International Journal of Innovations in Engineering and Technology (IJIET)
http://dx.doi.org/10.21172/ijiet.102.22

Application Example of Triz and Taguchi's Robust Design Techniques to Software Testing

Ljubomir Lazic[1]
1Faculty of Information Technology, METROPOLITAN University, Belgrade, Serbia

Abstract- This paper briefly points to the synergy between innovative TRIZ (Theory of inventive problem solving) i.e. creating Software Quality through invention tools of TRIZ and Taguschi's Design Of Experiments (DOE) methodology and techniques is described. All these techniques are aimed at improving the quality of a product/service. Then, a review of case study and a proposed sinergetic use of DOE TAGUCHI method is explained i.e. Software Quality Improvement using DOE Taguchi Method to analyze, extract, and eliminate contradictions and harmful effects and generate innovative solution concepts to make better, faster and cheaper software testing process. In this paper, we suggest a strategy for optimization of the action plans in the test process by applying TRIZ and Taguschi's Design Of Experiments to the Testing Maturity.

Keywords – Software testing, TRIZ, DOE, TAGUCHI

I. INTRODUCTION

The software development industry spends more than half of its budget on QA&QC (Quality Assurance and Control) and maintenance related activities. Software testing provides a means to reduce errors, cut maintenance and overall software costs. The importance of software testing has been emphasized more and more, as the quality of software affects its benefit to companies significantly [1-4]. It has been desired for a long time to make TRIZ applicable to the issues related to software quality applications and software-based systems [9]. Our research [5-7] proposed a new methodoogy for enhancing the software quality at the lowest cost of large systems during software development and testing process (SDLC-STLC). An approach that more adequately considers the cost-efficiency aspects of software fault detection is required. The method bridges contributions from mathematics, design of experiments, software test, and algorithms for application to usability testing. Also, this paper presents an approach for software testing process (STP) optimization study finding optimum combination of software defect detection techniques (DDT) choices for every software development phase that maximize all over Defect Removal Effectiveness (DRE) of STP. The optimum combination of software defect detection techniques choices were determined applying orthogonal arrays constructed for post mortem designed experiment with collected defect data of a real project [5].

This paper presents a set of best practice models and techniques integrated in optimized and quantitatively managed software testing process (OptimalSQM). This solution expanded testing throughout the software development lifecycle (SDLC) in a framework Business Intelligence Service Architecture (BISA - www.bisa.rs) which implemented synergy between innovative TRIZ and DOE TAGUCHI methods. STLC framework BISA provides a battery of methods and tools to support the work of the quality engineer and the quality manager [4], [5]. This paper describes the Software Quality & Testing Metrics Model, which is a framework for establishing and interpreting in-process metrics in software development. The model has been validated and used on large sample of software projects in a mature software development organization. The central issue for in-process metrics, the concept and definition of the model, and its use are discussed. Example of metrics real-life projects are provided, too.

The rest of the paper is organized as follows. Proposed TRIZ and TAGUCHI'S Robust Design Techniques To Software Testing are explained in section II. Experimental results are presented in section III. Concluding remarks are given in section IV.

II. TRIZ AND TAGUCHI'S ROBUST DESIGN TECHNIQUES TO SOFTWARE TESTING

2.1 Theory of inventive problem solving (TRIZ) and DOE Taguchi approach–

TRIZ is an approach for systematic creativity. TRIZ (/ˈtriːz/; Russian: теория решения изобретательских задач, teoriya resheniya izobretatelskikh zadach, literally: "theory of the resolution of invention-related tasks") is "a problem-solving, analysis and forecasting tool derived from the study of patterns of invention in the global patent literature" [8]. In English the name is typically rendered as "the theory of inventive problem solving", [9],[10] and occasionally goes by the English acronym TIPS. Modifications and derivatives:

- SIT (systematic inventive thinking)
- USIT (unified structured inventive thinking)
- Trizics (Methodology for the systematic application of TRIZ)

Main purpose of this approach is to form a guiding theory system for new product's design innovation, with the condition of researching the scientific principles and rules regulated in the processes of human's invention and solving technical problems, and also making a summary of them [9]. Fig. 1 shows the structure of TRIZ problem-solving steps.

