Scientific Research Publishing

Journal of Computer and Communications, 2016, 4, 40-60
http://www.scirp.org/journal/jcc
ISSN Online: 2327-5227
ISSN Print: 2327-5219

A Six Sigma Security Software Quality Management

Vojo Bubevski

Bubevski Consulting™, Brighton, UK
Email: vbubevski@aol.com, vojo.bubevski@landg.com

How to cite this paper: Bubevski, V. (2016) A Six Sigma Security Software Quality Management. *Journal of Computer and Communications*, **4**, 40-60.
http://dx.doi.org/10.4236/jcc.2016.413004

Received: July 15, 2016
Accepted: October 23, 2016
Published: October 26, 2016

Abstract

Today, the demand for security software is Six Sigma quality, *i.e.* practically zero-defects. A practical and stochastic method is proposed for a Six Sigma security software quality management. Monte Carlo Simulation is used in a Six Sigma DMAIC (Define, Measure, Analyze, Improve, Control) approach to security software testing. This elaboration used a published real project's data from the final product testing lasted for 15 weeks, after which the product was delivered. The experiment utilised the first 12 weeks' data to allow the results verification on the actual data from the last three weeks. A hypothetical testing project was applied, supposed to be completed in 15 weeks. The product due-date was Week 16 with zero-defects quality assurance aim. The testing project was analysed at the end of the 12th week with three weeks of testing remaining. Running a Monte Carlo Simulation with data from the first 12 weeks produced results which indicated that the product would not be able to meet its due-date with the desired zero-defects quality. To quantify an improvement, another simulation was run to find when zero-defects would be achieved. Simulation predicted that zero-defects would be achieved in week 35 with 56% probability, and there would be 82 defects from Weeks 16 - 35. Therefore, to meet the quality goals, either more resources should be allocated to the project, or the deadline for the project should be moved to Week 36. The paper concluded that utilising Monte Carlo Simulations in a Six Sigma DMAIC structured framework is better than conventional approaches using static analysis methods. When the simulation results were compared to the actual data, it was found to be accurate within −3.5% to +1.3%. This approach helps to improve software quality and achieve the zero-defects quality assurance goal, while assigning quality confidence levels to scheduled product releases.

Keywords

Security Software, Quality Management, Six Sigma, DMAIC, Monte Carlo Simulation

1. Introduction

Six Sigma methodologies were originally formulated by Motorola in the mid-1980s. Subsequently, Six Sigma evolved into a set of comprehensive and powerful improvement frameworks and tools. Six Sigma refers to having six standard deviations between the mean of the actual performance of the process and the expected performance limits of the process. That translates to a 0.999997 probability (99.9997%) that the process performance is as desired, or to fewer than 3.4 failures per one million opportunities.

Most industries today recognize Six Sigma as a standard means to accomplish process and quality improvements. One of the principal Six Sigma methodologies is Define, Measure, Analyse, Improve, and Control (DMAIC). DMAIC comprises [1] [2]:

1) Define: defining the process, objectives and quality goals;

2) Measure: establishing the metrics and measuring the current process performance;

3) Analyse: analysing the measurement results and collected data to determine the root causes of the process variability and risk;

4) Improve: considering alternatives to eliminate the root causes and determining and applying the improvement solution to upgrade the process; and

5) Control: continuous monitoring and establishing corrective mechanisms to rectify the deviations and control the process performance in the future.

It has been well understood for more than a decade now that the root-cause of most security exposures is in the software itself, and that these vulnerabilities are introduced during the development process. The software development is an inherently uncertain process, thus security defects are part of it. Software organisations in the past were not striving for Six Sigma quality as they had profited from quickly releasing imperfect software and fix the defects later in the field. In particular for security software, this is now changing because software security vulnerabilities can be exploited before the security defects are patched. The "ship, pray and patch" approach is not applicable anymore so the software organisations should aim Six Sigma quality for security software [3].

Nowadays, security is one of the most important quality challenges of software systems and organisations. It is difficult though to keep the systems safe because of the security evolution. In order to adequately manage the security evolution, it should be considered for all artefacts throughout the software development lifecycle. The article published by Felderer *et al.* elaborated on the evolution of security software engineering considering the security requirements, architectures, code, testing, models, risks and monitoring [4].

Falah *et al.* presented an alternative approach to security software testing. In this paper, the focus is on improving the effectiveness of the categorization of threats by using Open 10 Web Application Security Project's (OWASP) that are the most critical web application security risks in generating threat trees in order to cover widely known security attacks [5].

A fuzz approach to security testing was presented by Pietikäinen *et al.* The authors emphasised the challenges, experiences, and practical ways of utilizing fuzzing in soft-

ware development, focussing on software security aspects [6].

Software quality is a multidimensional property of a software product including customer satisfaction factors such as reliability, functionality, usability, performance, capability, installability, serviceability, maintainability and documentation. Software processes are inherently variable and uncertain, thus involving potential risks. A key factor in software quality is *Software Reliability* as it is the quality attribute most exposed to customer observation. In this chapter, the terms "reliability" and "quality" are used interchangeably.

The Orthogonal Security Defect Classification (OSDC) was established and used by *Hunny* to assess and improve the quality of security software [7]. OSDC also provides for applying qualitative analysis to the security software risk management. OSDC is based on the Orthogonal Defect Classification (ODC), which was elaborated by *Chillarege* and implemented by IBM™ (Ref., Chapter 9 in [8]). *Chillarege* applied the Inflection S-shaped Software Reliability Growth Model for relative risk assessment.

Software Reliability is a main subject in *Software Reliability Engineering* (*SRE*) [8]. The software reliability analytic models have been used since the early 1970s [8]-[10]. The need for a simulation approach to software reliability was recognized in 1993 by *Von Mayrhauser et al.* [11]. Subsequently, substantial work on simulation was published [12]-[15].

Applications of Six Sigma in software development have been published since 1985 [16]-[22]. Six Sigma software practitioners usually employ analytic models, but it has been reported that for Six Sigma, simulation models are superior [23]. *Nanda* and *Robinson* published a Six Sigma roadmap for software quality improvements [24]. *Galinac & Car* [25] elaborated an application of Six Sigma in continuous software processes' improvement. In addition, *Macke & Galinac* [26] applied Six Sigma improvements in software development. A six sigma DMAIC software quality improvement was presented by *Redzic & Baik* [27]. Also, *Xiaosong et al.* [28] used Six Sigma DMAIC to model the software engineering process.

Some other work related to software quality risk management was published [14], in which simulation was applied for software reliability assessment. *Gokhale*, *Lyu & Trivedi* [12] [13] developed simulation models for failure behaviour. Also, *Gokhale & Lyu* [29] applied simulation for tailoring the testing and repair strategies.

Monte Carlo simulation is a methodology which iteratively evaluates a deterministic model by applying a distribution of random numbers to account for uncertainty. Simulation allows the use of probability and statistics for analysis [30] [31].

Security software plays a major role in Information Security and Computer Fraud as defects in security software promote insecurity and fraud. So, the security software quality is crucial for improving security and reducing fraud. Considering the losses resulting from insecurity and fraud, nowadays Six Sigma quality of security software (which practically translates to zero-defects security software) is a necessity. Traditional security software quality management of ongoing software projects have two observed deficiencies: 1) structured methods are not utilised to accomplish Six Sigma quality;

and 2) analytic risk models are applied.

