Software Quality Enhancement Through Software Process Optimization Using Taguchi Methods

B. Kanchana
Indian Institute of Science, Bangalore, India.
Cen tre for AirBorne Systems, Defense Research and Development Organization
E-mail: kanchana@cabs.ernet.in
and
V. V. S. Sarma
Indian Institute of Science, Bangalore, India.
E-mail: vvs@csa.iisc.ernet.in

Abstract

This paper presents a methodology for selection of optimal software design parameters using the experimental design. When an organization is at the point of taking up a new project with an objective of improving the software quality, Taguchi method is appliedfor the software design process with an objective that not more than one error is found per software module. The strategy in robust design is to conduct off-line experiments using orthogonal arrays (OA) and to optimize the design by maximizing performance measures with respect to design parameters. Towards this a cause and effect diagrams for design errors was drawn with opinions from customer, production, quality personnel and engineers. This diagram gave three most likely parameters as candidate for software design error. they are coupling, number of requirements per module and McCabe's cyclomatic complexity. It was planned to consider coupling parameter at two levels as low coupling at level 1 and high coupling at level 2. in case of number of requirements per module parameter three levels were considered they are one requirement per module at level I, two requirements per module at level 2 and greater than two requirements per module as the level 3. In case of McCabe 's complexity value<5 is set at level I, value 5 to 10 is set at level 2 and value >10 is set at level 3. The possible number offactorial experiment required be conducted for levels selected for the threee parameters is 18. The appropriate orthogonal array based on the guidelines of Taguchi is L_9 that is nine experiments need to be conducted to find the optimal software design parameters.

Keywords : Quality assurance, software quality, software metrics, software process improvement, design of experiments and Taguchi methods.

1. Introduction

Recent literature on quality control in manufacturing has recognized the method of quality-based design as a more efficient alternative to the traditional approach of acceptance sampling. In particular, interest has been growing in the use of the concepts of robust design, which are attributed to the Japanese quality engineer, G.Taguchi [I]. Even though Taguchi's methods of data analysis have been under the scrutiny of statisticians [2], they are being applied with a fair degree of success in many industrial experiments [3][4]. Robust design addresses two major concerns faced by all product/process designers: i) how to economically reduce the variation of a product's/process' function in the user's environment, and ii) how to ensure that decisions found to be optimal during laboratory experimentation will prove to be so in the anufacturing and the customer environments. The strategy in robust design is to

conduct off-line experiments using orthogonal arrays, and to optimize the design by maximizing performance measures, termed signal-to-noise ratios, with respect to thedesign parameters.

In this paper, we attempt to show the applicability and relevance of Taguchi methods to engineer quality and performance into new software products and software development process. While applying these methods some characteristics of the software development process and its similarities and differences with the traditional manufacturing scenario are to be noted. Firstly, like in manufacturing software is a product; but unlike in manufacturing it involves development and not production. We do not reproduce the same object, but each product and each of its

O-7695-0028-5/99 $10.00

versions are different from the previous versions, Models for statistical quality control are considerably different and there is lack of useful models that allow us to reason about the software process, the software product and the relationship between them [5][6].

In this preliminary paper, we demonstrate the application of Taguchi methods, to optimize the software architecture design parameters in the software development process especially in the design phase of its life cycle. If the software user is experiencing an unacceptable number of faults, the common solution would be to improve the underlying software development process. But if the project manager wants to incur this extra expense, he might have to justify it quantitatively. Problems of this nature constantly arise in software industry. Before such large investments are made, one ought to see if the errors can be reduced through better design decisions. Software system developers are skillful in approximately setting of software design parameter values. Considerable additional experimentation is necessary, to obtain the information needed to do a design optimization. Classical statistical experiments, called full factorial designs requires trials under all combinations of factors. Taguchi has shown that if one runs orthogonally designed experiments instead, many product and process designs can be optimized economically and effectively [7].

The paper is organized as follows: In section 2, we discuss about robust design. The motivation of using this method in the context of an existing real-time software system development is explained in section 3. The software design parameters and their levels are described in section 4. Orthogonal array experiment, preliminary results, experimental verifications from this study are presented in section 5 and conclusion in section 6.

