Test Scenarios. Responsible for performing tests is Tester (Test Analytic). Tests begin at the moment when the testing environment is ready with the deployed version of software. Than the input criteria are verified, serving the test start. The task of verifying criteria required from the software product is called Acceptance Testing. It is based on Smoke Tests technique, authenticating the general functionality. Output is a Checklist, General Functionality Test.

Even if the tests results are recorded into partial Testing Scenarios, continuous report about overall testing situation is not written. Aim of Test Report shall be the continuous information about total number of Testing Scenarios, their results and tests quantity needed for finalizing the tests time-schedule.

When a valid error is reported, the life-time cycle of the error started. Defect Report is recorded using specialized software, in care of software administrator. Defect Report contains description of non-functional part of software, leading it to defect, environment, where the defect was observed, its importance, priority etc. Using this tool means monitoring the state of defect resolution.

Described missing aspects of processes are the result of lacking process of monitoring, managing and assessing the testing results. Non-existent processing areas are the results of missing metrics in a whole testing process. They were ignored and without proper attention. In Tab 1 there are missing aspects of actual software testing.

Test Planning	Test planning doesn't reflect the changes, which happens during the software testing process. On the bases of identified needs of involved subjects no prioritisation of testing extent is happening. Software is tested by the approach based on requirements. Even if the risks are identified, testing is not working with the approach based on potential risks.There are no metrics for process testing control defined.
Test Development	Missing process metrics
Test Implementation	No verification of database environment immediately after their handover. Missing process metrics.
Test Execution	No Test report for describing the current state of software testing is being generated. Missing process metrics.

Tab. 1. Identified imperfections in the current software testing process

Apart from imperfections noted further in Table 1, key problem are missing subprocesses of process control and assessment. Those subprocesses are from the quality control point of view essential and without them it is not possible to achieve the second level of software testing process maturity in the model TMMi. [Tab. 1]

In the next part of our work we propose the elimination of identified imperfections in the partial subprocesses of the software testing process with the implementation of missing subprocesses. Missing metrics will be as well implemented, because without them is not possible effective control of any process.

5 PROPOSAL OF NEW WAY OF SOFTWARE TESTING

5.1 Amendment of current subprocesses of software testing process

Following subchapters present the proposal of amendments of current subprocesses of software testing.

Test Planning

The first amendment in the subprocess of test planning will be simplifying of Test planning but with the maintaining of company Test policy and Test strategy. Test planning will clearly define the main target of testing, extent of testing and method of testing (strategy), risks, necessary resources, artefacts, testing schedule, responsibility matrix, metrics, identification of input, output, suspensive and recovery criterions. Current Test Planning will be revised and during the work will be used the standardised form described in the norm ISO/IEC/IEEE 29119-3 Software and systems engineering – Software testing – Test Documentation. Newly created test planning will be presented to all involved subjects for checking and verification of its content. Test planning will be regularly updated to reflect potential deviations in the test planning procedure.

Important part of test planning process is estimation of the work difficulty of testing with the help of the metrics based approach. Consequently there will be resources identification and testing schedule compiled.

Last amendment is prioritisation of the test extent. Prioritisation won't be done by involved groups, but will reflect testing based on requirements. Needs of involved subjects will be identified in the first phase of software life cycle, which will ensure projection of those needs into the test extent. Tester will consequently present the extent to involved subjects for approval.

Test Development

Subprocess which is dealing with the test development was found without any serious imperfectin, which would lead to the process amendment. Missing metrics will be inserted into the process Card and will be monitored, assessed and interpreted to involved subjects consequently.

Test Implementation

Test implementation subprocess will be extended by activity dealing with the checking of database environment immediately after they handover. This activity will be executed by Tester (Test Analyst), who verifies the correctness of datebase environment via proper smoke tests. Additional numbering of used revision of database environment will serve as supportive visual control and must match with the number of revision mentioned in requirements for testing environment. Logical structure of current subprocess is sufficient. Missing metrics will be inserted into the process Card.

Test Execution

During the subprocess of test execution new testing control subprocess for report creating will be introduced.