A practical and stochastic method is presented in this paper that is Six Sigma security software quality management. This method utilises proven-in-practice methodologies such as Six Sigma Define, Measure, Analyse, Improve and Control (DMAIC), Monte Carlo simulation and Orthogonal Security Defect Classification (OSDC). The DMAIC framework is used to tactically enhance the software process which is inherently uncertainin order to accomplish Six Sigma quality. Monte Carlo simulation is used to: 1) measure software process performance using Six Sigma process capability metrics; 2) measure and evaluate the quality (reliability); and 3) identifies and quantifies the quality risk. OSDC provides for considering the qualitative aspects of quality management, which complements the quantitative analysis.

DMAIC is a recognised structured methodology across industries today for systematic process and quality improvements. Analytic risk models are very much inferior to Monte Carlo simulation, which is a stochastic method. Significant qualitative improvements of quality management are provided by OSDC. Therefore, the perceived deficiencies are eliminated, which significantly improves the security software quality thus increasing information security and reducing the computer fraud risk. The method provides for achieving Six Sigma security software quality thus gaining imperative benefits such as savings and customer satisfaction. The method is compliant with CMMI® (Capability Maturity Model Integration).

Published real project data are used to elaborate the method. It should be noted that the method is presented from practical perspective only providing references for the reader to explore the theory. The method has been successfully used in practice on commercial projects gaining significant benefits [32].

The simulation models in the paper were implemented in Microsoft™ Excel® using the Palisade™ @RISK® add-in.

2. Six Sigma Security Software Quality Management: The Method

This paper is based on author's published work [32]-[35]. The method elaboration follows the DMAIC methodology stage-by-stage. The DMAIC stage reference is given in the topics for understanding.

3. Hypothetical Scenario (DMAIC Define)

The elaboration is based on a real IBM™ software development project using published data (Ref. Dataset ODC4 in [8]). The project is finished so this case is hypothetical. The original defects are classified by using the ODC (Ref. Chapter 9 in [8]). In order to emulate the security software scenario, the original defect classification is remapped to OSDC based on the ODC-OSDC mapping matrix published by Hunny [7]. So in this hypothetical security softwarescenario, the security software defects are available for the entire testing cycle of 15 weeks. The OSDC Defect Types considered are: Security Functionality (SF), Security Logic (SL) and Miscellaneous (Other). Toemulate the scenario of an ongoing process, the data from the first 12 weeks of testing are used. The data

from the last three weeks are used to verify the results of the method.

4. Project Definition (DMAIC Define)

The assumption is that the project is within the final testing stage at the end of Week 12 (TI(12)), which is three weeks from the targeted delivery date of the product. The project is defined as follows:

Project Objective: Complete final test phase by the end of Week 15 (TI(15)) as planned and deliver the system on time, whilst achieving the quality goal. The delivery date is at the beginning of Week 16.

Project Quality Goal: The aim is to ensure that the system is stable and ready for delivery. All detected defects should be fixed and re-tested before the end of testing. Also, the final week of testing (Week 15) should be defect-free for all defect types including total, *i.e.* zero defects to match the Six Sigma quality.

Problem Statement: Assess and mitigate the risk to deliver the system on time, whilst achieving the quality goal.

CTQ: Critical to Quality for the project is the software reliability of the system.

5. Project Metrics (DMAIC Measure)

For the data in details, please refer to the Appendix section. The *Failure Intensity Function* (*FIF*) is used. The Poisson distribution is used to simulate FIF in the model. So, FIF by Defect-Type need to be approximated. The logarithmic and exponential approximations were applied. The R-square values for the logarithmic approximations were the following: 1) SF: $R^2 = 0.9254$; 2) SL: $R^2 = 0.8981$; 3) Other: $R^2 = 0.7385$; and 4) Total: $R^2 = 0.9604$. For the exponential approximation the R-square values were as follows: a) SF: $R^2 = 0.8999$; b) SL: $R^2 = 0.9276$; c) Other: $R^2 = 0.7642$; and d) Total: $R^2 = 0.9665$. Comparing the R-square values, the exponential approximation is more accurate than the logarithmic approximation. Thus, the exponential approximation of the FIF is used applying the Musa's Basic Execution Time Model for simulation.

The aproximation of the transformed FIF by Defect Type (**Figure 1**) is as follows:

$$\mathrm{FIFsf}(k) = 262.33\exp(-0.274k) \tag{1}$$

$$\mathrm{FIFsl}(k) = 179.17\exp(-0.217k) \tag{2}$$

$$\mathrm{FIFother}(k) = 41.138\exp(-0.127k) \tag{3}$$

$$\mathrm{FIFtotal}(k) = \mathrm{FIFsf}(k) + \mathrm{FIFsl}(k) + \mathrm{FIFother}(k) \tag{4}$$

where, FIFsf, FIFsl, FIFother and FIFtotal are the FIF for SF, SL & Other Defect-Type and the Total, and k is the time interval ($k = 1, 2, \cdots, n$).

6. Process Simulation (DMAIC Measure)

In order to analyze the process, the software reliability for the future period of three weeks, *i.e.* from TI(13) to TI(15) inclusive, will be predicted (simulated). The simulation model is the discrete event simulation based on Musa's Basic Execution Time

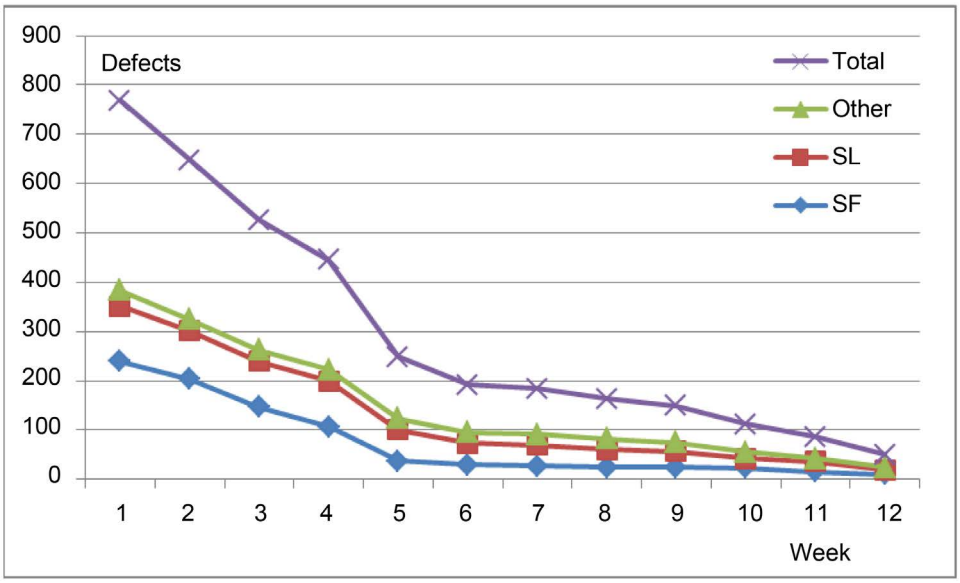

Figure 1. Transformed FIF by defect-type.

Model. This model is used to predict the future course of the FIF.

The Poisson distribution is used in the model in order to account for the variability and uncertainty of reliability. The FIF by Defect-Type are simulated for the three week period TI(13)-TI(15), thus the mean of the Poisson distribution for every defect type for TI(13), TI(14) and TI(15) will be equal to the value of the approximated FIF for TI(13), TI(14) and TI(15) respectively (1 - 3). The Total for TI(13), TI(14) and TI(15) is simply the sum of defects for all three types for TI(13), TI(14) and TI(15) respectively (4).

