Annales UMCS Informatica AI XI, 4 (2011) 21–32
DOI: 10.2478/v10065-011-0037-0

Annales UMCS
Informatica
Lublin-Polonia
Sectio AI
http://www.annales.umcs.lublin.pl/

Reliability of feedback fechanism based on root cause defect analysis - case study

Marek G. Stochel[1]*

[1] *Motorola Solutions*
Kraków, Poland

Abstract – The case study presented in this article focuses on the common pitfall in which many defect prevention activities fall, caused by inconsistency of data. Before any collected Root Cause Defect Analysis (RCA) data can be used reliably, an analysis of the variation due to the measurement system itself such as a Gauge R&R study, should be performed. This allows to understand of the integrity of the data in terms of data consistency and validity. The RCA data collection process chosen for this case study concentrates on the RCA assessment performed by engineers independently.

1 Introduction

Understanding software (SW) quality provides vital information contributing to planning future effort for development and maintenance of software artifacts. SW quality parameters are measured throughout the SW product lifecycle both qualitatively and quantitatively. This information, influencing e.g. staffing and testing planning, can be gathered both as objective data (size of software artifacts, defect count, test case coverage, etc.) and subjective data (importance of features, modification risk, etc.). An application of using this SW quality data has been piloted and successfully implemented for test prioritization of a large SW product embedded within a mission critical system. It is therefore imperative that the data is reliable and its integrity can be trusted [**1**]. Very importantly, we must never underestimate the impact of human factors when gathering subjective data. One example is the method of measuring SW development productivity by counting Lines Of Code (LOC), which potentially, may introduce a counter productive effect the developer contemplates: why should my code

*marek.stochel@motorolasolutions.com

be optimal if my LOC productivity decreases?, see [**2**]. A second example focuses on the RCA itself which is more subtle in nature and will be discussed in this paper.

2 Relation to other works

In [**3**] Leszak et al. focused on defect categorization, and further improvement actions, rather than on the collection process itself. A list of randomly chosen defects was made based on the predefined criteria to obtain general understanding of SW defects and their root causes. The study consisted of separate teams focusing on the subsets of defect data. Consistency checks were performed and the analysts were asked to provide reasons for atypical behaviour or to correct the classification if it was due to misinterpretation of terms [**3**].

However, the question remains what should be evaluated as an atypical behaviour? If there is a project completion time pressure placed upon the developer, this may lead to processing non-conformities to cut corners. An engineer would avoid any additional effort directly unrelated to the software development process. Moreover, from the psychological perspective pressure on altering results triggers two very powerful influence mechanisms [**4**]: power of authority – researchers know better what atypical behavior is, and social proof – the work of others was correct. On the other hand, identified outliers should be analyzed for potential measurement error and removed from the analysis if an error exists. Altering measurements to conform to the rest of the data should not be done, as residual diagnostics could indicate possible issues with the constructed model [**5**].

In this article, the analysis of the agreement among engineers is presented, where exactly the same defects were evaluated independently. Assuming the conclusions about data consistency provided by independent engineers can be generalized to the results given by independent teams this case study presents the risk in data consistency incurred in a collection process.

There are several articles focused on RCA e.g. Orthogonal Defect Classification proposed by Chillarege, et al. [**6**], Beizer's defect taxonomy [**7**], where the focus is rather on analysis of results than on data consistency and collection process.

3 Background

There are several questions which emerged during the root cause defect analysis process, which led to this study. What is the degree of differences among engineers in terms of experience, knowledge, specialization? How does it influence the data collection process? What can be done to diminish inconsistency and to build on the synergy, and how can this be achieved? And the most important question: what is the value of the data used as the feedback mechanism for planning of test process and maintenance effort?

The information describing software artifacts could take basically two forms:

- Objective – measurable via objective criteria (collection process is easy to automate),
- Subjective – the criteria are subjective in nature, based on human assessment.

This article focuses on assuring data integrity in the Root Cause Defect Analysis (RCA) process. An experiment was conducted to check consistency of the RCA data, which is subjective in nature.

All approaches to use the data in the common process should ensure that the results are consistent and reproducible. An experiment was conducted to check engineering assessment consistency i.e. attribute agreement analysis among operators.

4 Problem Description

Experiment was conducted in an environment governed by the following process.

Software releases of a given product are consecutively managed that is, for each new release a direct predecessor becomes a baseline.

Each major software release predicates a unique entity in the defect tracking application.