This report will be monitoring the actual state of the subprocess during the test execution. For the creation of Test report will be used standartised form described in the norm ISO/IEC/IEEE 29119-3 Software and systems engineering – Software testing – Test Documentation. Test manager is taking responsibility for the Test report. Metrics will be inserted into the process Card

5.2 Creation of new subprocesses of software testing

During the process analysis was identified missing subprocesses in the process of testing, Creation of those missing subprocesses is described in the following subchapters.

Test Management

The aim of new subprocess will be securing the accordance between executed activities and test planing. Test management will be based on continuous analysis of testing process and its results, monitoring of testing process and identifying of input data for process test metrics. Possible amendment in case of identified imperfections will be executed in two different ways:

- Testing plan amendment
- Arrangement of corrective measures

Part of the test management process will be the creation of so called Test reports. Input data will be collected via mapped metrics of testing process. They will be continually processed during all the testing. Metrics, which will be monitored through testing process are listed in the previous subchapter of our work. The outcomes will be sorted and interpreted to all involved groups.

Monitoring throughout all the testing process brings as well observing of the defect lify-cycle, which are recorded during the Test Execution subprocess. Apart from defect life-cycle status of testing environment is monitored as well. Communication has to be a strong feature of Test manager who makes sure that the all information are distributed to all involved groups.

Subprocess of test management is based on the monitoring of the test progress, during it data are being collected. Data are consequently evaluated, which can cause the decision of amendment of priorities based on risk occurance, amendment of testing schedule because of unavailability of testing data or environment, failure of software supplier during the defect removal etc. During the testing phase are constantly recorded output artifacts of the process which are shown on the process Card. The person responsible for the test management subprocess is a Test manager.

One part of test management certainly would be a motivation of testers, which has an influence on the quality of their work.

For quality control will not be created model of processes because of its complexity and difficulty of subprocesses and their influence on all other processes in the software testing process. [Fig. 2]

Process Card	
Number and name of the process	P5 – Test Management
Target of the process	Target of the process is to secure compliance between testing plan and really executed activities
Owner of process	Test manager
Customer of process	Involved groups
Inputs of the process	Test plan, Test metrics, Results of tests, Defect report

Process activities	
Number of activity	
1.	Test monitoring
2.	Collecting the data from the testing
3.	Recording report of testing status
4.	Amendment of identified defects
5.	Motivation of test team members

Outputs of process:	Test report, test plan, minutes from the meetings

Process metrics :	Amount of cost spend on testing, value of identified defect, quality level of testing process

Figure 2: Test Management Process card

Test Evaluation

Test Execution subprocess is the last subprocess in the current structure of testing process. In case of acceptation the whole process was over. Test acceptation was based on the comparison of output criterions of software product in the test plan and the actual data gained during general functionality test. By creation of a new Test Evaluation subprocess we estabilish the control of output criterions gathered during the test and whole process of testing will be closely interconnected. Output criterions will be determined in test plan. The person responsibile for this process is Test manager. Output artifact of this Test Evaluation subprocess will be Summary Test report

Test Evaluation of developing software is taking place in the very end of test execution, when acceptation testing is checking quality of tested software product. This activity will be transferred into the new subprocess Test Evaluation. This subprocess will be preceded by checking the quality of testing process. Test Evaluation will consist of comaprison of hard and soft output criterions described in test plan. In case of the failure to meet soft criterions the process will continue until the final acceptation of software. However if the hard output criterions will not be met, the test and the acceptation procedure cannot continue until the problematic output is fixed.

Rearrangement of subprocesses aims on test evaluation and acceptation, which checks the quality of software product. Implementation of this activity can cause the shift of current time milestones in the schedule because of uncompliance with hard output criterions. In case it will not be possible to secure, that hard criterions will be meet in adequate time, involved groups are to be contacted with the request for comment and request for accepting the exception, that during acceptance process this step can affect the software quality in form of defect, which can shift the time milestones in test schedule even further and therefore additional expenses may be generated.