To define the quality target according to the Quality Goal statement, the FIF will be used. The target is to achieve Six Sigma quality, *i.e.* paractically zero-defects in Week 15 for all Defect-Types including Total, *i.e.* the FIF should be equal to zero for all defect types including total defects in the final week of testing.

In order to define the quality targets in the model, the Six Sigma *Target Value*, *Lower Specified Limit* (*LSL*) and *Upper Specified Limit* (*USL*) will be used: a) Target Value is zero (0) for all defect types including Total; b) USL is three (3) for all defect types including Total; b) LSL should be zero, but for all defect types including Total it will be set to minus three (−3), *i.e.* a small negative number, to prevent an error in the Six Sigma metrics calculations.

The Six Sigma metrics in the model are used to measure the performance of the process. In this model the following Six Sigma metrics are used: a) *Process Capability* (*Cp*); b) *Process Capability Index* (*Cpk*); and c) *Sigma Level*.

To apply the Poisson distribution, the @RISK® Poisson distribution function was used. To calculate Standard Deviation, Minimum Value and Maximum Value, the corresponding @RISK® fuctions for Standard Deviation, Minimum Value and Maximum Value were used. To calculate the Six Sigma process metrics Cp, Cpk and Sigma Level, the @RISK® Cp, Cpk and Sigma Level functions were respectively used. For the Six

Sigma Target Value, USL and LSL, constants were specified. It should be noted that the Six Sigma Cp, Cpk and Sigma Level metrics are calculated from the resulting probability distribution from the simulation.

The Six Sigma process simulation results on Figure 2 show that the Total's distribution in the final week of testing TI(15) totally deviates from the process target specifications (LSL, Target Value and USL are marked on the graph). Also, there is a 0.9 (90%) probability that the Total in TI(15) would be in the range 11 - 24; 0.05 (5%) probability that the Total would be more than 24; and 0.05 (5%) probability that the Total would be less than 11.

Table 1 shows the predicted mean (μ) total number of defects by Defect-Type in the final week of testing TI(15) including the Total. Also, the associated Standard Deviation (σ) and Minimum and Maximum Values obtained from the simulation are shown.

The predicted Total in TI(15) is 17, with Standard Deviation of 4.17 defects. This strongly indicates that the product will not be stable for delivery at the end of Week 15.

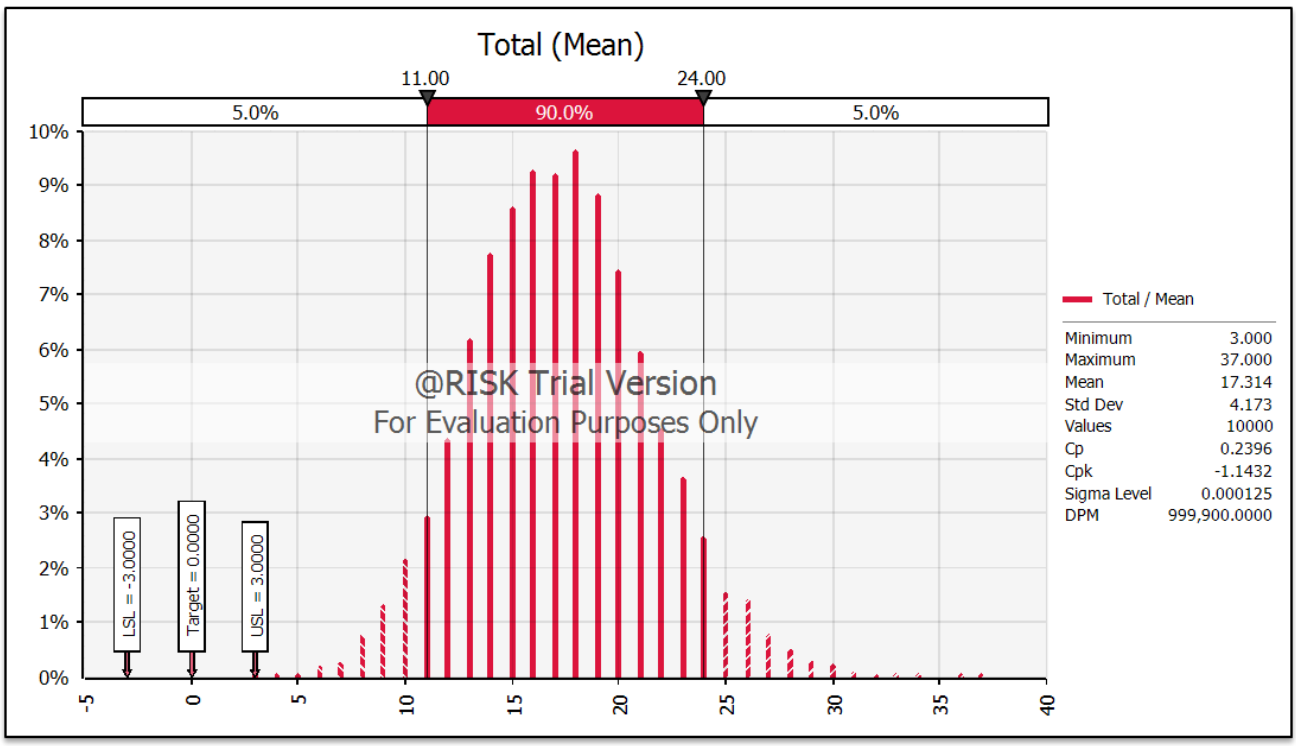

Figure 2. Total defects probability distribution for week 15.

Table 1. Predicted FIF for week 15.

Process	μ	σ	Min value	Max value
SF	4	2.08	0	15
SL	7	2.63	0	21
Other	6	2.48	0	18
Total	17	4.17	4	35

The Six Sigma metrics by Defect-Type for Week 15 is given in Table 2. The Cp metric is extremely low for all defect types including the Total, so the chances that the process will deliver the desired quality are extremely low. Also, the Cpk metric for all defect types incl. total defects is negative, which confirms that the quality target won't be met. Finally, the Sigma Level is almost zero for all defect types including the Total. Thus, there are no chances that the process will perform as expected. For example, the process Sigma Level metrics is 0.50, 0.11 and 0.18 for SF, SL and Other defects respectively. Finally, Sigma Level for total defects is practically zero (*i.e.* 0.0001), which confirms that the process will not perform at all, so it will not deliver the desired Six Sigma quality at the end of Week 15.

7. Sensitivity Analysis (DMAIC Analyse)

The Six Sigma simulation sensitivity analysis can show what factors have the most influence on the process variability and risk. It also can quantify this influence. It is actually used to identify the Critical-To-Qualities (CTQs) of the software process. The presented correlation and regression sensitivity analysis is performed on the predicted (simulated) data distribution for TI(13)-TI(15) only in order to determine the influence of the change of a particular Defect-Type to the change of Total Defects for all Defect Types.

On the regression coefficients graph (Figure 3) it can be seen that the top risk CTQ is the SL Defect-Type with correlation coefficient to the Total of 0.63; the less risky CTQ is the Other Defect-Type with correlation coefficient to the Total of 0.59; and the minimal risk CTQ is the SF Defect-Type with correlation coefficient to the Total of 0.50.

Also, on the regression mapped values graph (Figure 4) the quantitative parameters of the influence of the CTQs if they change by one Standard Deviation can be seen. Thus, if the SL defects increase by one Standard Deviation, the Total will increase by 2.64 defects; if the Other defects increase by one Standard Deviation, the Total will increase by 2.45 defects; and if SF defects increase by one Standard Deviation, the Total will increase by 2.10 defects.