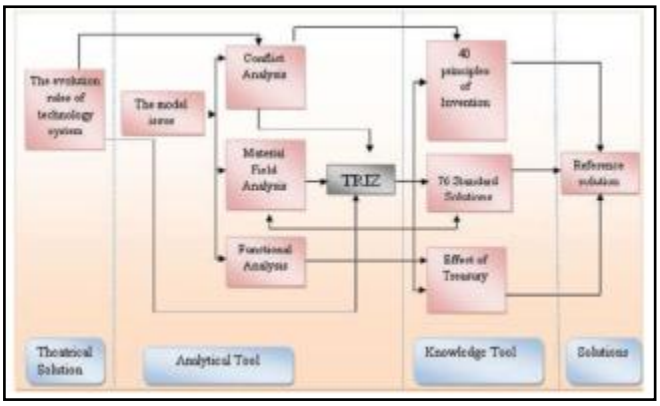

Figure 1. The structure of TRIZ problem-solving steps

Development of high quality software is very complicated and unreliable task, but the management of software development and testing process is much harder without appropriate software quality metrics and measurement establishment that assure process testing quantitative management in order to increase efficiency of software bugs detection and removal [5], [6]. The DOE using Taguchi approach [12] can economically satisfy needs of problem solving and product/process design optimization. Taguchi method involves reduction of variation in a process through robust design of experiments.

Many companies employ a defect containment strategy in an attempt to reduce software costs and increase software quality.

The testers should verify test cases at the end of test design, check the conformance of test cases to meet the requirements. It should also check the specification coverage. Validation is after test execution. Knowing the result, an effectiveness rate should count, and if it is under the threshold, the test suite should be analyzed, and the test process should be corrected.

A simple metric for effectiveness, which is only test suite dependent [3]. It is the ratio of bugs found by test cases (*Ntc*) to the total number of bugs (*Ntot*) reported during the test cycle (by test cases or by side effect):

$$TCE = 100 * N_{tc} / N_{tot} \ [\%] \quad (1)$$

This metric can evaluate effectiveness after a test cycle, which provides in-process feedback about the actual test suite effectiveness. To this metric a threshold value should create. This value is suggested to be about 75%, although it depends on the application.

A key metric for measuring and benchmarking the OptimalSQM [7,11] by measuring the percentage of possible defects removed from the product at any point in time. Both a project and process metric – can measure effectiveness of quality activities or the quality of a all over project by:

$$DRE\% = E/(E+D) \ [\%] \quad (2)$$

Where E is the number of errors found before delivery to the end user, and D is the number of errors found after delivery. The goal is to have *DRE%* close to 100%. The same approach is applied to every test phase denoted with i :

$$DRE_i = E_i / (E_i + E_{i+1}) \quad (3)$$

Where E_i is the number of errors found in a software engineering activity i, and E_{i+1} is the number of errors that were traceable to errors that were not discovered in software engineering activity i. The goal is to have this DRE_i approach slose to 100% as well i.e., errors are filtered out before they reach the next activity.

A major contribution is development of common defect metrics across all lifecycle phases that allow us to quantify process performance in terms of the "defect containment matrix" of each process stage for use in process improvement and estimating rework costs of similar projects, within the same company, using Taguchi's Design of Experiments method. Also, we provided Robust design approach for identifying factors that influence the Defect Removal Efficacy (DRE%) and Cost of Quality (CoQ) in order to find unique parametric equation for cost of quality,

which is in turn related to defect removal efficacy for their across-companies and across-project predictions with satisfactory precision [7].
Surface Response Method (SRM) is applied to find parametric equation for Cost of Quality related to overall Defect Removal Efficacy. These methods are often employed after you have identified a "vital few" controllable factors (DRE% in every phase) and you want to find the factor settings that optimize the response – the cost of quality. For demonstration purpose we will use some published results in our work [7].

2.2 Problem formulation –
According to Capers Jones [4], there are at least five types of defects that can occur. These are requirements defects, design defects, code defects, documentation defects, and bad fixes, which are defects that are induced accidentally when correcting another defect [3],[4]. The challenge to software engineers (developers and testers) and managers is to detect defects in these areas when bug or fault or error is entered or as early as possible. It is much less expensive to find and correct defects as early as possible when their costs are relatively low [3-5]. Our research [5-7] 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. We proposed the rules, benchmarking and process improvements in order to reduce the software lifecycle time and effort, as well as the number of defects.