An example of hard output criterion is Validation of sofware project. During validation process no defects are found, but there is discrepancy between what involved groups expected and what was the actual output of the testing process. Remedy of this criterion will be so called Change Request, which will be handed to to software supplier for incorporating. Change Request generates additional expenses no only on the supplier side, bud also on the side of XY company, because of shifting scheduled time of launching the software in real use. This aspect can for example cause financial loss because of unperformed business transactions. After test evaluation and acceptance testing the testing part is coming to its end. Responsibility for this activity is again carried by Test manager.

Finalisation of testing begins with checking of completeness of testing. Aim of this activity is in verification, wheter all the activities of testing process were completed. It usually consist of checking wheter all planned tests were executed, and reported defects, which were made during all the process of testing, were repaired or amended. Next activity is the storage and backup of all recorded output artifacts from the testing process. Aim of this activity is to keep and store all the information for the future use on new software projects. This archived artifacts are stored on shared disposal site, so all the involved groups can access it. Accessibility of all the data needs to be controlled by different levels of access rights so every involved subject will have access to proper information.

Throughout the whole process of testing the involved groups need to meet regularly so all the information about testing will be distributed properly. As well there will be introduced so called Retrospective meeting which will focus on sharing the experience between the members of test team after the finalisation of test, presentation of the work done, and the evaluation of the competitions, which will be part of the motivation factors within the testing phase. [Fig. 3]

Process Card

Number and name of the process	P6 – Test Evaluation
Target of the process	Test evaluation and acceptation of quality of software product with consequent finalisation of whole testing process
Owner of process	Test manager
Customer of process	Involved groups, Testing team
Inputs of the process	Test plan, Evolution study, Test Case, Test Scenario, Test Metrics, Test Report, Minutes of the meetings, Test data, Test sets, Results of tests, Defect report, General functionality test

Process activities	
Number of activity	
1.	Test evaluation
2.	Test Acceptance
3.	Verification of test completeness
4.	Archiving and storage of output artifacts
5.	Presentation of output artifacts

Outputs of process:	Summary Test report

Process metrics :	Level of work difficulty deviation, time milestones deviation, level of quality of testing process

Figure 3: New card of Test Evaluation Process

6 RESULTS AND PRESENTATION OF PROPOSED PROCEEDINGS

Overall evaluation of process maturity testing was conducted in five declared process areas of second level of TMMi model. Overall result was presented via Summary report called TMMi Report.

Evaluators went through each component. Each of this components was marked and the result was presented in evaluation card or process area. In cases where evaluators differ in evaluation of particular component, the meeting was hold where efforts were made to succeed on concordance evaluation scale. Particular components of TMMi for second level model were placed into the evaluation card .

Each particular component could reach the grade from 0 to 100% with the evaluation mark F, L, P, N, NR or NA. Evaluation scale was used from TAMAR documentation about evaluation of matrity of components for model TMMi and this scale is essential for evaluation of the particular components. [Tab. 2]

Evaluation	Description	Conditions
F	Fully Achieved	85-100% Achieved
L	Largely Achieved	50-85% Achieved
P	Partially Achieved	15-50% Achieved
N	Not Achieved	0-15% Achieved
NR	Not Rated	Impossible to evaluate
NA	Not Applicable	Not acceptable for the company