8. Analysis Conclusion and Recommendation (DMAIC Analyse)

The conclusions from this Six Sigma analysis: a) The testing process will not perform at all as shown by the considered Six Sigma metrics. Therefore, the system would not be ready for delivery as the Six Sigma quality goal will not be met at the beginning of Week

Table 2. Process six sigma metrics for week 15.

Process	Cp	Cpk	Sigma level
SF	0.4805	−0.2056	0.5024
SL	0.3801	−0.4962	0.1064
Other	0.4034	−0.4199	0.1757
Total	0.2396	−1.1432	0.0001

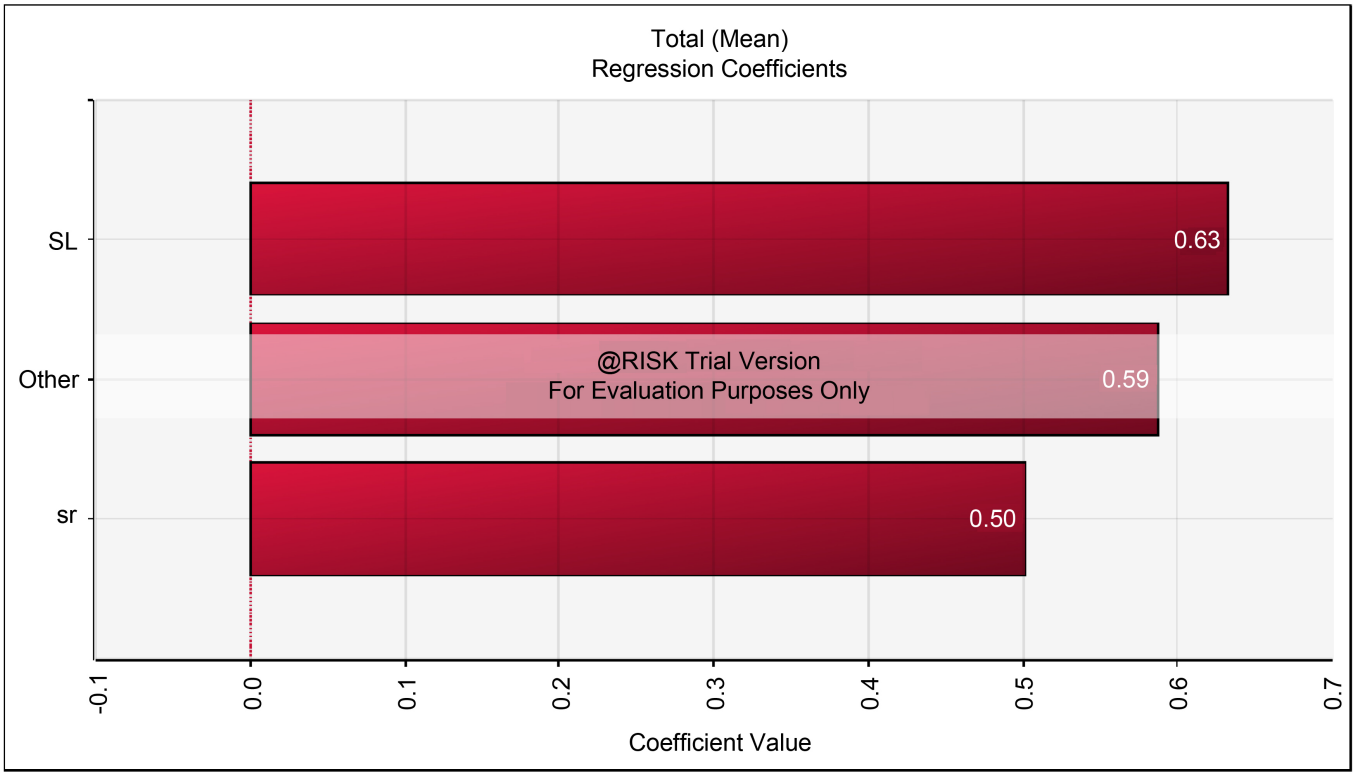

Figure 3. Regression coefficients of total defects by defect-type.

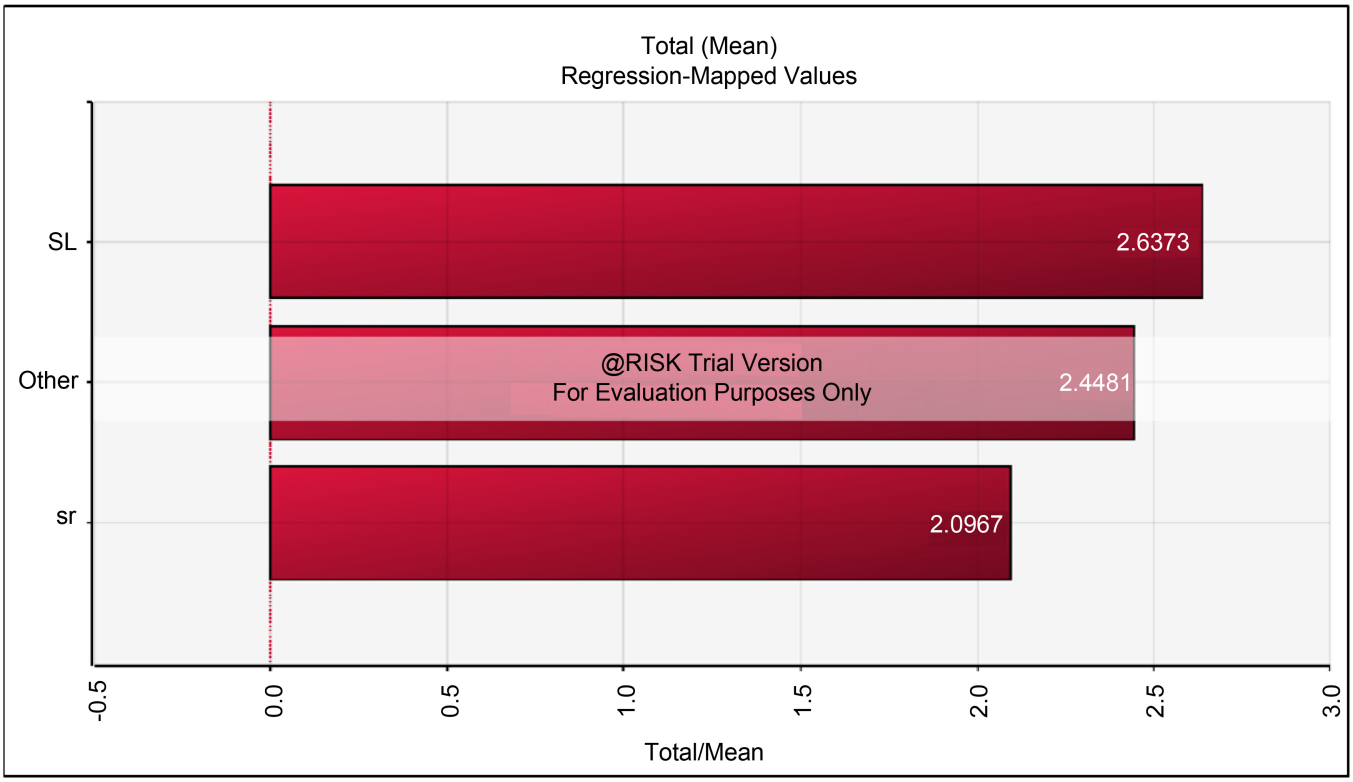

Figure 4. Regression mapped values of total defects by defect-type.

