Estimating Cost and Defect Removal Effectiveness in SDLC Testing activities

LJUBOMIR LAZIĆ, IVAN ĐOKIĆ
State University of Novi Pazar
Vuka Karadžića bb
36 300 Novi Pazar, SERBIA
llazic@np.ac.rs, idjokic@np.ac.rs

STEVAN MILINKOVIĆ
The School of Computing
Union University
Belgrade, Serbia
smilinkovic@raf.edu.rs

Abstract—**There are a variety of models, methods and tools to help organizations manage defects found in the development of software. Defect tracking and processing must be integrated in the project life cycle and the software testing process as we did in our Optimal Quantitatively Managing Testing (OptimalSQM) process. We describe phased defect injection and removal cost model for projects where test activities are performed sequentially. We can quantify process performance in terms of the "defect containment matrix" of each process stage for use in process improvement and estimating rework costs using Taguchi's Design of Experiments method and SRM (Surface Response Method) to find parametric equation for cost of quality (CoQ) related to DRE (Defect Removal Efficacy).**

Key words - software testing; test efficacy; defect cost; DOE

I. INTRODUCTION

Testing is an important method for quality control. It is also an important process that needs to be managed quantitatively for high maturity organizations. However, quantitative management model of testing is complex because it is constrained not only by the size of product, but also by the quality of implementation activities, such as design and coding. The more defects, the more effort is needed to fix and verify them. How to estimate the effort and the defects related data, establish performance baseline and quantitative management model of testing process is challenge. In fact, many software projects delay due to the slippage of the testing activities.

Software testing is a core activity in quality assurance and control. To improve the test process, we can use best practice models which describe in detail what to do in organizational test processes. The improvement activities using best practice models are performed as follows: checking the current status of test processes, suggesting and planning new actions, and implementing the actions. However, it is difficult to apply all of these actions to the organization due to the limitation of resources. In this paper, we suggest a strategy for optimizing test process action plans. The background of this research is TMM (Testing Maturity Model), which is the most representative test process model. By applying design of experiments to the TMM assessment procedure, we can accept the actions selectively by statistical significance and find the best solution.

In this paper, we suggest statistically verifying the effectiveness of improvement actions which are suggested through a test process assessment procedure applying Taguchi's Design of Experiments method and SRM (Surface Response Method) to find parametric equation for cost of quality (CoQ) related to DRE (Defect Removal Efficacy).

If we know the statistical significance of each action, we can prioritize the actions by their significance and select influential actions more effectively. This strategy maximizes the positive effects of action plans.

II. RELATED STUDIES

In this paper, we present a strategy to utilize statistical methodology in TMM-based test process improvement. We cannot find any paper which deals with the same subject. Thus, we describe some strategies here which can be used for decision making.

Work related to our research includes similar models that work as a decision-based system to help project managers control the cost, schedule and quality of their software projects. [1,2,6,7] One of the most well-known cost and quality estimation model is the COnstructive QUALity MOdel (COQUALMO) [5] which is an extension to the COnstructive COst MOdel (COCOMO) [4]. It consists of two sub-models: 1, defect introduction and 2, defect removal models which in turn integrate into COCOMO to help determine the cost of defects removal using the three QA practices: Automated analysis, Reviews and Testing. However, In COQUALMO the effort to fix a defect introduced and removed by each of the three practices is not quantified directly by the model; it is calculated as a percentage of the total estimated effort by COCOMO model [3]. Also, along with the COCOMO, COQUALMO are calibrated based on data manipulation of many previous software projects which may incur difficulties for an organization to tailor any of those models to itself. Moreover, with COQUALMO there are only three applicable practices without taking into account the diversity of other techniques and the variations in their efficiency.

Capture recapture models is used in software engineering to predict the total number of defect in an artifact based on a sample taken from a population of defects [10]. However, the accuracy of defect estimations given by capture and recapture models is not stable and influenced by different factors like the type of QA practice [7], number of people involved [2,6] etc.

Gou et al [3], analyzed data related to QA activities of historical projects and found that based on these data industrial

baselines can be established to estimate defect removal and detection efficiency of current and future QA activities.

Since the early days of the software industry has faced up with challenges to provide QA (Quality Assurance) software and test software in an effective and efficient manner. Our research [8,9] concluded that existing approaches for assessing and improving the degree of early and cost-effective software fault detection are not satisfactory since they can cause counter-productive behavior. An approach that more adequately considers the cost-efficiency aspects of software fault detection is required.