Table 2: Evaluation scale for model TMMi

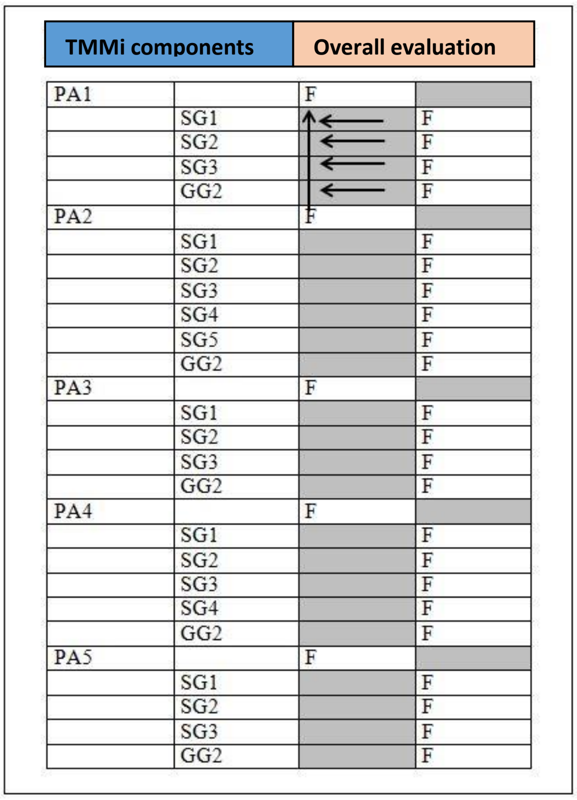
F mark requires full compliance of evaluated component according to reference documentation. Mark NR doesn't discriminate the component but the condition is automatically considered as unsatisfied. On the other hand mark NA means that the condition not unapplicable for a company and is excluded from the evaluation. Company needs to present reasons why it was excluded form evaluation.

Evaluators started with filling the grade on the level of specific and generic practics. The lowest grade were then transferred into the field for specific or general target. The lowest grade from the level of specific or generic targets was then transferred into the field or process area. To achieve the second level of process maturity of model TMMi must all the process areas have mark F (Fully Achieved). Evaluation of particular process areas is found further in the text.

6.1 Summary report

Final part of evaluation demonstrate the fact, that second level of software testing process maturity in TMMi model was achieved. [Fig. 4]

The goal of companies providing software infrastructure is to succeed with their products on advanced markets (such as american market, which requires maximal degree of reliability of the products). In order to achieve this, software companies have to reach the competition advantage via software testing process maturity (quality). The aim of our work was achieved, because testing process after implementation or solution proposal has reached second level of software testing process maturity according to TMMi model. Hypothesis H1 turned to be correct. This hypothesis predicted competitive advantage and business success at the US market. The condition for such a success was to increase the the quality of software testing process according to model TMMi. The company applied the product during its entrance at the US market in April 2015, which can be seen in fig. 2, which demonstrates company sales increase at the US market. [Fig. 5]



TMMi components		Overall evaluation	
PA1		F	
	SG1		F
	SG2		F
	SG3		F
	GG2		F
PA2		F	
	SG1		F
	SG2		F
	SG3		F
	SG4		F
	SG5		F
	GG2		F
PA3		F	
	SG1		F
	SG2		F
	SG3		F
	GG2		F
PA4		F	
	SG1		F
	SG2		F
	SG3		F
	SG4		F
	GG2		F
PA5		F	
	SG1		F
	SG2		F
	SG3		F
	GG2		F

Figure 4: Overall evaluation for second level of software testing process maturity in TMMi model

Secondary goal of our work was the decrease of the number of defect reports, which were recorded and were removed during the pilot project. In the fig. 6 is obvious, that the number of defect reports was decreasing until July, when there was zero recorded defects. The number of repaired defects had decreasing tendency as well. This demonstrates that new process of software testing control gradually leads to zero number of recorded and repaired defects, which proves the validity of hypothesis H2. [Fig. 6]