16 if the project maintains the current situation; and b) The CTQ to deliver the system is the software reliability, *i.e.* the predicted Total in TI(15) is 17 defects, versus the target value of sero defects.

Analysis Recommendation: In order to deliver the system on time and achieve the quality goal, immediately undertake an improvement project to improve the process and enhance the software reliability to accomplish the Six Sigma quality, which is the CTQ.

9. Improvement Simulation (DMAIC Improve)

In order to undertake the recommended action, the software process needs to be analysed again. The purpose of this Six Sigma simulation is to quantitatively determine the solution for improvement, which is an example of Six Sigma Design of Experiments (DOE). For this purpose, all the escaped defects will be predicted, *i.e.* the defects that are believed to be in the system but they are not captured. Therefore, the software reliability for the future period will be simulated to predict when the reliability goal will be achieved, *i.e.* in which time interval (week) there will be zero defects in total.

It was identified that the Six Sigma quality target could be met in Week 35. Hence again, the discrete event simulation is applied based on Musa's Basic Execution Time Model to predict the future course of the FIF. All the parameters of this simulation were exactly the same as the parameters of the previous simulation. FIF by Defect-Type was simulated for the future period of 23 weeks, *i.e.* from Week 13 to Week 35.

As Figure 5 shows, the Total's distribution in Week 35 of testing fits in the process

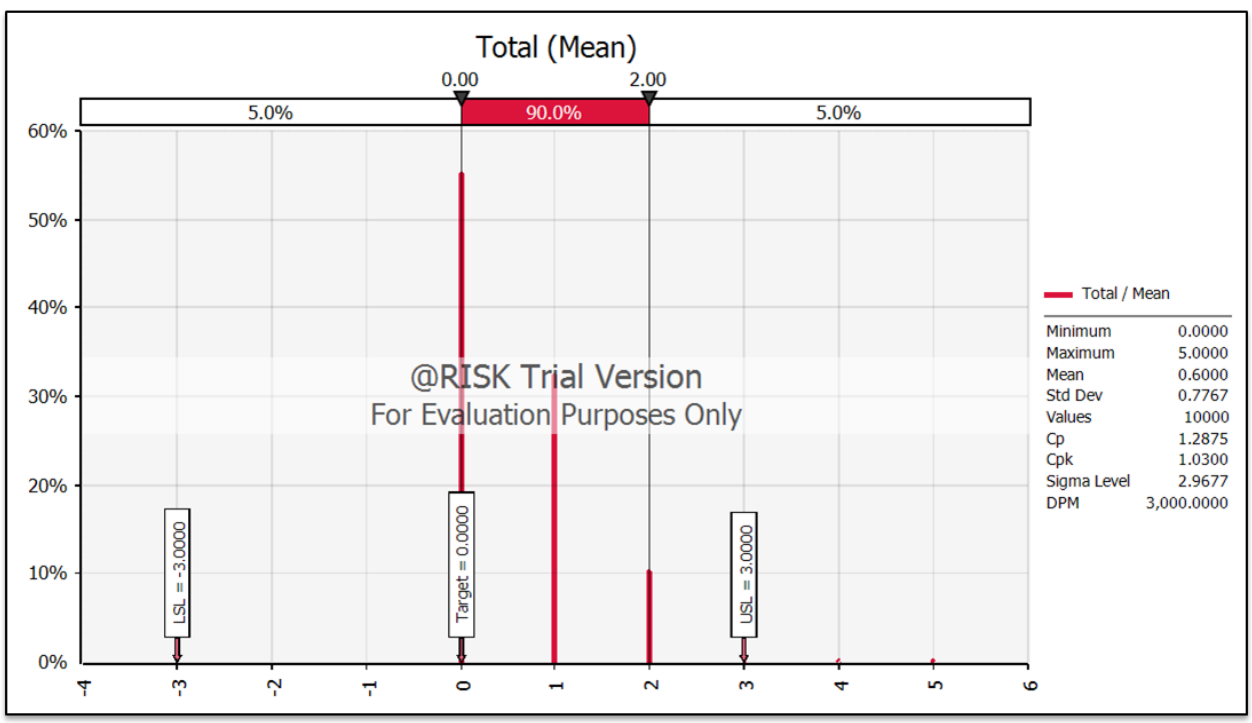

Figure 5. Total defects probability distribution for week 35.

target specifications (LSL, Target Value and USL are marked on the graph). Also, there is a 0.90 (90%) probability that the Total in TI(35) would be in the specified target range 0 - 2 defects; and 0.05 (5%) probability that the Total would be more than two defects. The probability that there will be zero defects in total is approximately 0.56 (56%).

According to this prediction, the process can achieve the Six Sigma quality goal (*i.e.* zero defects in total) in Week 35 if the project maintains the current situation. The confidence levels of the predicted Total for Week 35 are around: 1) 56%, for zero defects; 2) 32%, for one defect; and 3) 10% for two defects. Thus, this indicates that the system will be stable for delivery at the end of the prolonged testing achiving the Six Sigma quality goal.

The simulation results for Week 35 are presented in Table 3. The predicted number of defects for all types, including the Total, is within the specified target range. The Standard Deviation for all types including the Total is low, *i.e.* nearly zero defects.

The process Six Sigma metrics at the end of Week 35 are given in Table 4. The Cp metric is greater than one for all defect types including the Total, so the chances that the process will deliver the desired quality are extremely good. Also, the Cpk metric for all defect types incl. total defects is greater than one, which confirms that the quality target will be met. Finally, the Sigma Level is very high for all defect types including the Total. Thus, there are very good chances that the process will perform as expected. For example, the process Sigma Level metrics is 16.91, 7.28 and 3.17 for SF, SL and Other defects respectively. Finally, Sigma Level for total defects is almost three (*i.e.* 2.97), which confirms that the process will certainly perform well, so it will deliver the desired Six Sigma quality at the end of Week 35. Compared with the Six Sigma metrics of the preceding Six Sigma simulation in Table 2, the metrics in Table 4 are exceptionally better.

All three Six Sigma metrics suggest that there are very realistic chances that the process will perform and deliver the desired Six Sigma quality at the end of Week 35. This is

Table 3. Predicted FIF for week 35.

Process	μ	σ	Min Value	Max Value
SF	0	0.13	0	2
SL	0	0.31	0	2
Other	0	0.70	0	5
Total	0	0.78	0	5

Table 4. Process six sigma metrics for week 35.

Process	Cp	Cpk	Sigma Level
SF	7.6031	7.5590	16.9116
SL	3.2710	3.1673	7.2756
Other	1.4273	1.1953	3.1747
Total	1.2875	1.0300	2.9677

also strongly supported by the very low Standard Deviation, that is 0.13, 0.31, 0.70 and 0.78 for SF, SL, Other and Total defects respectively in Week 35, which is practically zero defects.

10. Improvement Recommendation (DMAIC Improve)

The following defines and quantifies the solution for the improvement. The predicted total numbers of defects by Defect-Type including the Total for the future periods are shown in Table 5.

The predicted defects for Week 13 - 15 are expected to be detected and removed by the current project until the end of Week 15. The predicted defects expected to be found in the system from Week 16 to Week 35 are unaccounted for. These defects need to be detected and removed until the end of Week 15 in order to achieve the quality goal.