Each new defect is assessed for in which release it was sourced, and what releases are impacted by this defect. The target release for the defect fix is established for either the current main release and/or request for patches on previous releases. The defect fix is performed, inspected (e.g. Fagan's inspection [**8**]), and then tested on all releases where it is to be introduced.

The root cause analysis process is typically performed on the defect based on the established predefined criteria (e.g. orthogonal defect classification, Beizer's, various internal specific classifications). In this case study an internal specific RCA classification method was used.

Based on the data sets from all the root cause defect analyses, preventive actions are planned (according to the CMMI guidelines for the optimizing process [**9**]), and the corresponding testing processes are adjusted (see [**1**]).

This creates a typical feedback-loop system, see below.

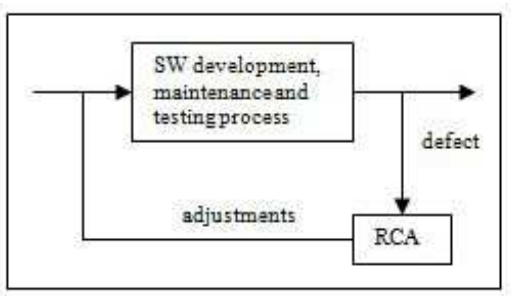

Fig. 1. A feedback-loop system using RCA.

The testing process uses the RCA data as an input for further prioritization of test cases. Any data which is used was evaluated against its validity and consistency. Any defect which is raised and being tracked in a tool is assumed to be properly assessed, fixed, and correctly root caused to avoid similar issues in the future. This activity is assigned as a responsibility to a skilled engineer who is responsible for fixing the defect. An experiment was conducted because the quality of such assessment, and its reproducibility were questioned.

5 Attribute Agreement Analysis

Each resolved defect is assessed in three distinct categories:

1. *Root Cause* – process root cause(s), why the defect was introduced in the first place
2. *Screen Failure* – reason(s) why the defect was not detected earlier during the testing process
3. *Problem Type* – engineering root cause(s), type/classification of the defect.

The data in all of the three categories are nominal attribute data (i.e. the outcome has two or more possible levels with no natural ordering). Six engineers were assessing twenty defects each. The assessments were performed independently of each other. The study was the first of the three independent studies. Each of them was run by entirely different teams focused on different software artifacts.

In **[10]** it is recommended to use 50 – 100 "parts" (distinct items being measured) in the attribute Measure System Analysis (MSA). However, in the experiment the quantity was limited to twenty items (defects) that were analyzed by one team. The reason for such an approach is that it is a very time consuming process, requiring detailed investigation of defect details. Later, the study was repeated in two more teams increasing the total number of defects analyzed to 60. However, for drawing conclusions, each of the studies has to be analyzed separately (different operators were engaged).

Reproducibility could not be analyzed because the assessment was treated as a destructive measurement. When an engineer spends a lot of time assessing the defect he remembers the result provided. From the short time perspective it is recall rather than reassessment.

Additionally, consideration was made if the information provided by the experienced technical lead can be treated as a "standard", to which the rest of appraisers were compared.

The results were assessed "Between Appraisers". In other words, do the appraisers' measurements agree with each other?

The kappa statistic test (see **[11, 10]**) was used to assess a degree of absolute agreement among ratings. It is worth mentioning that:

- If kappa = 1, agreement is perfect,
- If kappa = 0, agreement is the same as would be accepted by chance,
- If kappa $<$ 0, agreement is weaker than expected by chance.

Kappa > 0.9 indicates acceptable agreement; kappa < 0.7 indicates poor agreement and measurement system for defect assessment needs improvement.

6 Results

For all three categories of RCA (*Process Root Cause, Screen Failure, and Engineering Problem Type*) the data was analyzed and the following results were obtained in each of the three perspectives: two "standards": measuring against the person who fixed a defect, measuring against the very experienced technical lead, and the last perspective – measuring between engineers (appraisers), with no standard defined.

6.1 Problem type

Engineering Problem Type assessment is presented in the following figures.

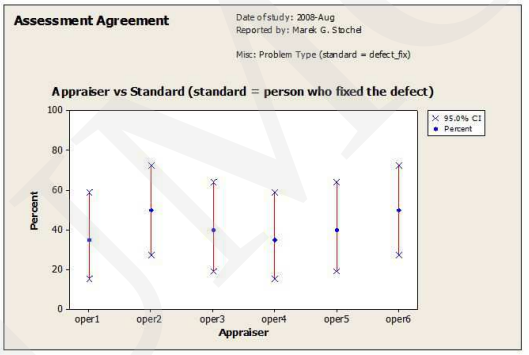

Fig. 2. Problem Type. Assessment agreement of appraiser vs. standard (the engineer who fixed the defect).