The aforementioned findings are alarming signals that some SDP, as our OptimalSQM solution, capable to act in old and new software engineering environments is needed. The OptimalSQM framework consists of several integrated Software expert tools that act on established rules which are based on related SDP-STP activities employed throughout the software lifecycle to positively influence and quantify the quality of the delivered software [8,9]. We proposed the rules, benchmarking and process improvement which are a self-evident need for every software development organization concerned to demonstrate value for money and to reduce the software lifecycle time and effort as well as the number of defects.

III. DEFECT PHASE CONTAINMENT ANALYSIS (DPC) MODEL

As noted before, some projects don't use the same set of sequential phases. For example, many projects do not record Unit Test (UT) defects so we could eliminate this phase from the table.

Basically, each organization needs to decide which phases to use. Our mathematical model applies to whatever set an organization chooses by Measuring Software Process Effectiveness.

A. Software Development Phases

An effective metrics program must be tightly coupled to the software development process. The metrics program and the development process are mutually supportive. The metrics program, by adding quantitative measurement, makes the development process more visible and understandable to the software development managers and team. At the same time, the development process defines integral points of data collection in support of the metrics program. Continuous improvement of the development process and the metrics program go hand-in-hand. The process need not be perfect since a primary objective of the metrics program is to identify process improvements over time. The software development process is made up of phases.

For our purposes, the development process will be divided into four phases: 1. Requirements (software systems engineering); 2. Design (analysis models and designs); 3. Implementation (coding, unit testing, subsystem testing) and 4. Testing (integration testing, system testing, acceptance testing).

Within each of these phases, unique development products are produced and analyzed for defects. A *defect* is defined as any flaw in the specification, design, or implementation of a product [7]. The IEEE defines a defect as an anomaly in a software product. Thus, a defect is a tangible flaw, either a commission or an omission, in a software development artifact that must be detected and repaired in order for the software product to be correct. A defect differs from an error and a fault in standard IEEE terminology. An *error* is the human action that results in the software containing a fault and a *fault* is an accidental condition in a software module that causes the software system to fail [8]. Therefore, uncorrected defects may create faults leading to system failure during software testing or operation. A goal of software development then is to minimize the number of defects entering into the software artifacts and to discover and repair those that do rapidly and efficiently. The two primary types of defect discovery activity are inspections and testing.

Inspections, formal and informal, are used to remove defects throughout all phases of the software development process [2- 8]. They can also be used to ensure the full functionality of the system via requirements tracing. Formal Fagan-type inspections have been shown to provide an effective means of defect removal. Inspections can be performed on critical work products, such as: a) Requirements documents; b) Analysis models; c) Software designs; d) Software code; e) Test suites and f) Documentation.

The *testing* activities of the software development process validate system functionality while identifying and eliminating remaining defects. Automated testing is performed on implemented software code. Effective testing procedures, such as the Cleanroom techniques [6], will allow the final, integrated system to be certified at a defined level of quality before it is released. The recording of defect metrics from inspections and testing over all development phases provides the base data for realizing the objectives of phase containment. The development of phase containment metrics requires that the defect data be consistent throughout the project life cycle. A major contribution of this research is the development of common defect metrics across all life cycle phases.

B. Metric definitions and phase containment analysis model

Effective phase containment metric begins with raw data on defect counts from inspections and testing activities. We will demonstrate how to implement these data counts in the Section IV. First, let us take a close look at the analysis model for phase containment. For any discovery defect, the inspection team or the testing team will determine the development phase in which the defect was introduced. This is known as the defect origin phase. The phase in which the defect is discovered is known as the defect detection and fixing phase. If the defect is found in the same phase that it originated, the defect is found "in-phase," else the defect is found "out-of-phase." The out-of-phase defects are also termed "escapes."

Let X=#DRefInP represent the number of defects found in-phase P i.e. prompt detection.

Let Y=#DROutP represent the number of defects found out-of-phase P i.e. late detection.

Let Z represent defects not yet found i.e. #CRUD (number of Customer Reported Unique Defects). The Z data will be estimated based on empirical or theoretical models (e.g. [2]**.** In this paper we supposed that Z=0 which is the case for software defect tracking when software is in operation longer than one year.

We will number the phases of software development as:

P1. Requirements phase; P2. Design phase; P3. Implementation phase; P4. Testing phase and P5. Operations phase.