2. Parameter design

Robust product design aims at the selection of proper software design techniques, testing techniques, management techniques and nominal operating conditions so that the product will be producible at minimum cost. The three steps involved in robust design are [7] are:

1. Planning the statistical experiments is the first step and includes identification of the product's main function(s), constituting failure. This planning step spells out the quality characteristics Y to be observed, the control factors $\{\theta_1, \theta_2, \theta_3\}$, the observable noise factors $\{w_1, w_2, w_3\}$, and the levels at which they will be set during the various test runs (experiments). It also states which orthogonal design will be applied to conduct statistical experiments and how the observed data $\{y_1, y_2, y_3, y_4, \ldots\ldots y_n\}$ will be analyzed.
2. Actual conducting of the experiments.
3. Analysis of the experimental observations to determine the optimum settings for the control factors, to predict product performance at these settings, and to conduct the validation experiments for confirming the optimized design and making plans for future actions. Taguchi has[4], [7] recommended that one should analyze this, using specially transformed form of the performance characteristics Y, known as the signal-to-noise ratio $\{S/N(\theta)_j\}$, rather than using the observed response $\{y_1\}$ directly.

3. Motivation

In this section, we describe the real-time computer system used as on-board system of an Airborne Surveillance Platform (ASP). The development of an ASP involves integration of a large number of diverse subsystems which typically include Airborne Surveillance Sensor 1 which is active, a Navigation system, Sensor Data Processors, Bus Controller, Backup Bus Controller, Man Machine Interface (MMI) and display system as shown in Fig. 1.

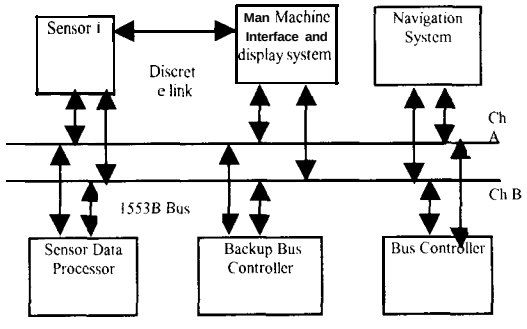

Fig. 1. ASP System Configuration

Each subsystem is configured as a Remote Terminal (RT) on the 1553 B bus. In order to develop a dependable ASP, it is necessary to check the dependability of the software resident on each subsystem. Sensor 1 detects targets from an elevated platform with a certain range. Sensor Data Processor processes the raw sensor data to give a cumulative detail of the target and the data is transmitted to the display system. The Man Machine Interface (MMI) and display system provide the surveillance scenario to the operator.

The organization in charge of developing ASP is presently designing an enhanced version of the ASP software. The aim is to reduce the number of errors introduced in the design phase per module. MMI and

display system software architecture is shown in Fig. 2. The total ASP system is controlled by the operator from the MMI. The MMI consists of user interface device to provide various facilities such as, access protocol, equipment control and operation facility. The display system provides the air surveillance situation on a tactical display with appropriate symbology and colour codes to indicate the sensor correlation and classification respectively. An interactive menu display is provided to aid the operator all through the mission. The surveillance is carried out using a tactical and a tote display with a target density of the order of a few hundreds. Target information monitored is in the order of thousands of words per scan of a few seconds for period of 4 to 6 hours. This briefly gives us an idea of the software complexity and the timing requirements of the MMI and display subsystem.

4. Quality enhancement through process optimization

In this paper, our focus is an ensuring enhanced software product quality by improving the underlying software development process. There are many process factors, affecting the product quality. The product metric that is considered is the number of defects introduced in the design phase of the software life cycle. A cause-effect diagram captures the causes for the same from among the process characteristics. A cause-effect diagram brings out more clearly the potential factors than a group discussion. For applying the Taguchi method in software process for the data available a representative of this cause effect diagram is applied which is shown in Fig. 3.

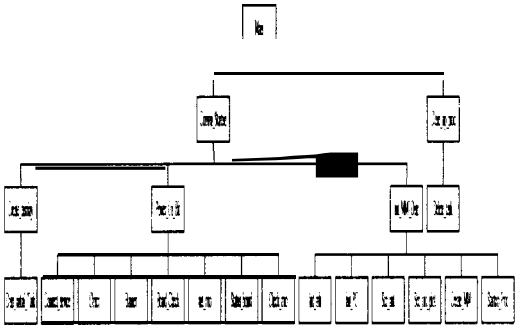

Fig 2 Software Architecture of the MMI and Display System

4.1 Choice of Process Parameters

Discussion with computer scientists and the customers led to the selection of three factors listed in Table. 2. for improving the software modules from the point of view of quality. The software parameters selected are coupling, number of requirements per module, and cyclomatic complexity [8]. The classification of coupling are data coupling, stamp coupling, control coupling, common coupling and content coupling.

4.1.1 Number of requirements per module

The software architecture of two main modules of MMI and Display software is shown in Fig. 4. The Receive_plot_data() module displays the plot data received after converting the data to the screen coordinates. It addresses one functional requirement. The Command_proc_task() receives the range ring, azimuth display and maximum range change command and displays the range rings, azimuth strobes and toggles the maximum range between 100 and 200 Kms respectively. This module addresses three function requirements. They are azimuth display, range ring display and maximum range selection. Based on the number of requirements the module address the requirement classification is carried out in this study.