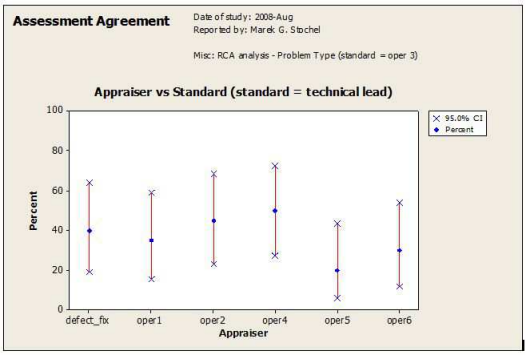

Fig. 3. Problem Type. Assessment agreement of appraiser vs. standard (experienced technical lead).

Problem Type, Agreement Between Appraisers (no standard)

Assessment Agreement

# Inspected	# Matched	Percent	95 % CI
20	1	5.00	(0.13, 24.87)

Matched: All appraisers' assessments agree with each other.

Fleiss' Kappa Statistics

Response	Kappa	SE Kappa	Z	P(vs > 0)
Box Component Interface	0.134053	0.0487950	2.7473	0.0030
Data Definition	0.132912	0.0487950	2.7239	0.0032
Documentation	-0.007194	0.0487950	-0.1474	0.5586
Error Handling	0.292929	0.0487950	6.0033	0.0000
Logic / Algorithm	0.177350	0.0487950	3.6346	0.0001
Other	0.013267	0.0487950	0.2719	0.3929
Performance / Capacity	-0.014493	0.0487950	-0.2970	0.6168
Standards Non-compliance	-0.007194	0.0487950	-0.1474	0.5586
Third Party / Tools	0.564603	0.0487950	11.5709	0.0000
Timing / Race Condition	0.071310	0.0487950	1.4614	0.0719
User Interface	0.586543	0.0487950	12.0206	0.0000
Overall	0.242720	0.0206930	11.7296	0.0000

Fig. 4. Problem Type. Assessment agreement between appraisers (no standard defined).

6.2 Root Cause

Process Root Cause analysis is presented in the following figures.

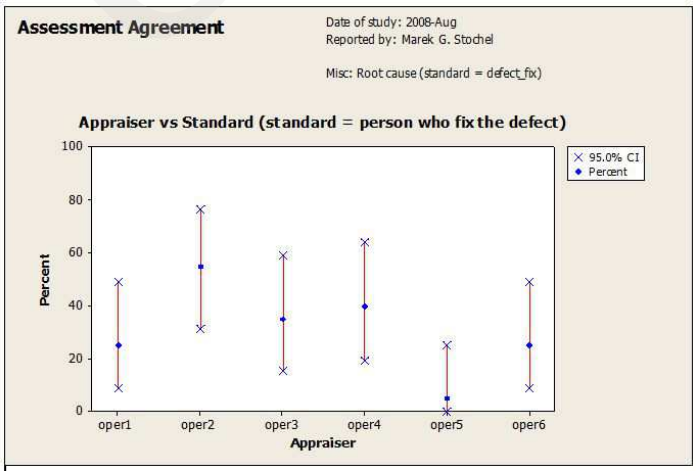

Fig. 5. Root Cause. Assessment agreement of appraiser vs. standard (the engineer who fixed the defect).