The operations phase is included in the analysis to capture and analyze defects found during operation of the system. Also, the operations phase may be the origin of defects, such as operator errors. Table I captures all defect metrics for the study of phase containment, where:

$$X = \sum_{i=1}^{5} X_i \ , \quad Y_i = \sum_{j=i+1}^{5} Y_{ij} \ , \quad Y = \sum_{i=1}^{4} Y_i \ , \quad Z = \sum_{i=1}^{5} Z_i \qquad (1)$$

This matrix (DPCM) represents the collection of defect data during one pass through all phases of the development life cycle. Thus, the lower left portion of the matrix is empty. Future research, as discussed in the conclusion, will expand this analysis model to include consideration of incremental development process models in which defects may be discovered that were introduced in previous increments.

From the base defect data, several important calculated metrics can be defined:

- Let T_i = #DefInP$_i$ represent the total defects introduced in phase i. Then, $T_i = X_i + Y_i + Z_i$. The total number of defects introduced into the system is T =*#TDIns.*= $\sum T_i$, $i = 1...5$.
- Let E_i=*#ResPi*, represent the total defect escapes from phase i. Then, $E_i = Y_i + Z_i$. The total number of escapes in the system *is* $E = \sum E_i$, $i = 1...5$.
- The percentage of defect escapes from phase i is E_i/T_i. The percentage of total defect escapes from phase is E/T.

TABLE I. DEFECT PHASE CONTAINMENT MATRIX (DPCM)

	Defect detected in					
Defect origin	***P1 Req.***	***P2 Design***	***P3 Imp.***	***P4 Testing***	***P5 Ops.***	***Not found***
P1. Req.	X_1	Y_{12}	Y_{13}	Y_{14}	Y_{15}	Z_1
P2. Design	-	X_2	Y_{23}	Y_{24}	Y_{25}	Z_2
P3. Imp.	-	-	X_3	Y_{34}	Y_{35}	Z_3
P4. Testing	-	-	-	X_4	Y_{45}	Z_4
P5. Ops.	-	-	-	-	X_5	Z_5

To evaluate the cost of defect escapes during the system life cycle, a coefficient, CM_{ij}, is introduced. Parameter CM_{ij} is the cost multiplication (expansion) that occurs when a defect escapes from phase i to phase j. Initially, we will just consider expansion coefficients between adjacent phases. Using this coefficient, we can study the cost of weighted defect escapes (Cost of Quality actions – CoQ) from phase i as:

$$WE_i = \sum_{j=1}^{4} (\prod_{k=i}^{j} CM_{k(k+1)}) Y_{i(j+1)} \ , \quad i = 1...4 \qquad (2)$$

And cost to detect and fix faults injected in phase P_i is

$$CoQ\ Pi..5 = X_i * CM_{ii} + \sum CoQ\ P_{Pi->Pj},\ i=1...5,\ j=i+1,\ i+2,...,5 \qquad (3)$$

Where, the first article represent cost of prompt defects number X_i with detection and removal cost CM_{ii} , and

$CoQ\ P_{Pi->Pj} = \#DRinP_i * CM_{ij}$.

The CM_{ii} used to be normalized to 1 cost unit ($CM_{ii}=1$). The total weighted escapes (rework) is defined as WE:

$$WE = \sum_{i=1}^{4} WE_i = \sum_{i=1}^{4} \left[\sum_{j=1}^{4} (\prod_{k=i}^{j} CM_{k(k+1)}) Y_{i(j+1)} \right] \qquad (4)$$

And, finally total cost of detection and fixing of all injected defects in a project is CoQ:

$$CoQ = \sum X_i * CM_{ii} + WE \ , \ i=1...5 \qquad (5)$$

The above phase containment analysis model is readily implemented in programmable spreadsheets (see Table II) in which we present actual project data [11] in the model with accompanying data tables and charts.

In order to analyze the weighted cost of defects as they escape from phase to phase, we must estimate the CM_{ij} values.

Expansion coefficients reported in the literature range from 2 to as high as 10 [1,6,9]. Based on a preliminary study of rework times in the project, we define the following conservative cost coefficients for this study:

Req. to Design: to Imp.: to Test: to Ops. – CM_{12} = 2.5: CM_{13}= 6.25: CM_{14}= 15.63: CM_{15}= 39.07.

Design to Imp.: to Test: to Ops. – CM_{23} = 2.5: CM_{24}= 6.25: CM_{25}= 15.63

*Imp. to Test: to Ops. – CM_{34} = 2.5: CM_{35}= 6.25*l

*Test: to Ops. – CM_{45} = 2.5*l

In Table II, the weighted repair costs of defects are presented.