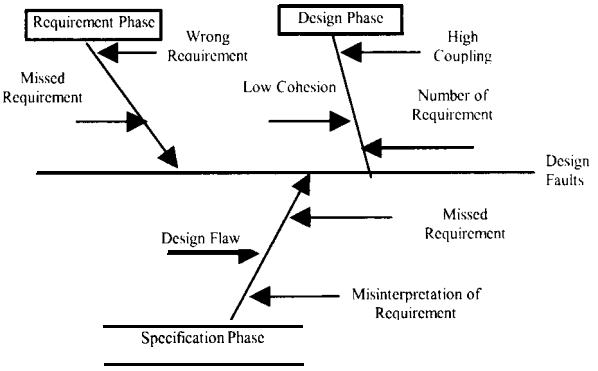

Fig. 3. A Representative Cause-and Effect Diagram for Design Faults used for design of experiments

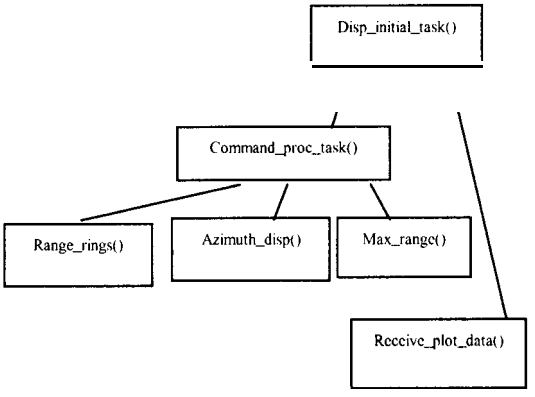

Fig. 4. Software Architecture of Disp_initial_task()

4.1.2 Coupling between modules

As long as a simple module parameter list is present i.e. simple data are passed, *data* coupling is exhibited in this portion of the software structure [9]. A variation of data coupling is called *stump* coupling is found when a portion of a data structure is passed via a module interface. *Control* coupling is characterized by passage of control between modules by a flag on which decisions are made in a subordinate or superordinate module. *Common* coupling occurs when number of modules reference a global data area. Diagnosing problems in structures with considerable common coupling is time consuming and difficult. *Content* coupling occurs when one module makes use of data or control information maintained with the boundary of another module. Secondarily content coupling occurs when branch are made into the middle of the module.

4.1.3 Cyclomatic complexity

McCabe's classic cyclomatic complexity metric simply evaluates the control graph of the module given the source code as input [8]. Cyclomatic complexity is a software metric that provides a quantitative measure of the logical complexity of a program.

The three parameters are number of requirements per module, coupling and cyclomatic complexity and the various levels used are displayed in Table. 2. This design parameters is broken down into either two or three levels as shown in the Table. 2. The first control factor is number of requirements per module which is set at three levels. At level 1 the value set is one requirement per module, at level 2 the number of requirements is set to 2 and at level 3 the number of requirements is set to greater than 2. The second control factor is cyclomatic complexity which is set at three levels. At Level 1 the modules with cyclomatic complexity less than five are considered, at level 2 the modules with cyclomatic complexity between 5-10 is considered and at level 3 the modules whose cyclomatic complexity is greater than 10 is considered. In any application it is very difficult to program without any coupling hence at level 1 the data, stamp and control coupling are considered and common and content coupling is considered at level 2.

Table. 2. Control Factors and their Levels

Label	Factor Name	Level 1	Level 2	Level 3
A	Number of Requirements *per module*	1	2	>2
B	Cyclomatic Complexity	<5	5-10	>10
C	Coupling	Data, Stamp	Common and Control	Common and Content

5.0 Taguchi Design based on orthogonal array

Details of Taguchi method and other possible application in software engineering are discard in the Ph.D thesis of the first author [10]. The factorial experiment to explore all possible factor level combinations would require $2*3^2=18$ experiments because the control factor A is at 3 levels, B is at 3 levels and C is at 2 levels. It is unnecessary to perform full factorial experiment; a system's behavior can usually be adequately characterized by relatively a few parameters. A number of orthogonal arrays have been tabulated along with associated linear graphs which can be used conveniently to construct orthogonal designs for any experimental solution [7]. The appropriate

Table. 3. Experimental design L_9 array

Experiment	A	B	C
1	1	1	1
2	1	2	2
3	1	3	3=2
4	2	1	2
5	2	2	3=2
6	2	3	1
7	3	1	3=2
8	3	2	1
9	3	3	2

orthogonal array selected for the parameters in Table. 2 is L_9 array. This array is shown in Table. 3. The assignments to the columns are shown in the Table. 3. The table has nine rows, each row showing different characteristics. For instance, row 1 composed of level 1 for each of the three control factors. As only two levels are identified for factor C, in Table. 2 and level 3 is taken to be at level 2.