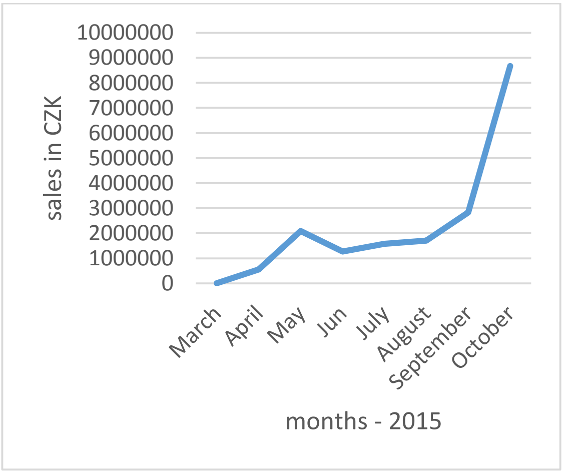


Figure. 5: Sales growth in the US market

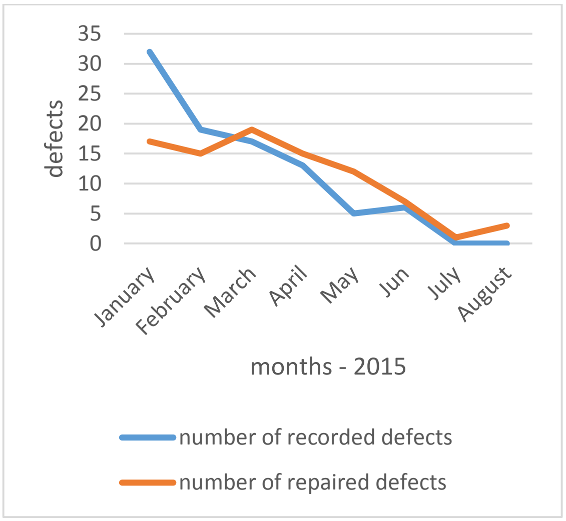


Figure 6: Summary of defect reports

7 CONCLUSION

Our work is focused on quality of software testing process with the goal to propose and implement the improvement of this process in the way, that this software testing process can reach the second level of software testing process maturity according to TMMi model requirements.

Key area of analytical part is process analysis of the software testing process. This was based on verbal documentation and introducing of the Process Cards with consequent modelling of subprocesses occuring in the test process. At the end the main imperfections and defects coming out of process analysis were summarised.

The project part is focused on the solution proposal for improvement of the quality of the software testing process with subsequent realisation of the proposal into process structure. Authors of this work used their experience and knowledge which they gained during taking part on other software projects in the position of testers or test analytics.

Aim of the work was to design improved software testing process, which can reach the second level of software testing process maturity according to TMMi model in order to achieve competitiveness advantage, which will allow the software company to enter on advanced markets (accomplishment of Hypothesis H1). Second goal (Hypothesis H2) was as well accomplished. Implementing of newly tested software caused decrease number of Defect reports as well as unauthorised defects.

Of course, there are other goals which could be accomplished if we would continue working on quality control of software testing process. Higher maturity grades in TMMi model can be achieved, as well as accredited evaluation methods can be reached.

CONTACTS
doc. Ing. Petr Bris, CSc.
Tomas Bata University in Zlin
Faculty of Management and Economics
Mostni 5139, 760 01 Zlin
telephone: +420 602 717 032
e-mail: bris@fame.utb.cz
http://www.utb.cz/fame

REFERENCES

Book:
[Dolezel 2015] Dolezel, M., Buchalcevova, A., Test Governance Framework for contracted is development: Ethnographically informed action research. Information and Software Technology, 2015. ISSN: 0950-5849
[Felderer 2014] Felderer, M., Ramler, R., Integrating risk-based testing in industrial test processes. Software Quality Journal, 2014. ISSN: 1573-1367
[Galin 2004] Galin, D., Software quality assurance. New York: Pearson Education Limited, 2004.
ISBN 02-017-0945-7.
[Patton 2002] Patton, R., Software testing (in Czech). 1st ed. Praha: Computer Press, 2002. ISBN 80-722-6636-5.
[Roudensky 2013] Roudensky, P., Havlickova, A., Software quality control: guide testing (in Czech). 1st ed. Brno: Computer Press, 2013.
ISBN 978-80-251-3816-8.
[Sommerville 2011] Sommerville, I., Software engineering. 9th ed. Boston: Pearson, 2011.
ISBN 978-013-7053-469.
WWW page:
[Goslin 2014] Goslin, A., TMMI Foundation. TMMi Assessment Method Application Requirements: TAMAR, 2014. Available from:
http://www.tmmi.org/pdf/TMMi.TAMAR.pdf
[Zelinka 2014] Zelinka, B., Unicorn systems a.s., Software testing (in Czech), 2013. Available from: http://d3s.mff.cuni.cz/teaching/commercial_workshops/previous/1213/zelinka-zajisteni_kvality_softwarovych_produktu.pdf