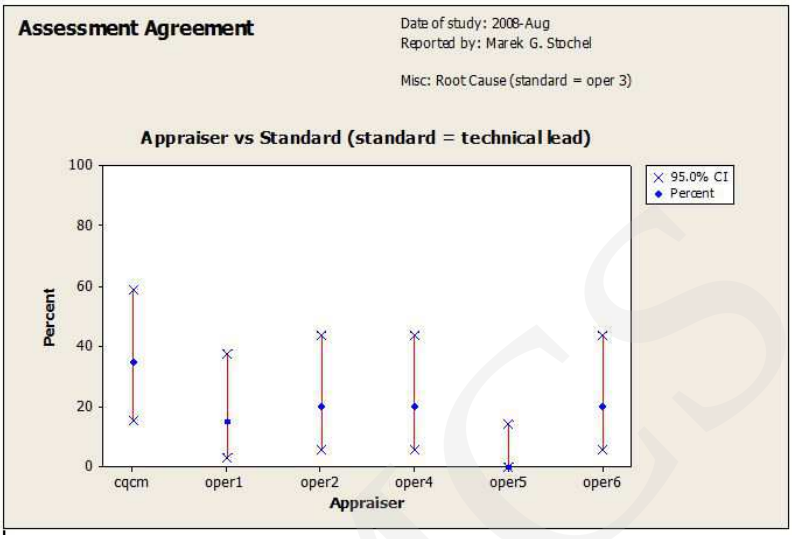

Fig. 6. Root Cause. Assessment agreement of appraiser vs. standard (experienced technical lead).

Root Cause, Agreement Between Appraisers (no standard)

Assessment Agreement

# Inspected	# Matched	Percent	95 % CI
20	0	0.00	(0.00, 13.91)

Matched: All appraisers' assessments agree with each other.

Fleiss' Kappa Statistics

Response	Kappa	SE Kappa	Z	P(vs > 0)
CM Process	0.091646	0.0487950	1.87819	0.0302
Engineer Error	0.139957	0.0487950	2.86827	0.0021
Feature Interaction	-0.005128	0.0487950	-0.10510	0.5419
Incomplete Knowledge	-0.045752	0.0487950	-0.93763	0.8258
Legacy	-0.052632	0.0487950	-1.07863	0.8596
Missed Scenario or Impact	0.099602	0.0487950	2.04124	0.0206
Other	-0.021898	0.0487950	-0.44877	0.6732
Propagated / Child SR	0.154589	0.0487950	3.16814	0.0008
Reuse	-0.076923	0.0487950	-1.57645	0.9425
Team Coordination	-0.007194	0.0487950	-0.14744	0.5586
Third Party	0.425505	0.0487950	8.72026	0.0000
Overall	0.072848	0.0216159	3.37010	0.0004

Fig. 7. Root Cause. Assessment agreement between appraisers (no standard defined).

6.3 Screen Failure

Screen Failure analysis is presented on the following figures.

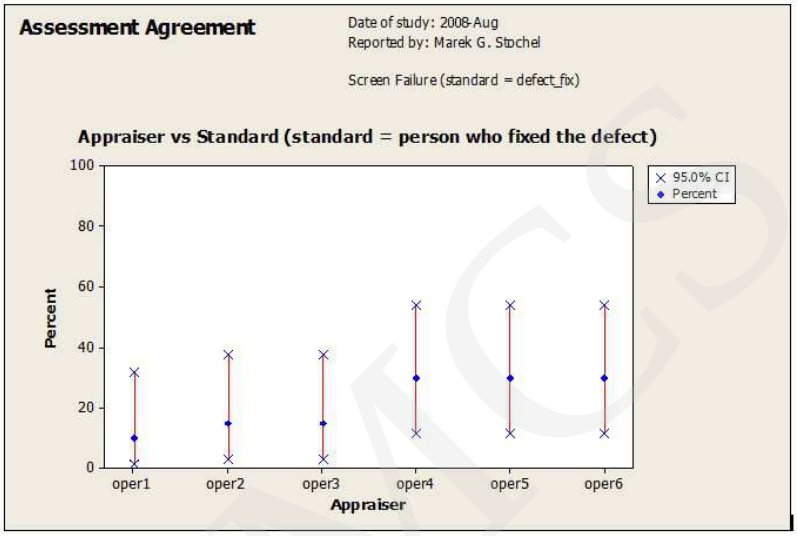

Fig. 8. Screen Failure. Assessment agreement of appraiser vs. standard (the person who fixed the defect).