2.3. Application of TRIZ Method for Defect phase containment analysis model –
Throughout the paper we use the terms defect and error interchangeably. A defect is also referred to in the literature as a program fault, and refers to a flaw or problem within the software. An error refers to the underlying cause of the defect. The term "error" implies a mistake that the developer has made. In our experience, the cause of most defects can be traced to human error. Defect relevant measures such as defect density, numbers of defects introduced, detected, and removed per development phases, etc. serve as important in-process indicators for monitoring, controlling and predicting the quality of software product [1-4].

2.4. Software Development Phases
An effective metrics program must be tightly coupled to the software development process, i.e. they are mutually supportive [5]. The metrics program, by adding quantitative measurement, makes the development process more visible and understandable to the software development managers and team [3]. 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. Of course, the process is never perfect and therefore, a primary objective of the metrics program is to identify process improvements over time. For our purposes, the development process will be divided into five phases:

1. Requirements (software systems engineering)
2. Design (analysis models and designs)
3. Implementation (coding, unit testing, subsystem testing)
4. Testing (integration testing, system testing, acceptance testing)

2.5. Operations (software behavior after release).
Within each of these phases, the unique development products SWA - software artefact (software code, dokumentation etc.), are produced and analyzed for defects. Developer use knowledge and software development tools to produce software artefact in each SDLC phase.
Ideal developer function require not to make any bug or fault or error, but in practice developer makes bugs and we define SWA_{Accept} as acceptable level of software quality (Useful function) which leed to CoQ_{Accept} – acceptable cost of produced software quality as next Useful function in the SDLC process. Using TRIZ S-Field model this is depicted in next figure.

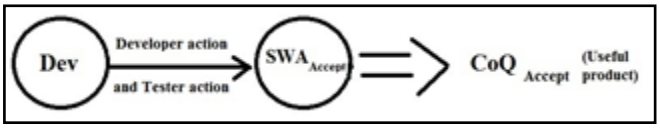

Figure 2. Developer and Tester Useful function to produce useful SWA

In each SDLC phase some test activities are planned but with unsuficient defect removal efficacy which result is useful but insufficient SWA depicted in next figure.

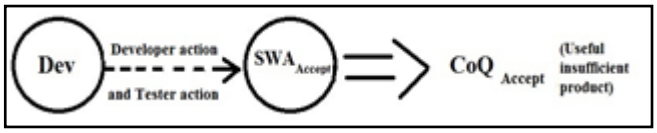

Figure 3. Developer and Tester Useful function to produce useful but insufficient SWA

Finally if software quality project management is not adequate, at the end of SDLC, the result is a software product that can not be released (handed over to the buyer) to use or to be sold on the market. In TRIZ terminology it is H.P. a harmful (unwanted) product [9] depicted in next figure.

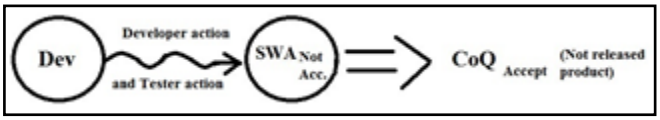

Figure 4. . Developer and Tester Harmful function not to release SWA

The two primary types of defects discovery od Tester activities are inspections and testing. Inspections, formal and informal, are used to remove defects throughout all phases of the software development process. They can also be used to ensure the full functionality of the sy stem via requirements tracing. Formal Fagan-type inspections have been shown to provide an effective means of defect removal. Inspections can be performed on critical products (SWA - software artefact), such as:

- Requirements documents
- Analysis models
- Software designs
- Software code
- Test suites
- Documentation (User manual, Installation manual, Maintenance manual, etc.)

The testing activities of the software development process validate system functionality while identifying and eliminating remaining defects. The recording of defect metrics from inspections and testing during all development phases provides the base data for accomplishing the objectives of phase containment. The development of phase containment metrics requires from defect data to be consistent throughout the project life cycle with five or more pahses as presented in next figure.