It is assumed that the normalized cost of repairing an in-phase defect is one cost unit [*cu*] i.e. CM_{11} =CM_{22} =$CM33$ =CM_{44} =CM_{55} =1.

The effective defect removal rate is used to measure the rate of defect removal throughout that lifecycle. The calculation involves calculating the number of defects removed at each phase of a lifecycle divided by the total number of defects discovered. This activity occurs at the various phases of a lifecycle. So for a waterfall lifecycle, you may have defect rates attributable to your requirements phase, your design phase, your coding phase, etc. This proves to be a very powerful quality measurement tool that provides insight as to the effectiveness of your quality practices. Data in Fig. 1 is an example of measuring defect removal effectiveness using average defect removal rates for defect preventive test activities (inspections, reviews etc.) and best practices data we adopted from [2] and used in our calculations model.

Rate = Ave. Defects/KSLOC	Peer Reviews			Testing				
Range	Reqs	Design	Code	Unit Test	Int Test	Sys Test	Field	Total
Insertion Rate	2.5	7.2	22.7	0.9	0.3	0.1	0.0	
Detection Rate	1.0	5.0	17.0	4.0	2.5	2.0	2.0	33.6
Leakage Rate	1.5	3.6	9.3	6.1	3.9	2.0	0.0	
Removal Effectiveness	40%	58%	65%	40%	39%	50%	100%	
Average | Best in Class*	40% | 50%	70% | 85%	65% | 85%	35% | 55%	35% | 45%	40% | 55%		
Peer Review Effectiveness->			71%	Total Effectiveness->		94%		
Best in Class->			85+%	Best in Class->		99+%		

* from Capers Jones, *Software Assessments, Benchmarks and Best Practices*, Addison Wesley, 2000

Fig.1 Example of measuring defect removal effectiveness from [2]

Our approach is to apply measurement and quality best practices is to conduct an internal assessment. This involves collecting quantitative data relating to productivity and quality indicators for a selection of projects and at the same time

collecting qualitative data about the development practices used on those same projects [9].

IV. EMPIRICAL METHOD

This section will present our empirical method of establishing quantitative management model for testing process. The three steps of the method are to: (1) identify the performance objectives to be managed quantitatively and construct data samples; (2) establish the process performance baseline for the identified metrics; and (3) analyze the correlations between the identified objectives.

TABLE II. DEFECT PHASE CONTAINMENT MATRIX AND WEIGHTED DEFECT REPAIR COSTS

DEFECTS ↓ / PHASE INSERTED :		FOUND IN PHASE : P1 Requirement	P2 Design	P3 Implementation	P4 Testing	P5 Operation	Total in Phase Pi (i=1,2,3,4,5)
CM_{ij}	Cost mult. factor	1	2.5	6.25	15.63	39.07	
P1	Requirement	184 / 67% / 91 #DRInP11 / %DRRInP11 / #ResP1 184.0 CoQP1->P1	57 / 63% / 34 #DRInP12 / %DRInP12 / #ResP2 142.5 CoQP1->P2	23 / 68% / 11 #DRInP13 / %DRInP13 / #ResP3 143.8 CoQP1->P3	8 / 73% / 3 #DRInP14 / %DRInP14 / #ResP4 125.0 CoQP1->P4	3 / 100.0% / 0 #DRInP15 / %DRInP15 / #ResP5 117.2 CoQP1->P5	275 / 22.0% / 26.2% #DefInP1 / %DefInP1 / %CoQ P1 712.5 CoQ P1..P5 [cu]
P2	Design		285 / 65% / 153 #DRInP22 / %DRInP22 / #ResP2 285.0 CoQP2->P2	101 / 66% / 52 #DRInP23 / %DRInP23 / #ResP3 252.5 CoQP2->P3	36 / 69% / 16 #DRInP24 / %DRInP24 / #ResP4 225.0 CoQP2->P4	16 / 100.0% / 0 #DRInP25 / %DRInP25 / #ResP5 250.1 CoQP2->P5	438 / 35.0% / 37.3% #DefInP2 / %DefInP2 / %CoQ P2 1,012.6 CoQ P2..P5 [cu]
P3	Implementation			365 / 68% / 173 ,, / %DRInP33 / #ResP3 365.0 CoQP3->P3	121 / 70% / 52 #DRInP34 / %DRInP34 / #ResP4 302.5 CoQP3->P4	52 / 100.0% / 0 #DRInP35 / %DRInP35 / #ResP5 325.0 CoQP3->P5	538 / 43.0% / 36.5% #DefInP3 / %DefInP3 / %CoQ P3 992.5 CoQ P3..P5 [cu]
P4	Testing				0 / #DIV/0! / 0 #DRInP44 / %DRInP44 / #ResP4 0.0 CoQP4->P4	0 / #DIV/0! / 0 #DRInP45 / %DRInP45 / #ResP5 0.0 CoQP4->P5	0 / 0.0% / 0.0% #DefInP4 / %DefInP4 / %CoQ P4 0.0 CoQ P4..P5 [cu]
P5	Operation					0 / #DIV/0! / 0 #DRInP55 / %DRInP55 / #ResP5 0.0 CoQP5->P5	0 / 0.0% / 0.0% #DefInP5 / %DefInP5 / %CoQ P5 0.0 CoQ P5 [cu]
Total in Phase Pi (i=1,2,3,4,5)		%CumDRE in P1 184 / 66.9% / 91 #DRInP1 / 184.0 CoQ in P1 / #ResInP1	%CumDRE in P2 342 / 73.8% / 187 #DRInP2 / 427.5 CoQ in P2 / #ResInP2	%CumDRE in P3 489 / 81.1% / 236 #DRInP3 / 761.3 CoQ in P3 / #ResInP3	%CumDRE in P4 165 / 94.3% / 71 #DRInP4 / 652.5 CoQ in P4 / #ResInP4	%CumDRE in P5 71 / 100.1% / 0 #DRInP5 / 692.3 CoQ in P5 / #ResInP5	%DRE (TCE) 1,250 / 94.3% / 0 #TDIns. / 2,717.6 CoQ / #CRUD