The experimental procedure followed in each of the nine experiments is next described. For each experiment 3 or 4 modules were tested. The outcome of the testing process is the identification of presence of errors. The errors encountered in each module during the testing process is recorded. The errors encountered for various experiment is listed in Table. 4 the modules selected were from the real-time metric software and MMI and Display software written for iRMX environment and DOS environment. The modules are written in C- language and by the programmers who have an experience of 5 to 10 years.

In Table. 4 for instance experiment type 1 which refer to modules with McCabes complexity less than 5, the modules have data coupling or stamp or control coupling only and they address one requirement per module. Experiment type 1 was conducted on 5 modules and only 1 error was found from these modules on testing. Experiment type 2 was conducted on 5 different modules and 3 errors were found on testing of these modules. Experiment type 3 was conducted on 5 modules and 10 errors were found when they were tested. Experiment type 4 was conducted on 2 modules and 1 error was found when the modules were tested. Experiment type 5 was conducted on 3 modules and 4 errors was found when the modules were tested. Experiment type 6 was conducted on 2 modules and 4 errors was found when the modules were tested. Experiment type 7 was conducted on 3 modules and 3 error was found when the modules were tested. Experiment type 8 was conducted on 3 modules and 4 errors was found when the modules were tested. Experiment type 9 was conducted on 3 modules and 10 errors was found when the modules were tested.

To seek robustness one should measure performance by software ratios. If the performance characteristics y happens to be such that the smaller it is the better, then one express the signal to noise ratio as follows

$$S/N(\theta) = -10 \log\left[\sum_i y_i^2 / n\right],$$

where θ is the vector of control parameter [4],[7]. Where one repeats observation n times at each selected combination of design parameter sets in the control orthogonal array. In design optimization one attempts always to maximize the S/N ratio. The best setting of the design parameter is at that setting where the S/N ratio is maximum. The optimum parameter configuration needed for developing a reliable software module from Table. 4 after the analysis on the signal to noise ratio we see that the experiment type 1 gives maximum S/N ratio. Therefore in order develop a quality software it is necessary that the cyclomatic complexity should be maintained at level 1, coupling also be kept level 1 and number of requirement per module at level 1. Because of the special structure of orthogonal array used here and the manner in which one assigned the control factors A to C to OA columns, it was not difficult to calculate the effects of each of the control factors, assuming that additivity (separability) of the main effects held and there were no factor-factor interaction effects. The additive results are given is Table. 5 at the various levels. From Table. 5, we get a similar result i.e. cyclomatic complexity should be kept less than 5, number of requirement for a module should be one and the coupling should be low, for an objective function not more than one error per module. The results obtained confirm the ordinary software practices of cyclomatic complexity should be less than 10, number of requirements module be one or two and the coupling should be less.

5.2 **Experimental verification**

To illustrate the effectiveness of the methodology proposed, we have undertaken the following experiment. As an example, we considered a particular modules of the MMI and display system software i.e. set_mission_time() and Security_check().

Table. 4. Experimental results.

Experiment Type	Name of the Modules	Errors	Average error per module	Signal to noise ratio
1	Restore_interrupt(, Azimuth( ) Draw_rect_border( ) Draw_border( ) Calculate_xy_pos( )	0 0 0 0 1	0.2	6.99
2	Curr_opt_option( ) Del_com( ) New_map_coords( ) Sector_proc( ) End_proc(,	1 0 0 2 0	06	0
3	Glb3_var( ) Fan_out( ) Fan_in( ) Tx_sect_menu_proc( ) Security_clearance(,	2 2 2 2 2	2.0	-6 02
4	Time_display( ) Sys_data_in_out( )	0 1	05	3.01
5	Plot_display( ) Prev_curr_scan_disp( ) Date_time_input( )	0 1 3	1 33	-5 224
6	Elapsed_time( ) Mod1_nam( )	1 3	2 0	-6 9x
7	Receive_data( ) Display( ) Current_status( )	2 0 1	1.0	-2 2
8	Peridoc_hlth_task( ) Handle_dbit_PRF( ) Get_dbit_PRF_ack( )	1 2 1	1 66	-3 0 10
9	Overlay( ) Set_mission_time( ) Receive_plot_data( )	1 6 3	3 33	-1 1 x9