Therefore, the process improvement recommendation is: Immediately undertake an improvement project to deliver the system quality improvements as required to achieve the quality goals. It is very important to assign a Surgical Team to accomplish the improvement [36]. The objectives of this project are:

1) Reanalyse the unstable defects applying *Casual Analysis and Resolution* (*CAR*);

2) Consider and prioritise defects by type as identifid in the sensitivity analysis (Ref. Sec. Sensitivity Analysis) to complement the quantitative analysis with qualitative analysis provided by OSDC;

3) Determine the quality improvement action plan, establishing an additional tactical test plan;

4) Execute the tactical test plan to additionally test the system and detect and repair the escaped defects, *i.e.* the defects that is believed are in the system but have not been detected. According to the simulation above, there are 82 predicted escaped defects in total (TI(16)-TI(35), Table 6);

Table 5. Predicted defects per defect-type for future periods.

Time Period	SF	SL	Other	Total
TI(13)-TI(15)	17	27	21	65
TI(16)-TI(35)	11	28	43	82
TI(13)-TI(35)	28	55	64	147

Table 6. Results comparison by week.

	Week 13		Week 14		Week 15	
	(Actual) Predicted	Error %	(Actual) Predicted	Error %	(Actual) Predicted	Error %
SF	(6) 7	16.67	(7) 6	−14.29	(4) 4	0
SL	(14) 11	−21.43	(7) 9	28.57	(15) 7	−53.33
Other	(8) 8	0	(3) 7	133.33	(1) 6	500
Total	(28) 26	−7.14	(17) 22	29.41	(20) 17	−15

5) The additional testing, detection and correction of the escaped defects should be completed by the end of Week 15 to achieve the quality goal.

11. Improvement Definition (DMAIC Improve)

It should be stressed that the process improvement is a new testing project, which is totally independent of the current testing in progress. For project planning purposes, it is needed to determine the desired performance of the improvement testing process during the future three weeks. There are only three weeks available to accomplish the improvement, as the quality goal needs to be met at the end of testing (*i.e.* at the ond of Week 15).

Within the last stage of DMAIC it is required to monitor the process performance and variances, and implement corrective measures if deviations from the desired performance are found in the future. For this purpose, keeping one week as the time interval for observation is not good because it provides for only two future check points. Thus, the time interval for observation will be reduced to one day, which provides for five check points per week.

The three weeks available for the project is equivalent to 15 working days. Thus, the proposed schedule for the testing improvement project during the next 15 Days is: 1) one day to start the project and appoint the staff; 2) three days to complete the required analysis and test plans; and 3) 11 days of testing where the escaped defects will be detected and fixed.

The predicted distribution of the escaped defects by Defect-Type including the Total, which need to be detected and fixed during the testing period of 11 days, *i.e.* TI(1)-TI(11), is: 1) SF: 11 Defects; 2) SL: 28 Defects; 3) Other: 43 Defects; and 4) Total: 82 Defects.

12. Alternative (DMAIC Improve)

Alternatively, if the project maintains the status quo situation, the system will be ready for delivery in Week 35. Thus, testing needs to continue until Week 35 inclusive, *i.e.* 23 weeks more than initially planned. The confidence level is 56% that the Six Sigma quality requirements will be met.

13. Simulation for Monitoring (DMAIC Control)

It should be emphasized that in order to manage the quality risk, it is imperative to establish continuous monitoring in order to discover any variances in the process performance, and determine and implement the appropriate corrective actions to eliminate the deviations. This will ultimately mitigate the risk and allow for the delivery of the product on time and the achievement of the quality goals.

In order to deliver the product on time and meet the quality goals, the control phase should be applied to both the current and the improvement testing process. It is recommended to create two additional Six Sigma simulation models and to apply them regularly on a dayly basis to both processes until the end of the projects. The Six Sigma

simulation models for monitoring of the improvement testing process will be very similar to the models presented above.

Considering the fact that we have no original failure (defect) data for the purpose of monitoring, it will be speculative to craft the data to demonstrate the DMAIC Control phase.

14. Verification of Results

The experimental results, *i.e.* the predictions, are compared with the actual available data for verification. It should be underlined that there are no data available from System's Operation. Thus, it is impossible to verify the predictions for improvments and predictions for control.

Three comparisons are performed as presented below: 1) Comparison by Week; 2) Partial Data Comparison; and 3) Overall Data Comparison.

Comparison by Week: The results are verified by comparing the predicted total number of defects per Defect-Type by week including the Total, versus the corresponding actual defects for period TI(13)-TI(15). This comparison is presented in Table 6.

The SF defects and the Total are reasonably predicted. The SL defects are underestimated for two weeks and overestimated for one week with moderate errors. The Other defects are precisely predicted for one week and badly overestimated for two weeks. Overall, these prediction results are tolerable.

Partial Data Comparison: The results are verified by comparing the predicted total number of defects by Defect-Type including the Total for the three weeks period TI(13)-TI(15), versus the corresponding actual defects. The software reliability MTTF is also compared (Table 7).

Here, the SF defects and the Total are exactly predicted. The SL defects are underestimated with a moderate error. The Other defects are substantially overestimated. Overall, these prediction results are acceptable.

Overall Data Comparison: The results are verified by comparing the actual and predicted total number of defects by Defect-Type including the Total for the entire period TI(1)-TI(15), with the corresponding actual defects; The MTTF is also compared. The period of observation is 15 weeks. This comparison is presented in Table 8.

Again, the SF defects and the Total are accurately predicted. The SL defects are underestimated with a minimal error. The Other defects are slightly overestimated. Thus,

Table 7. Partial data comparison.

Process	Defects			MTTF (Weeks)		
	Actual	Pred.	Error %	Actual	Pred.	Error %
SF	17	17	0	0.8824	0.8824	0
SL	36	27	−25	0.4167	0.5556	33.3333
Other	12	21	75	1.2500	0.7143	−42.8571
Total	65	65	0	0.2308	0.2308	0

Table 8. Overall data comparison.

Process	Defects			MTTF (Weeks)		
	Actual	Pred.	Error %	Actual	Pred.	Error %
SF	907	907	0	0.0165	0.0165	0
SL	712	703	−1.2640	0.0211	0.0213	1.2802
Other	251	260	3.5857	0.0598	0.0577	−3.4615
Total	1870	1870	0	0.0080	0.0080	0

these prediction results are very good.

Considering the calculated errors in Tables 6-8, the experimental results are satisfactorily verified. It should be emphasized that the DMAIC-Simulation analysis is more reliable compared to the conventional models. This is because the variability and uncertainty in the software quality process are catered for by applying probability tools. This substantially increases the confidence in the DMAIC-simulation decision support, which is very important for the project.

15. Conclusions

The quality of security software is one of the crucial contributing factors to Information Security and Computer Fraud. So, efficient and comprehensive security software quality management is a necessity to improve the quality, enhance information security and reduce the risk of computer fraud. Considering the importance of Information Security and Computer Fraud today, the demand for security software is Six Sigma quality, *i.e.* practically zero-defects security software.

The conventional security software quality management of ongoing projects has two major weaknesses: 1) analytic risk models are used; and 2) structured methodologies for process and quality improvements are not systematically applied. The proposed practical method applies Six Sigma DMAIC, Monte Carlo simulation and OSDC methodologies. Simulation is superior to analytic risk models and DMAIC is a proven and recognized methodology for systematic process and quality improvements. OSDC provides for qualitative analysis offering qualitative improvements. This synergetic method eliminates the observed limitations of the conventional approach.

The method fully follows the DMAIC framework including the five phases: define, control, analyse, improve and control. The elaboration of the method is outlined below.