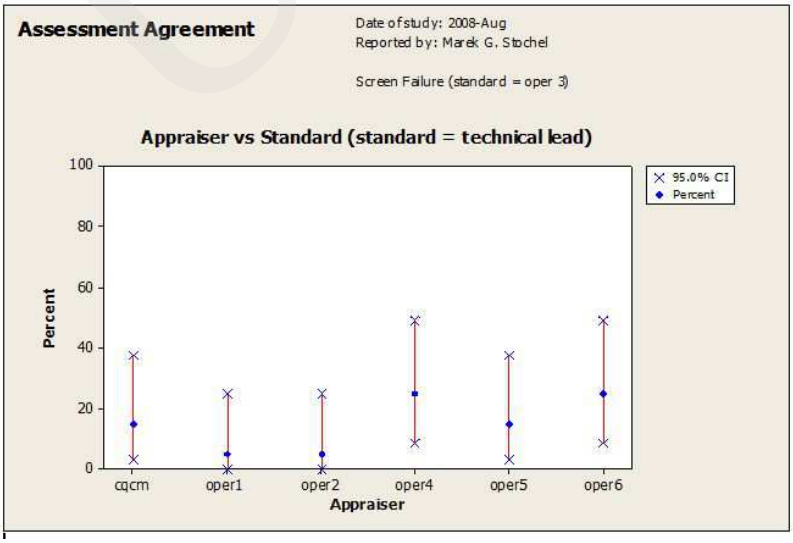

Fig. 9. Screen Failure. Assessment agreement of appraiser vs. standard (the experienced technical lead).

Screen Failure, Agreement Between Appraisers (no standard)

Assessment Agreement

# Inspected	# Matched	Percent	95 % CI
20	0	0.00	(0.00, 13.91)

Matched: All appraisers' assessments agree with each other.

Fleiss' Kappa Statistics

Response	Kappa	SE Kappa	Z	P(vs > 0)
Found Properly	0.320074	0.0487950	6.55957	0.0000
Legacy	-0.076923	0.0487950	-1.57645	0.9425
Other	-0.041026	0.0487950	-0.84078	0.7998
Process Insufficient	0.032099	0.0487950	0.65783	0.2553
Process Not Capable	-0.021898	0.0487950	-0.44877	0.6732
Process Violation	0.089624	0.0487950	1.83674	0.0331
Propagated / Child SR	-0.014493	0.0487950	-0.29701	0.6168
Test Case Error	0.010461	0.0487950	0.21438	0.4151
Test Case Traceability	0.043716	0.0487950	0.89591	0.1852
Test Never Run	-0.109872	0.0487950	-2.25171	0.9878
Test Process Violation	0.032099	0.0487950	0.65783	0.2553
Overall	0.072771	0.0201304	3.61500	0.0002

Fig. 10. Screen Failure. Assessment agreement between appraisers (no standard defined).

7 Conclusions

A standard is understood as a proper assessment of a particular problem (as in the Kappa Analysis). Therefore no single standard can be readily defined as each defect is a unique entity by its nature and could be viewed differently from varying perspectives. If we try to define a standard by the person fixing a defect or by the most skilled technical lead, operators rarely agree with standard by more than 50%. Neither the experienced technical lead nor the person who fixed the defect can be perceived as a consistent "reference level".

Establishing the so called "standard" by the skilled team will be of little value as each defect would be analyzed several times. Typically during the development cycle there is little resource available for this degree of analyses.

As bias against standard cannot be taken into account, the focus in this experiment was on an agreement between engineers (precision of measurements) instead. However, the results show that in any of the RCA categories: *Root Cause, Screen Failure, Problem Type* engineers could not agree with each other, so statistically data is unusable for planning further defect prevention activities. Agreement was unacceptable, as establishing assessment consistency is vital to assure proper focus and successful direction of defect prevention actions.

A twin study was performed by two more teams, which are developing other SW products, however, using the same approach for defect tracking and assessing their root

causes. The initial findings were confirmed. None of the teams in any category reached level kappa>0.7, which indicates the weakest acceptable agreement for an assessment accuracy (however unacceptable for critical decisions, because it introduces high risk).

Similar issues can be observed when people have to decide whether a defect is must-have fix for the customer or not. As a result of this criticality towards the customer, the quality organization should treat any of such defect assessments with great caution.

There are variations to consider within and between the SW engineering community: domain knowledge, experience, exposure to the problem discovered – just to name a few. These factors influence their perspectives and judgments. Similarly, the root causing process through assessing defects and assigning them to predefined categories is by itself a subjective process, adding additional variance. The difference observed was negligible whether root causing is focused on an engineering problem, process problem or test screening effectiveness. The process of collecting the data itself brought the inconsistency of results.

8 Solution proposal

Two approaches were evaluated to address the problem:

1. Implementing teamwork to gain a synergetic effect
2. Implementing an up front engineering analysis based on code assessment.

The first approach was to leverage the benefits from human factor engineering practices, to establish a process of assessment by a common group – achieving synergy with a commonality of understanding and language. All defect prevention actions will be based on the team knowledge, not the variety of the individual views. Equally important, any of the resultant actions will have buy-in from the team and will be supported. This approach can be used to assess process issues and screening effectiveness criteria.