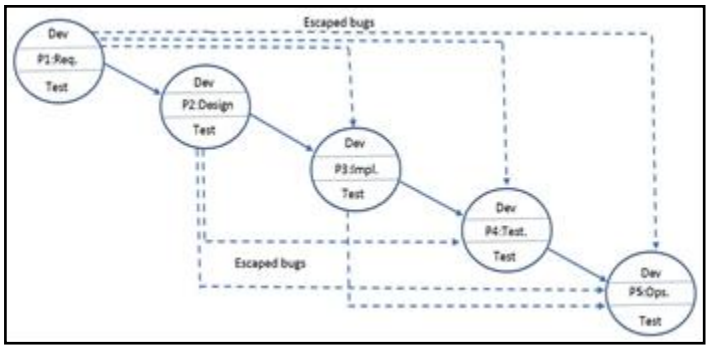

Figure 5. Developer and Tester Useful function to produce useful software trough SDLC-STLC with five pahses

III. EXPERIMENT AND RESULT OF OPTIMUM DDT COMBINATION SELECTION AND OPTIMIZATION STUDY FOR AN SDLC-STLC

Traditionally, the objective in a Multidisciplinary design optimization (MDO) study has been to search the design space to determine the values of design variables (such as DDTs- Defect Detection Techniques) that optimize a performance characteristic (such as DRE%) subject to software testing process constraints. However, research shows that up to 82% of the life software testing cycle cost is committed during the early design phase [1-4]. Therefore, significant cost savings could be realized if designers and test managers were better able to evaluate their designs on a cost basis. Response surface methods (RSM) can be utilized for MDO in cases where computerized design tool integration is difficult and design effort is costly.

In order to help project and test managers to make a forecast during the planning phase for required quality metrics *DRE%* and *CoQ*, we applied Surface Response Modeling (SRM) method in Minitab 16 in order to find second order regression equation for DRE% as function of controlled parameters *DRE%* in P_i *(i=1,2,3,4,5,6,7)* in the form:

$$y = b_0 + \sum_{i=1}^{n} b_i X_i + \sum_{i \neq j,1}^{n} b_{ij} X_i X_j + \sum_{i=1}^{n} b_{ii} X_i^2 \quad (4)$$

where : $\hat{y}$ – is approximation of output variable i.e. *%DRE* in our case, n – is number of factors (7 in our case), b_0 , b_{ij} – regression coefficients, and X_i – real i^{th} factor values in the experiment (*DRE in P1; DRE in P2; …; DRE in P7* in our case).

This study focuses on rapid multidisciplinary analysis and evaluation-on-a-DRE maximum-basis for DDT combination choices selection for each test phase activities in an traditional SDP i.e. P1- software requirement, P2- High level design, P3- Low Level Design, P4- code under test, P5- integration test, P6- system under test and finally P7- Acceptance test, recall our works [5-7] . Different Defect Detection Strategy and Techniques options, together with critical STP variables performance characteristics (e.g. DRE%, cost, duration), are studied to optimize design, development, test and evaluation (DDT&E) cost using orthogonal arrays for computer experiments [5], [7]. Calculus-based optimizers could not have been used in this case since material and technology options selection require the study of design variables that have discrete values. This study has the following steps:

3.1. Identify the design variables to be studied and alternative levels

In this study, design of maximum DRE percentage of STP optimization problem solving with best DDT choice combination in each phase P1 to P7 as controlled variables values is determined by designed experiment plan using orthogonal arrays designed for this computer experiment. To simplify the analysis such as decreasing factor's values (only three DDT number) applying Borda Ranking of DDT candidates with highest rank, several design disciplines were decoupled from the present analysis.

Seven major test phases P1 to P7 for accounting maximum DRE percentage all over STP fault injection and removal model (see Figure 5) for DDT candidate selection in each test phase were determined. These were the Static Test Techniques – TT1 (consisting of three DDTs as one factor in optimization experiment applying Orthogonal Arrays as Optimization Strategy), the TT2 i.e. DDT4= CEG+BOR+MI [6] and TT3 – Hybrid Detection Technique= DDT5 (consisting of Category Partition, Boundary value analysis, Path testing etc.). The objective of this investigation was then to determine the best combination of Test Techniques options for the seven major test phase activities sections optimized for STD&STP maximum DRE percentage under cost and time constraints according to OptimalSQM benefit index maximization [5]- [7].

3.2. Design the experiment and select an appropriate orthogonal array

Owen [15], lists a set of orthogonal arrays for computer experiments. For this study, an orthogonal array that enables the study of seven variables with three levels each was selected (http://lib.stat.cmu.edu/designs/owen.small). If a full factorial design where all possible variable/TT combinations studied would have required 2,187 (37) experiments while in Orthogonal Array design of experiment plan only 18 experiments are enough as shown in Table 1. Variable interactions were assumed to be insignificant for this study.