1) DMAIC Define:

a) The hypothetical security software scenario was expalined highliting that:

A. The published real project was finished;

B. The testing data were claisified by using ODC;

C. To emulate the scenario of security software, the ODC classification was remapped to OSDC by using ODC-OSDC mapping matrix;

D. The data were available for the entire testing cicle of 15 weeks; and

E. The OSDC defect types used were SF, SL and Other.

b) The project was defined from Six Sigma DMAIC perspective including:

A. The assunption that the testing stage was at the end of Week 15
B. The objective to complete the testing in Week 15 and deliver the product in Week 16 achieving the quality target;
C. All defects should be fixed by the end of testing;
D. The quality goal is that the final week of testing (*i.e.* Week 15) should be defect-free;
E. The problem statement is to assess and mitigate the risk to deliver the product on time achieving the quality goal; and
F. The Six Sigma CTQ is the software reliability.

2) DMAIC Measure:

a) The project metrics was elaborated focusing on the testing data:

A. The original FIF was transformed in order to be used;
B. The transformed FIF was approximated;
C. The exponential approximation was sellected based on the R-square values;
D. Poisson distribution was used to simulate FIF; and
E. The Poisson distribution parameters were determined for the simulation.

b) Monte Carlo Simulation was run with data from the first 12 weeks;

c) Simulation results strongly indicated that the product would not be able to meet its due-date with the desired zero-defects quality.

3) DMAIC Analyse: Simulation sensitivity analysis was performed to identify and quantify the influence of the individual defect types, on the total defects:

a) The Regresstion Coefficients determined the coefficients of: 0.63 for the top risk SL defects; 0.59 for the less risky Other defects; and 0.50 for the minimal risk SF defect type.

b) Regression Mapped Values quantified the influence to the Total defects in terms of standard deviation if individual defect types change for one standard deviation. So the Total will increase as follows: 2.64 defects for SL; 2.45 defects for Other; and 2.10 for the SF defect type.

4) DIMAIC Improve:

a) Monte Carlo Simulation was run with data from the first 12 weeks to predict when the zero-defects would be achieved;

b) Simulation results indicated that the product would achieve the desired zero-defects quality in week 35 with confidence level of 56%;

5) DIMAIC Improve: Improvement recommendation was determined as follows:

a) Immediately undertake an improvement project to deliver the system with zero-defects on time;

b) Assign a Surgical Team to accomplish the improvement;

c) The objectives are:

A. Reanalyse the unstable defects applying *Casual Analysis and Resolution* (*CAR*);
B. Consider and prioritise defects by type as identifid in the sensitivity analysis (#3 above) to complement the quantitative analysis with qualitative analysis provided by OSDC;
C. Determine the quality improvement action plan, establishing an additional tactical

test plan;

D. Execute the tactical test plan to additionally test the system and detect and repair the escaped defects, *i.e.* there are 82 predicted escaped defects in total (TI(16)-TI(35);
E. The additional testing, detection and correction of the escaped defects should be completed by the end of Week 15 to deliver the product on time and achieve the quality goal.

6) DIMAIC Improve: Improvement Definition:
a) There are only three weeks available to accomplish the improvement, *i.e.* 15 working days;
b) Tthe proposed project schedule is:
A. One day to start the project and appoint the staff;
B. Three days to complete the required analysis and test plans; and
C. 11 days of testing where the escaped defects will be detected and fixed.
c) The predicted escaped defects which need to be detected and fixed was:
A. SF: 11 Defects;
B. SL: 28 Defects;
C. Other: 43 Defects; and
D. Total: 82 Defects.
7) DIMAIC Improve: Alternatively:
a) If the project continues as is, the product would achieve zero-defects in Week 35;
b) Testing needs to continue until Week 35 inclusive, *i.e.* 23 weeks more than initially planned;
c) Product would be delivered in Week 36;
d) Probabilty could be 56% that the Six Sigma quality requirements would be met.
8) DIMAIC Control:
a) It is vital to continuously monitor the 11 days of testing on daily bases;
b) Two additional Six Sigma simulation models need to be applied for monitoring to provide for measuring and analysis of the testing process;
c) As there are no original defects data for the purpose of monitoring, the control stage is not demonstrated.

The method results were satisfactorily verified. Comparing the simulation results with the actual data, the results were found to be accurate within −3.5% to +1.3%.

The method is compatible with CMMI® and can substantially help software projects to deliver the product on time and achieve the Six Sigma quality goals. It tactically uses the synergy of the three applied methodologies, *i.e.* Six Sigma DMAIC, Monte Carlo Simulation and OSDC, which provides for strong performance-driven software process improvements and achieves important benefits including savings, quality and customer satisfaction.

In comparison with the conventional methods, the stochastic approach is more reliable and comprehensive as the inherent variability and uncertainty are accounted for, allowing for probability analysis of the risk. Therefore, the confidence in the method's decision support is substantial, which is of mission-critical importance for software

projects.

Acknowledgements

I would like to acknowledge Lyu, Michael R. [8] for publishing the real software projects data. I have used these data to experiment with the new method and additionally prove the concept on external commercial projects. This substantially increased the confidence in the new approach and allowed me to publish my work. Very special and warm thanks to my daughter, Ivana Bubevska, for reviewing the manuscript and suggesting very relevant improvements. She has also substantially helped with the editing and formatting of the text. Her contribution has been essential for the successful publication of this work.

References

[1] Siviy, J.M., Penn, L.M. and Stoddard, R.W. (2007) CMMI® and Six Sigma: Partners in Process Improvement (SEI Series in Software Engineering). Addison-Wesley Professional, Boston.

[2] Borror, C.M. (2009) The Certified Quality Engineer Handbook. 3rd Edition, ASQ Quality Press, Milwaukee, 321-332.

[3] Wysopal, C. (2008) Building Security into Your Software-Development Lifecycle. *SC Magazine*, USA, Article 104705. http://www.scmagazine.com/building-security-into-your-software-development-lifecycle/article/104705/

[4] Felderer, M., *et al.* (2014) Evolution of Security Engineering Artifacts: A State of the Art Survey. *International Journal of Secure Software Engineering*, **5**, 48-98. http://dx.doi.org/10.4018/ijsse.2014100103

[5] Falah, B., Akour, M. and Oukemeni, S. (2015) An Alternative Threat Model-Based Approach for Security Testing. *International Journal of Secure Software Engineering*, **6**, 50-64. http://dx.doi.org/10.4018/IJSSE.2015070103

[6] Pietikäinen, P., Kettunen, A. and Röning, J. (2016) Steps towards Fuzz Testing in Agile Test Automation. *International Journal of Secure Software Engineering*, **7**, 38-52. http://dx.doi.org/10.4018/IJSSE.2016010103

[7] Hunny, U. (2012) Orthogonal Security Defect Classification for Secure Software Development. PhD Thesis, Queen's University, Kingston.

[8] Lyu, M.R. (1996) Handbook of Software Reliability Engineering. IEEE Computer Society Press, Washington DC.

[9] Kan, S.H. (2002) Metrics and Models in Software Quality Engineering. Addison-Wesley Professional, Boston.

[10] Xie, M. (1991) Software Reliability Modelling. World Scientific, Singapore. http://dx.doi.org/10.1142/1390

[11] Von Mayrhauser, A., Malaiya, Y.K., Srimani, P.K. and Keables, J. (1993) On the Need for Simulation for Better Characterization of Software Reliability. *Proceedings of 4th International Symposium on Software Reliability Engineering*, Denver, 3-6 November 1993, 264-273. http://dx.doi.org/10.1109/issre.1993.624296

[12] Gokhale, S.S., Lyu, M.R. and Trivedi, K.S. (1997) Reliability Simulation of Fault-Tolerant Software and Systems. *Proceedings of Pacific Rim International Symposium on Fault-Tol-*

erant Systems, Taipei, 15-16 December 1997, pp. 167-173. http://dx.doi.org/10.1109/prfts.1997.640143

[13] Gokhale, S.S., Lyu, M.R. and Trivedi, K.S. (1998) Reliability Simulation of Component-Based Software Systems. *Proceedings of Ninth International Symposium on Software Reliability Engineering*.