Set_mission_time()

In the present version , the characteristics of this module is as follows: cyclomatic complexity is 18, corresponds to level 3 of this control parameter. Number of requirements are setting time and setting date corresponds to level 2 of requirements control parameter (2 requirements per module) and the coupling used is common coupling corresponds to level 2 of coupling control parameter. In version 1, the top

level designs obviously done in an ad hoc manner. In order to increase the software quality this module is split into six modules using the optimal design parameters derived and they are get_input(), validate_month(), validate_year(), vaildate_date(), Validate_min_sec() and validate_hr() in version 2. The version 2 modules are tested and one error is found as against six in version 1. This confirms the orthogonal array optimized parameter when applied leads to less error prone modules.

Table. 5 Control Parameter details.

Control Parameter	S/N ratio Level 1	S/N ratio Level 2	S/N ratio Level 3
Module Complexity	5.288	-2.808	-8.234
Requirement	-2.38	-4.77	-7.52
Coupling	-2.3	-6.48852	-

Security_clearance()

In the present version , the characteristics of this module is as follows: cyclomatic complexity is 12, corresponds to level 3 of this control parameter. Number of requirements is security clearance, corresponds to level 1 (1 functional requirements per module) and the coupling used is common coupling corresponds to level 2 of this control parameter. In version 1, the top level designs obviously done in an ad hoc manner. In order to produce a reliable module this module was split into 5 modules in version 2 keeping the module complexity less than 5 and avoided the use of common and content coupling. In version 2 this module was split into CompilepasswdData(), CompileUserData(), Del_invalid_user(), Del_invalid_passwd() and Display_cursor() with appropriate parameters. The version 2 Security_clearance() module is tested and no errors were found as against 2 in version 1. This confirms the orthogonal array optimized parameter when applied leads to less error prone modules.

5. Conclusion

Understanding the functional software design parameter is an essential step in the improvement in the software quality. The Taguchi method of experimental design for the software recommends that optimal design parameter are one requirement per module, low coupling and cyclomatic complexity should be less than 5 in order to reduce the number of errors per module to one or less. Most notable of all this study demonstrates that high quality can be obtained at a low cost, by avoiding the faults to be identified at the later stages of the development phase, which was the central theme of Taguchi quality philosophy. Philosophically speaking , the software programmer can do very little alone, it is the process that has to be improved and this requires action from management. This method could be applied to any radial dimension of the spiral model to obtain the optimized parameter and to improve the quality of the final product.

Acknowledgements

The support and encouragement given by Dr K. Ramchand, Director, at Centre for AirBornc Systems (CABS), Bangalore is acknowledged with gratitude. Thanks are also due to Dr A Pedar of National Aerospace Laboratories, Bangalore for useful suggestion on an earlier draft of the paper. The authors thank the MMI and the display group of CABS for their support in collecting the defect data.

References

[1] Song, A. A., Mathur, A. and Pattipati , K. R., " Design of Process Parametes Using Robust Design Techniques and Multiple Criteria Optimization", *IEEE Trans on Systems, Man and Cybernetics,* **Vol.** 25, No. 11, November 1995.

[2] Nair, V. N., " Taguchi's Parameter design: A panel Discussion", Technometrics, vol. 34, no. 2, pp.127-161, May 1992.

[3] Bendell, A., Disney, J., and Pridmore, W. A., Eds, *"Taguchi Methods: Applications in world industry," New* York: IFS Publications and Springer-Verlag, 1989.

[4] Phadke, M. D., *" Quality Engineering Using Robust Design* ", Englewood Cliffs, NJ: Prentice-Hall, 1989.

[5] Boehm, B., " A Spiral Model for Software Development and Enhancement", *IEEE Computer,* **vol.** 21, no. 5, May 1988, pp. 61-72.

[6] Kan, S. H., " *Metrics and Models in Software Quality Engineering",* Massachusetts: Addison-Wesley, 1995.

[7] Bagchi, T. P., *"Taguchi methods Explained: Practical step to Robust Design,"* India: Prentice-Hall, 1993.

[8] McCabe, T., "A Software Complexity Measure," *IEEE Trans. Software Engineering,* Vol. 2, NO. 6, December 1976, pp. 308-320.

[9] Pressman, R., S., 1992, "*Software Engineering a Practitioner's Approach",* Mcgraw-Hill, New York.

[10] Kanchana, B., 1998, "Software Quality and Dependability Issues for the Airborne Surveillance Platform: A Systems Engineering Study", Ph.D dissertation submitted to the dept. of Computer Science and Automation, Indian Institute of Science, Bangalore, India.