3.3 Conduct the orthogonal array experiments

The eighteen matrix experiments were conducted using a DRE% estimating relationships in an post-mortem real project data doing "what-if" analysis i.e. which DRE percentage of all over IOSTP will be reached if we combine DDTs in different way par test phase activities (P1 to P7) according to Borda ranking result and Orthogonal Array design of experiment plan in 18 experiments from table in Table 1 where TT1 is coded as 1, TT2 as 2 and TT3 as 3.

The analysis results of the 18 experiments for DRE percentage and corresponding TT selection per each test phase are presented in table in Table 2. Note that Full Fractional Experiment requires 37 = 2187 experiments for the same purpose. For the 18 combinations of DDTs presented in table in Table 1, the highest DRE% is 94.44 % (experiment number seven). In real project we are investigated, test team applied TTT combination as in row 3 (shaded) in Table 1, which resulted in DRE% = 87.66 %.

3.4 Analyze the data to determine the optimum levels and verify results

The average DRE (%) for each variable P and for each of the three levels i.e. TT are calculated and displayed in the response table given in Table 2. This response table shows the DRE (%) effects of the variables at each level. These are separate effects of each parameter and are commonly called main effects.

Table - 1 The "what-if" analysis results of DOE Taguchi experiment [5,6]

Exp.No.	P1	P2	P3	P4	P5	P6	P7	DRE (%)
1	1	1	3	2	3	1	1	90.51
2	3	2	2	1	1	3	1	84.79
3	2	3	1	3	2	2	1	87.66
4	1	2	1	1	2	1	2	91.27
5	3	3	3	3	3	3	2	80.34
6	2	1	2	2	1	2	2	81.66
7	1	3	2	1	3	2	3	94.44
8	3	1	1	3	1	1	3	83.14
9	2	2	3	2	2	3	3	82.99
10	1	3	1	2	1	3	1	87.89
11	3	1	3	1	2	2	1	85.05
12	2	2	2	3	3	1	1	89.77
13	1	1	2	3	2	3	2	83.91
14	3	2	1	2	3	2	2	85.19
15	2	3	3	1	1	1	2	81.72
16	1	2	3	3	1	2	3	84.11
17	3	3	2	2	2	1	3	82.58
18	2	1	1	1	3	3	3	92.94

The optimum level (TT) for the design variables (P) can now be selected by choosing the level with the highest DRE percentage presented in next table.

Table - 2 The DRE (%) Response table per phase P (P1..P7)

Phase/ TT	*P1*	*P2*	*P3*	*P4*	*P5*	*P6*	*P7*
1	**88.69**	86.20	**88.02**	**88.36**	83.91	86.35	84.02
2	86.12	**86.35**	86.19	85.14	85.58	**86.51**	**87.61**
3	83.52	85.77	84.12	84.82	**88.72**	85.50	86.70

For example the highest DRE percentage is got when variable P1 was at level 1 at 88.69 % as opposed to 83.52 % at level 3, and 86.12 % at level 2. Similarly, the levels that optimize total SDLC-STLC defect removal effectiveness (DRE%) were chosen. The optimum levels are indicated by bold & underlined in Table 2. As the next step, least squares regression analysis is used to fit the second order approximation model (Equation 4) to the DRE% data (y_i) given in Table 1 in terms of the seven design variables (Xi). This prediction parametric model accounts for the response surface curvature (square terms) and two factor interactions (cross terms) i.e. RSM:

$$DRE\ (\%) = 111.71 - 2.58\,(P1) + 1.22\,(P2) - 1.95\,(P3) - 7.61\,(P4) - 0.69\,(P5) + 0.94\,(P6) - 13.04\,(P7) - 0.36\,(P2)^2 + 1.46\,(P4)^2 + 0.79\,(P5)^2 - 0.36\,(P6)^2 + 3.15\,(P7)^2 \quad (5)$$

Note that, in this response surface approximation model, the parameter values are restricted to 1 (TTl), or 2 (TT2), or 3 (TT3). At these levels, the DRE in SDLC-STLC was predicted to be 94.03 % using a second order prediction model (3). As a next step, a verification analysis was performed. The DRE (%) of an SDLC-STLC calculated from these test techniques choices combinations, according to the post-mortem real project data using optimized DDT choices from table on Table 3 i.e. if we chose in real project TT combinations as in Table 3, we would have DRE (%) to be 93.43 %.