A general assumption is that the effort of defect detecting and fixing should consume a certain percentage in total development effort, and the effort of defect fixing is influenced by the defect number and the defect injected phase. In our method, three objectives have been identified including:

(1) Cost of quality (CoQ): Detecting and fixing effort means the effort for all detecting activities including test planning, test case preparing, test implementation and fix verifying.

(2) Defect Injection Distribution (DID): In general, many software organizations collect defect data for quality control. There are always some defects injected in early phases which are only detected until the testing activities even in the high maturity organizations. In our method, three primary phases, namely requirements, design and coding, are used to classify the corresponding injected phases for each defect.

(3) Percentage of Defect Detecting and Fixing Efficacy (%DRE): Fixing effort data means the effort for all defect fixing activities including defect analysis and fixing. In the testing activities, it is the common knowledge that the earlier a defect is injected, the more effort is needed to fix it. In contrast, the later a defect is injected, the less effort is needed to fix it. So, defects injected in an earlier phase, such as the requirements phase, have the effort of increasing the defect fixing effort, whereas, defects injected in a later phase, such as the coding phase, have the effort of decreasing the defect fixing effort. Our method is based on this hypothesis. There are some statistical methods which can be used to analyze the correlation between %DRE and CoQ, such as multiple regression analysis. After the correlation between %DRE and CoQ has been analyzed, the regression equation between %DRE and %DRE in every phase P_i (i=1,2,3,4) can be used to refine the estimation of fixing effort after testing. The outcome can provide a guideline to estimate the effort of defect fixing based on the defects and the distribution of injection phases. So, after testing, the project managers could re-estimate and re-plan their fixing effort effectively. The factors of regression equation could be refined and calibrated based on the historical data of software organizations. Then it can be more applicable in these organizations.

A. Strategy to Blend DPC model and Design of Experiments (DOE)

In this section, we describe the strategy to blend DPC model (Defect Phase Containment analysis model) and Taguchi's method of experimental design. Design of experiments can be applied to the "Action Planning" phase in TMM assessment procedure to prioritize actions by their statistical significance. We give priorities to improvement actions as a form of statistical significance in this research so that we can rank actions and find the best solution. This leads to optimizing action plans in organizations. It is appropriate to use design of experiments for verifying statistical significance of actions which are related to test technology. Test techniques and tools used in sub-processes influence the performance of the test process. The SPI group wants to select the best solution to maximize the performance of the test process and apply it to the organization. This enhances the effectiveness of test process improvement. The procedure to prioritize actions and select a few actions is based on our optimization results given on Table III from an industrial finished project experimental data [11].