However, if the individual behaviour varies significantly, this could be as well a problem while assessing the behaviour of independent teams. Unfortunately, no reproducibility studies were performed by the teams in the case study presented in the article referenced in Section 2 [**3**]. If we assume that variation of observation and judgment on the individual level could be observed on the team level as well, the problem persists.

To avoid this issue and reach agreement on the defect prevention actions, the focus should be moved from gathering more data by independent entities and then choosing actions to collecting less data by one team (on more critical issues) and have this team engaged in choosing the defect prevention activities. Also important is that the proper assessment (if a proper one could be defined at all) is the consistent cause and effect relation between the observed problems and the chosen defect prevention activities.

The second approach utilizes the mapping of software code in the predefined system areas (unique functionalities), and the mapping of features on the code (if possible, very specific tracking of defects to the code area could be widely useful in testing prioritization and planning [**1**]. This approach is narrowed to a strict engineering point of view (identifying system areas and features which are the main source of defects).

This approach could at least minimize the problems depicted earlier:

1. A form of standardization made upfront (based on the assessment of code areas and a decision which functional areas are impacted)
2. Execution of RCA process in such a way that little or no ambiguity is observed (a set of engineers together assesses the defects):
   a. By sharing the knowledge/experience we gain synergy and commonality of observation and judgment
   b. Bias must not be considered, as we cannot establish an independent standard to which we want to compare
   c. Precision is not an issue (this implies no issue because no measurements are done by different operators). This leads to a consequence: we are unable to question precision, however, we are sure that the actions taken based on such data will address the issue in a more convenient and consistent way (due to the synergy which could be observed).

Both approaches produce meaningful data; however, how they will be applied depends on the nature of expected output.

There is still no 'silver bullet' when taking into account human factors during the assessment. The issue is caused mainly by the fact that the measurement system for subjective data collection, actually the very instrument, which we want to calibrate, is the human brain.

9 Summary

The RCA process should take advantage on differences among engaged engineers, building on creativity and synergy of complementary views. People are different and will differ in terms of their perspective, past experience, and a level of knowledge. All these characteristics constitute a filter through which they derive their conclusions and, in particular, classify defects.

The vital information for further processing should be consistent and valid, as it becomes feedback information which guides the testing process towards revealing more defects before they reach a customer and directly influences testing efficiency and effectiveness. Thus, until we can completely eliminate the human out of defect categorization process, the uncertainty about subjective information will remain. Any approach which brings us closer to understanding the nature of our data and assessment process itself, will result in better 'calibration' of the feedback system used in software engineering.

Our case study shows that defect categorization done by different engineers should be used only with great caution as the data obtained proved to depend largely on the assessor. Nevertheless, two new approaches were proposed (i.e. team assessment and system area/feature categorization) and the first studies on the data they provide show promising improvement. Still further research is required in this direction.

References

[1] Stochel M.G., Sztando R., Testing optimization for mission-critical, complex, distributed systems, Proceedings of 32nd Annual IEEE International Computer Software and Applications Conference, 28 July - 1 August, 2008, IWSC 2008, Turku, Finland (2008): 847.

[2] Cockburn A., Agile Software Development: The Cooperative Game (2nd Edition), Addison-Wesley Professional (2006).

[3] Leszak M., Perry D.E., Stoll D., A case Study in Root Cause Defect Analysis, Proceedings of the 22nd international conference on Software engineering, ICSE 2000, Limerick, Ireland, (2000): 428.

[4] Cialdini R.B., Influence. Science and Practice (4th Edition), Allyn & Bacon (2000).

[5] Motorola University, Six Sigma Black Belt Program, Participant Guide 3, Version 4.0, Motorola (2008).

[6] Chillarege R., et al., Orthogonal Defect Classification – A concept for In-Process Measurements, IEEE Transaction on Software Engineering 18(11) (1992).

[7] Beizer B., Software Testing Techniques, International Thomson Computer Press (1990).

[8] Fagan M.E., Design and Code inspections to reduce errors in program development, IBM Systems Journal 15(3) (1976): 182.

[9] CMMI for Development, Version 1.2. Carnegie Mellon University Software Engineering Institute. (2006).

[10] Motorola University, Six Sigma Black Belt Program, Participant Guide 1, Version 4.0, Motorola (2008).

[11] George M.E., Rowlands D., Price M., Maxey J., The Lean Six Sigma Pocket Toolbook, McGrawHill (2005).