Table - 3 Maximum DRE (%) value and corresponding test techniques choices per test phase solution

P1	**P2**	**P3**	**P4**	**P5**	**P6**	**P7**	**DRE (%)**
TT1	***TT2***	***TT1***	***TT1***	***TT3***	***TT2***	***TT2***	***94.03***

Difference is 0.6% = 94.03%-93.43% that is acceptable to validate our prediction model for DRE (%) in equation (4) for optimal DDT combination choice given in Table 3.

Optimal combination of DDT choices per phase P given in Table 3 made increase of about 6 percent, compared to un-optimized DDTs combination per each test phase we used in our real project in which we achieved DRE% of 87.66 % (table in Table 1, row 3).

IV. CONCLUSION

Testing is one of the most resource-intensive activities in software development and consumes between 30 and 50% of total development costs according to many studies. We described our Software Quality Optimization (OptimalSQM) strategy which is a continuous, iterative process throughout the application lifecycle resulting in zero-defect software that delivers value from the moment it goes live, with synergy between TRIZ and TAGUCHI'S design of experiments techniques. The results determined that the. 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 for the organizations. For example, data shown in Table 1 present „as-is" state of an real project test effectiveness %DRE=87.66% (Table 1, row 3) which can be improved to „to-be" state, applying best in class techniques presented by prediction equation (4), improving %DRE to 93.43% and decrease cost of software development (Table 3).

V. 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.

VI. REFERENCE

[1] D. Galin, Software Quality Assurance:From theory to implementation, *Pearson Education Limited, ISBN 0201 70945 7, 2004*.
[2] A. Frost and M. Campo, " Advancing Defect Containment to Quantitative Defect Man", *CrossTalk*, December 2007.
[3] S. H. Kan, Metrics and Models in Software Quality Engineering, Second Edition, *Addison-Wesley, 2003*.
[4] C. Jones, Estimating Software Costs. 2nd edition. *McGraw-Hill, New York: 2007*.
[5] [27] Lj. Lazić, D. Velasević: "Applying simulation and design of experiments to the embedded software testing process", *STVR, Volume 14, Issue 4, p257-282, John Willey & Sons, Ltd., 2004.*
[6] Lj. Lazić, N. Mastorakis, Orthogonal Array application for optimal combination of software defect detection techniques choices, *WSEAS TRANSACTIONS on COMPUTERS, Issue 8, Volume 7, August 2008*.
[7] Lj. Lazić, S. Milinković, Reducing Software Defects Removal Cost via Design of Experiments using Taguchi Approach, *Software Quality Journal, Springer-Verlag New York, Inc.*, ISSN:0963-9314, Volume 23, Issue 2, June 2015, pp. 267-295.
[8] G. Altshuller, H. Altov, L. Shulyak, And Suddenly the Inventor Appeared: Triz, the Theory of Inventive Problem Solving. Technical Innovation Ctr., 1996.
[9] H. Harmann, A. Vermeulen, M. v. Beers, Application of TRIZ in software development, *TRIZ Journal*, 2004.
[10] Hu M, Yang K, Taguchi S. Enhancing robust design with the aid of TRIZ and axiomatic design, Part 1. *TRIZ Journal*, October 2000, The TRIZ Institute.
[11] Lj. Lazic, Synergy Between Qfd, Doe Taguchi And Triz Six Sigma Utilization In Software Engineering, ICSD - Innovation, Competitiveness and Sustainable Development, 2nd International Conference of the Faculty of Management, Belgrade, Serbia, 25 May, 2017.
[12] G. Taguchi,, Taguchi on Robust Technology Development, *ASME Press*., 1993.
[13] M. S. Phadke, Quality Engineering Using Robust Design, *Prentice Hall, Englewood Cliffs*, NJ. 1989.
[14] R. Hevner, Phase containment metrics for software quality improvement. *Information and Software Technology*, 39, 1997. pp867-877.
[15] Owen, A.: "Orthogonal Array Designs for Computer Experiments," Department of Statistics, Stanford University, 1994.