B. An Example Optimizing Action Plans

In this section, we exemplify the use of our strategy under several assumptions as follows. There is an organization [11]

which assessed its test process with TMM which test process results are given in Table II. For this organization we analyzed weak and strong points in the test process and identified several improvement goals. To achieve the improvement goals, several actions were suggested by calculated metrics from mentioned table. When improvement actions are suggested, we select a few actions to be analyzed and design experiments. The reason why we select a few actions before the experiments is due to experimental cost. As controlled factors increase, the number of experiments also increases exponentially. Thus, when an organization defines the number of factors, levels, and response variables, the cost of experiments should be considered. In this example, we want to verify statistical significance of %DRE and %DRE in every phase Pi (i=1,2,3,4) and CoQ based on used testing techniques, testing tool and tester's experience through controlled experiments. Based on TMM and DOE assessment results, the organization decides that it is necessary to select the best test technology in its environment.

Controlled factors and levels of factors are shown in Table III. Factors include four %DRInPi (i=1,2,3 and 4) that can be reached by applying various testing techniques, testing tools, and tester's experience, each of which has three levels. The total number of treatments is nine. To reduce any possible bias of experiments, we should measure several times at each treatment. We define the response variables (%DRE and CoQ) and their relative errors calculated by regression equation (6) in Table III, too. In this example, we exploited the DOE using Taguchi approach can economically satisfy the needs of problem solving and product/process design optimization projects.

TABLE III. EXPERIMENTAL DEFECT REMOVAL EFFICACY and COQ

Exper. No. run	F1	F2	F3	F4	DRE %	DRE Rel. Err. SRM	CoQ No.1 [cu]	CoQ Rel. Err. SRM
1	0%	0%	0%	0%	0.00%	0.4%	20953	-2.1%
2	0%	70%	65%	35%	86.33%	9.2%	4075	-14.9%
3	0%	85%	85%	55%	96.52%	-1.2%	2376	-24.3%
4	40%	0%	65%	55%	85.63%	-15.2%	5747	24.0%
5	40%	70%	85%	0%	82.59%	10.1%	2807	-2.2%
6	40%	85%	0%	35%	67.35%	35.3%	5064	-16.0%
7	50%	0%	85%	35%	91.32%	4.7%	4102	-30.6%
8	50%	70%	0%	55%	74.44%	2.7%	5393	18.4%
9	50%	85%	65%	0%	82.53%	8.6%	3161	-4.7%
		Best in class %DRE results						
10	50%	85%	85%	55%	96.63%	-2.51%	2,031	29.0%
No test activity Level 1=0%	Average Level 2=40%		Best in class Level 3=50%		AverRE%	5.2%		-2.4%
					SDV	12.3%		19.3%
F1=%DRE (TCE) in P1	F2=%DRE (TCE) in P2	F3=%DRE (TCE) in P3	F4=%DRE (TCE) in P4					

By learning and applying this technique, engineers, scientists, and researchers can significantly reduce the time required for experimental investigations. DOE can be highly effective when you wish to:

- Optimize product and process designs, study the effects of multiple factors (i.e. variables, parameters, etc.) on the performance, and solve production problems by objectively laying out the investigative experiments.
- Study influence of individual factors on the performance and determine which factor has more influence, which ones have less. The information from the experiment will tell you how to allocate quality assurance resources based on the objective data.

Taguchi developed a method for designing experiments to investigate how different parameters affect the mean and variance of a process performance characteristic that defines how well the process is functioning. The experimental design proposed by Taguchi involves using orthogonal arrays to organize the parameters affecting the process and the levels at which they should be varies. Instead of having to test all possible combinations like the factorial design, the Taguchi method tests pairs of combinations. This allows for the collection of the necessary data to determine which factors most affect product quality with a minimum amount of experimentation, thus saving time and resources.

Taguchi's emphasis on minimizing deviation from target led him to develop measures of the process output that incorporate both the location of the output as well as the variation. These measures are called signal to noise ratios. The signal to noise ratio provides a measure of the impact of noise factors on performance. The larger the S/N, the more robust the product is against noise. Calculation of the S/N ratio depends on the three experimental objective:

1. Bigger-the-Better

$$\frac{S}{N_{(Bigger)}} = -10\log\left(\frac{\sum_{i=1}^{n}\left(1/y_i^2\right)}{n}\right)$$

2. Smaller-the-Better

$$\frac{S}{N_{(Smaller)}} = -10\log\left(\frac{\sum_{i=1}^{n} y_i^2}{n}\right)$$

3. Nominal-is-Best

$$\frac{S}{N_{(No\ min\ al)}} = 10\log\left(\frac{\overline{y}^2}{s^2}\right)$$

Where y_i is process output (in our case study %DRE – Bigger-the-Better objective and CoQ – Smaller-the-Better objective) of testing process, *n*-number of experiments, *s* – output standard deviation.