[14] Tausworthe, R.C. and Lyu, M.R. (1996) Chap. 16. Software Reliability Simulation. In: Lyu, M.R., Ed., *Handbook of Software Reliability Engineering*, IEEE Computer Society Press and McGraw-Hill Book Company, 661-698.

[15] Lakey, P.B. (2002) Software Reliability Prediction is not a Science… Yet. Chillarege Press, Denver.

[16] Tayntor, C.B. (2002) Six Sigma Software Development. Auerbach, Boca Raton.

[17] Mandl, R. (1985) Orthogonal Latin Squares: An Application of Experiment Design to Compiler Testing. *Communications of the ACM*, **28**, 1054-1058. http://dx.doi.org/10.1145/4372.4375

[18] Tatsumi, K. (1987) Test Case Design Support System. *Proceedings of International Conference on Quality Control*, Tokyo, 1987, 615-620.

[19] Brownlie, R., Prowse, J. and Phadke, M.S. (1992) Robust Testing of AT&T PMX/StarMAIL Using OATS. *AT&T Technical Journal*, **71**, 41-47. http://dx.doi.org/10.1002/j.1538-7305.1992.tb00164.x

[20] Bernstein, L. and Yuhas, C.M. (1993) Testing Network Management Software. *Journal of Network and Systems Management*, **1**, 5-15. http://dx.doi.org/10.1007/BF01026825

[21] Murugappan, M. and Keeni, G. (2003) Blending CMM and Six Sigma to Meet Business Goals. *IEEE Software*, **20**, 42-48. http://dx.doi.org/10.1109/MS.2003.1184165

[22] Zhao, X.S., He, Z., Z.M., Wang, J. and Yu, D.N. (2008) Process Integration of Six Sigma and CMMI. *Proceedings of* 6*th International Conference on Industrial Informatics* (*INDIN*), Daejeon, 13-16 July 2008, 1650-1653.

[23] Ferrin, D.M., Miller, M.J. and Muthler, D. (2002) Six Sigma and Simulation, So What's the Correlation? *Proceedings of the* 2002 *Winter Simulation Conference*, San Diego, 8-11 December 2002, 1439-1443. http://dx.doi.org/10.1109/wsc.2002.1166415

[24] Nanda, V. and Robinson, J.A. (2011) Six Sigma Software Quality Improvement. McGraw-Hill Professional, New York.

[25] Galinac, T. and Car, Z. (2007) Software Verification Improvement Proposal Using Six Sigma. In: Münch, J. and Abrahamsson, P., Eds., *Product-Focused Software Process Improvement*, Springer, Berlin, 51-64. http://dx.doi.org/10.1007/978-3-540-73460-4_8

[26] Macke, D. and Galinac, T. (2008) Optimized Software Process for Fault Handling in Global Software Development. In: Wang, Q., Pfahl, D. and Raffo, D.M., Eds., *Making Globally Distributed Software Development a Success Story*, Springer, Berlin, 395-406. http://dx.doi.org/10.1007/978-3-540-79588-9_34

[27] Redzic, C. and Baik, J. (2006) Six Sigma Approach in Software Quality Improvement. *Proceedings of* 4*th International Conference on Software Engineering Research, Management and Applications* (*SERA*), Seattle, 9-11 August 2006, 396-406. http://dx.doi.org/10.1109/sera.2006.61

[28] Zhao, X., He, Z., Gui, F. and Zhang, S. (2008) Research on the Application of Six Sigma in Software Process Improvement. *Proceedings of* 4*th International Conference on Intelligent Information Hiding and Multimedia Signal Processing* (*IIH-MSP*), Harbin, 15-17 August 2008, 937-940. http://dx.doi.org/10.1109/iih-msp.2008.63

[29] Gokhale, S.S. and Lyu, M.R. (2005) A Simulation Approach to Structure-Based Software Reliability Analysis. *IEEE Transactions on Software Engineering*, **31**, 643-656. http://dx.doi.org/10.1109/TSE.2005.86

[30] Bratley, P., Fox, B.L. and Schrage, L.E. (1983) A Guide to Simulation. Springer-Verlag, New York. http://dx.doi.org/10.1007/978-1-4684-0167-7

[31] Rubinstein, R.Y. and Kroese, D.P. (2008) Simulation and the Monte Carlo Method. John Wiley & Sons, Hoboken.

[32] Bubevski, V. (2013) A Novel Approach to Software Quality Risk Management. *Journal of Software*: *Testing*, *Verification and Reliability*, **24**, 124-154. http://dx.doi.org/10.1002/stvr.1488

[33] Bubevski, V. (2013) A Stochastic Approach to Security Software Quality Management. 2013 *European Modelling & Simulation Symposium*, Athens, September 2013.

[34] Bubevski, V. (2010) An Application of Six Sigma and Simulation in Software Testing Risk Assessment. 2010 3*rd International Conference on Software Testing*, *Verification and Validation* (*ICST*), Paris, 6-10 April 2010, 295-302. http://dx.doi.org/10.1109/ICST.2010.23

[35] Bubevski, V. (2009) A Simulation Approach to Six Sigma in Software Development. *Proceedings of the* 2009 *Summer Computer Simulation Conference*, Istanbul, 13-16 July 2009, 125-132.

[36] Brooks Jr., F.P. (1995) The Mythical Man-Month (Essays on Software Engineering, Anniversary Edition). Addison-Wesley, Boston.

[37] Conover, W.J. and Iman, R.L. (1981) Rank Transformations as a Bridge Parametric and Nonparametric Statistics. *The American Statistician*, **35**, 124-129.

Appendix

The elaboration is based on a real IBM™ software development project using published data (Ref. Dataset ODC4 in [8]). The project is finished so this case is hypothetical. The original defects are classified by using the ODC (Ref. Chapter 9 in [8]). In order to emulate the security software scenario, the original defect classification is remapped to OSDC based on the ODC-OSDC mapping matrix [7]. So in this hypothetical security software scenario, the security software defects are available for the entire testing cycle of 15 weeks.

Therefore, the pretended security software defects found in testing are given in **Table A1**. Considered are the OSDC Defect Types of Security Functionality (SF), Security Logic (SL) and Miscellaneous (Other). The time interval of observation is one week.

In order to demonstrate the method, it is assumed that the project is at the end of the week 12. So, the data from week 1 - 12 is used for analysis, shown in Black in **Table A1**. The data from the final three weeks of testing (week 13 - 15) shown in Blue, in **Table A1**, will be used to verify the method's results. The time interval of analysis start with week one and ends with week 12, that is the start Time Interval TI(1), etc., and the end Time Interval TI(12).

The *Failure Intensity Function* (*FIF*) cannot be derived from raw data. So we apply Rank Transformation [37] to get a smooth FIF, which will be approximated for the analysis. The entire time period of our observation is 15 weeks based on selected data available from the first 12 weeks.

Table A1. Security software defects count data.

Week	SF	SL	Other
1	23	93	23
2	25	33	23
3	240	43	21
4	37	20	7
5	147	98	23
6	203	36	22
7	27	64	18
8	30	112	23
9	107	43	13
10	24	93	23
11	16	20	32
12	11	8	6
13	4	15	1
14	7	7	3
15	6	14	8