We applied Taguchi screening designs for three levels of each influence factors: F1=%DRE in P1, F2=%DRE in P2, F3=%DRE in P3 and F4=%DRE in P4, using Orthogonal array design plan L9 (see Table III). Each factor was varied at 3 levels: %DRE at Level 1=0% which means that company doesn't apply any test activity to the phase, %DRE at Level 2= 40%;70%;65%;35% (per Reqmnts.; Design; Impl.; Test phase respectively), which is Average test efficacy per phase and Best in class test efficacy with %DRE Level 3= 50%;85%;85%;55% (per Reqmnts.; Design; Impl.; Test phase respectively).

To analyze our project Defect Phase Containment data we used MINITAB ver.16 statistical software tool. Some results are given in Fig. 2 and Fig. 3. The main effects plot and the intersection plot are useful tools for visualizing and analyzing the effects for factors. In this paper, we only use the main effects plot because we conclude from MINITAB 16 Taguchi experiment results. The main effects plots for factors are shown in Figure 2. High slope of line means that the factor gives more impact on the experimental results than other factors. Referring to the %DRE column of Table III and the slope of Figure 2, we observe that the most influential factor is the F3 factor (%DRE

in P3), then F4 (%DRE in P4), and F2 (%DRE in P2) factor in our experiments.

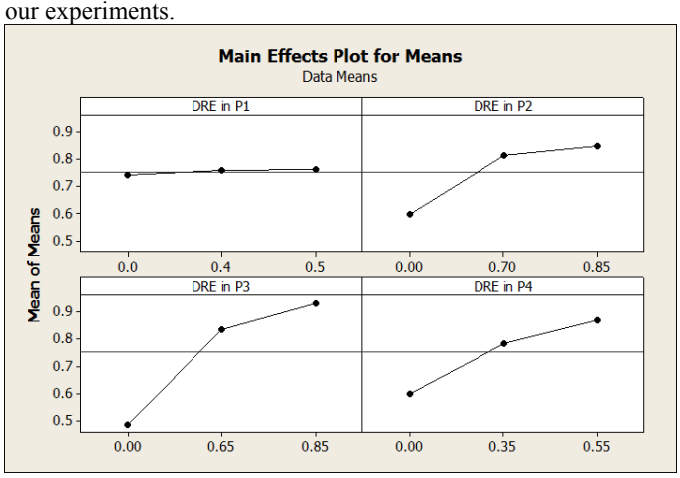

Fig. 2 Main Effects plot for %DRE (TCE) means

Furthermore, we can estimate the optimal combination of treatments from Table III it is combination of best practices values for %DRE in phases i.e. experiment No. 10 with lowest CoQ equal 2,031 [cu]. We can identify the best combination easily among levels of factors with these effects plots. We can choose to omit the insignificant terms from the model and pool their effects into the error term.

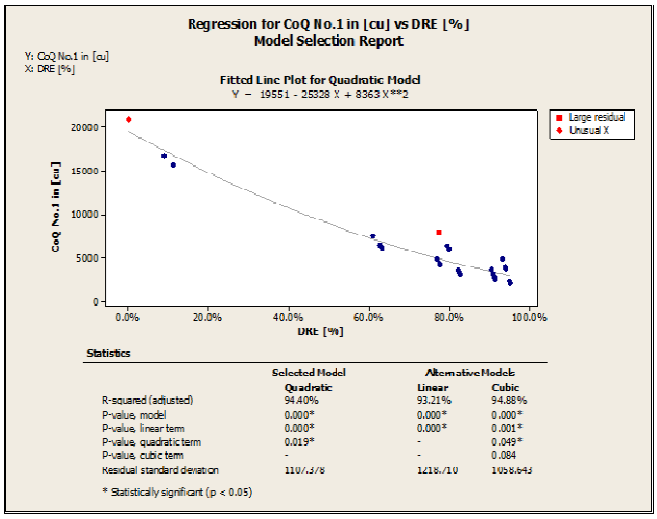

Fig. 3 SRM result of CoQ vs. %DRE fitting Quadratic model

We run Surface Response Modeling (SRM) method in MINITAB 16 for selected Taguchi's L9 experimental model and derive an equation in terms of coded factors:

The total *%DRE = 0.00468 +0.07394*F1 +1.09474*F2 +1.21111*+0.21707*F4-0.14846*(F2*F2) -0.15724*(F3*F3) -0.41916*(F4*F4) -0.04144*(F1*F2) -0.04063*(F1*F3) -0.05042*(F1*F4) -1.19177* (F2*F3+0.16794*(F2*F4)* (6)

Then, we apply, again, SRM method in MINITAB 16 software tool and found quadratic regression equation (see Fig. 3) that fits the best to estimate CoQ depending on %DRE of the new similar ongoing project in next form:

$$CoQ\ [cu]=19551-25328*\%DRE+8363*\%DRE^2 \quad (7)$$

These equations we can use to forecast at the beginning of new company's project to estimate final %DRE depending on achieved defect removal efficacy in ongoing phases P1, P2 … and after that using equation (7) project manager can estimate final CoQ in [cu] in real time very easy.

V. CONCLUSIONS AND NEXT STEPS

In this paper, we suggest a strategy to optimize action plans in an organizational test process by applying design of experiments to the TMM assessment procedure, and we illustrate this strategy with an example. We apply statistically verifying the effectiveness of improvement actions which are suggested through a test process assessment procedure applying Taguchi's Design of Experiments method and SRM (Surface Response Method) to find parametric equation for cost of quality (CoQ) related to DRE (Defect Removal Efficacy).

Our strategy has several benefits. First, it supports the selection of the best solution among alternatives. It makes test process improvement more effective by adopting the best solution suited to the organizations. Table II present „as-is" state of SDLC test effectiveness %DRE=94.3% and CoQ=2,717.6 [cu] which can be improved „to-be" state, applying best in class techniques, improving %DRE to 96.63% and decrease CoQ to 2,031 [cu]. This result is impressive, only 2.33% increase of %DRE decreases CoQ for ≈34%. Second, it gives a chance to measure feasible risks before organizations implement actions, so that they prevent or minimize risks in advance.

ACKNOWLEDGEMENTS

This work has been done within the project 'Optimal Software Quality Management Framework', supported in part by the Ministry of Science and Technological Development of the Republic of Serbia under Project No.TR-35026.

REFERENCES

[1] A.R. Hevner, "Phase containment metrics for software quality improvement", Information and Software Technology, 39 (1997) 867-877.

[2] C. Jones, "Software Assessments, Benchmarks, and Best Practices", Addison-Wesley Professional: Boston, MA. 2nd ed. McGraw-Hill, New York, 2000.

[3] Gou.L,Wang.Q, Yuan.J, Yang.Y, Li.M, Jiang.N, Quantitative defects management in iterative development with BiDefect, Software Process: Improvement and Practice, 14,4, pp.227-241, 2008.

[4] Boehm BW, Horowitz E, Madachy R, Reifer D, Clark BK, Steece B, Brown AW, Chulani S, Abts C. 2000. Software Cost Estimation with COCOMO II. Prentice Hall PTR: Upper Saddle River, NJ.

[5] Chulani.S, Boehm.B, Modeling Software Defect Introduction Removal: COQUALMO (Constructive QUALity MOdel), USC-CSE-99-510, The Center for software Engineering, University of Southern California, Los Angeles, CA, 1999.

[6] D. Galin, "Software Quality Assurance: From theory to implementation" Pearson Education Limited, ISBN 0201 70945 7, 2004..

[7] S. H. Kan, "Metrics and Models in Software Quality Engineering," Second Edition, Addison-Wesley, 2003.

[8] Lj. Lazić and N. Mastorakis, "Cost Effective Software Test Metrics," WSEAS TRANSACTIONS on COMPUTERS, Issue 6, Vol. 7, no 6, 2008, pp. 599-619.

[9] Lj. Lazić, "Software Testing Optimization by Advanced Defect Management," ComSIS, vol. 7, no 3, 2010, pp.459-487.

[10] Briand.L, El Emam.K, Freimut.B, Comparison and Integration of Capture-Recapture Models and the Detection Profile Method, Procs. Ninth International Conference on Software Reliability Engineering, Paderborn, Germany, pp. 32-41, 1998.

[11] http://www.isixsigma.com/methodology/voc-customer-focus/software-inspection-value-added/.