USING TAGUCHI METHODS IN INFORMATION SYSTEMS QUALITY

Ion BULIGIU, Lect., Ph.D.
Valentin LIȚOIU, Prof., Ph.D.
Anca Mehedințu, Lect., Ph.D.
University of Craiova, Faculty of Economics and Business Administration

Keywords: software quality, loss function, Taguchi, information systems.

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 applied for 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 offline experiments 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 and quality personnel.

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. Even though Taguchi's methods of data analysis have been under the scrutiny of statisticians, they are being applied with a fair degree of success in many industrial experiments.

Robust design addresses two major concerns faced by all product/process designers: how to economically reduce the variation of a product's/process' function in the user's environment, and how to ensure that decisions found to be optimal during laboratory experimentation will prove to be so in the manufacturing 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 the design parameters. In this paper, we attempt to show the applicability and relevance of Taguchi methods to ensure 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 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. 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 require trials under all combinations of factors. Taguchi has shown that if one runs many orthogonally designed experiments instead, product and process designs can be optimized economically and effectively.

2. The Taguchi Loss Function for Information Systems

One of the major contributions of the Taguchi methods is the concept of loss junction. The underlying idea is that the system should have the minimum loss to society over its entire life cycle. The cost analysis and comparisons of products and services should take into account all of the following costs: planning, design, implementation, warranties, maintenance and support, disposal, upgrade, replacement [Zahedi, 1995].

Planning, design, and implementation costs. In information systems, the concept of the system's life cost is not new. The costs of planning, design, and implementation are the development costs, which are normally used in determining the development budget.

In analysis of development costs, there are hidden quality costs that should be taken into account. These are the cost of non-conformance and the cost of lost opportunities.

The cost of non-conformance is when the system is under-designed or over-designed. An under-designed system lacks some of die features that are of value to the system customers. An over-designed system has features that system customers do not need or use. A bad design could be both under-designed and over-designed at the same time, in that it lacks what customer’s value, and offers what customers do not need or use.

The cost of lost opportunities is the lost profits or benefits due to the failure of the information system to provide its customers with the needed information. Although this cost is not an easy cost to estimate before the system becomes operational (or even after it has become operational), its existence should be a significant incentive for the development team to invest adequate time in requirements analysis and careful incorporation of the requirements into the system design.

Warranty Cost. Information systems normally do not provide warranties because damage due to wrong information could be hard to assess and is potentially limitless. However, when the information system provides sensitive information to customers or contains information of high value to the corporation, it may take measures to protect against risks, such as buying insurance or extra security measures that would add to the life-cycle cost of systems.

In many systems with external customers, the service provider offers customers the option of canceling their service or getting a full refund. A poorly designed system could generate a substantial cost in this respect.

Maintenance and Support Cost. Since maintenance and support represent a large share of information-systems projects expenditure, it is natural to include the maintenance and support cost in the analysis. The projects in which development time is severely limited may require a significant amount of maintenance and support, because systems that lack adequate investment in analysis and design will require a much greater degree of correction and customer support than those with more careful design. Therefore, it is essential to include the maintenance and support cost for various scenarios of systems development in order to give the project decision makers a more realistic sense of trade-offs involved in scheduling and resource allocations in various stages of systems development [Roy, 1990].

Disposal, Upgrade, and Replacement Cost. Disposal of products such as autos and house appliances is a major environmental concern for those manufacturing companies that have a quality-based culture and a proactive approach to social issues. Is there a disposal cost for information systems?

It is interesting to observe that information systems have disposal, upgrade, and replacement costs of a different sort. An information system has three major physical components: hardware, software, and people. Once an information system is obsolete, one may ask: What are the costs of disposing of these components?

Hardware: The cost of disposing hardware is similar to the cost of disposing any obsolete machinery. Selling, scrapping, reusing, or storing the machinery would involve a cost to the company as well as to the society and environment. This cost is more critical for large projects involving many small computers, specially designed network connections, special input and output devices, or related office equipments. When a company goes into the buying mode for the hardware of a project, it should have a plan that includes the disposal cost and disposal strategies once the equipment becomes outdated.

Software: Similar consideration should be given to the obsolescence and disposal of software. The cost of disposing software once it has become obsolete is an even thornier issue. Aside from the investment in preparing the software for the system that now has no value, the obsolete system includes data and knowledge that should be transferred to the new system [Kan, 1995].

Again, considering the issue of obsolescence may have an important impact on the design decisions of the software. For example, the system that allows for a seamless data migration or knowledge transformation could prove to be a cheaper option, although its initial cost is higher than competing products.

People: Disposing an information system is also costly for the people who interact with the system:

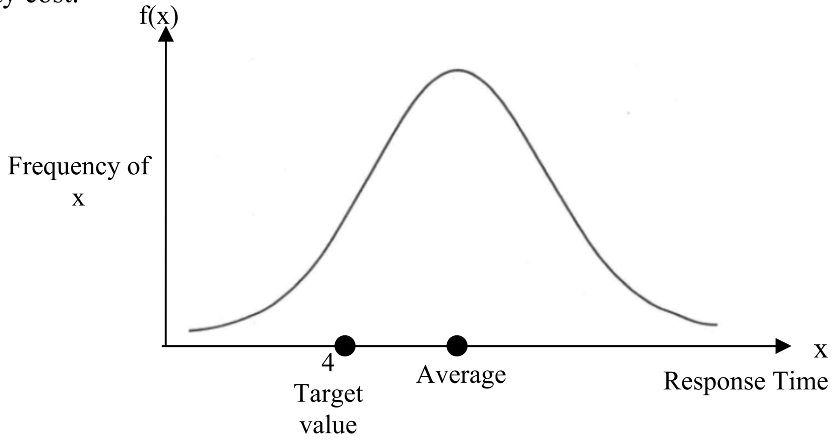
- End-users will lose all their invested time in learning how to work with the system;
- Programmers' and technical people's skills in working with the software will be lost;
- All customers will have to incur the cost of dealing with a completely different environment;
- The internal and external customers need to be retrained for the new system.

The cost of disposing an information system is so high that the company may end up working with an obsolete system for a long time. This is a hidden cost in that it is rarely taken into account in the optimistic cost estimation of developing a new system. Furthermore, the formal treatment of disposal costs provides incentive to introduce obsolescence-prevention factors in design decisions, hence postponing obsolescence and making disposal less costly.

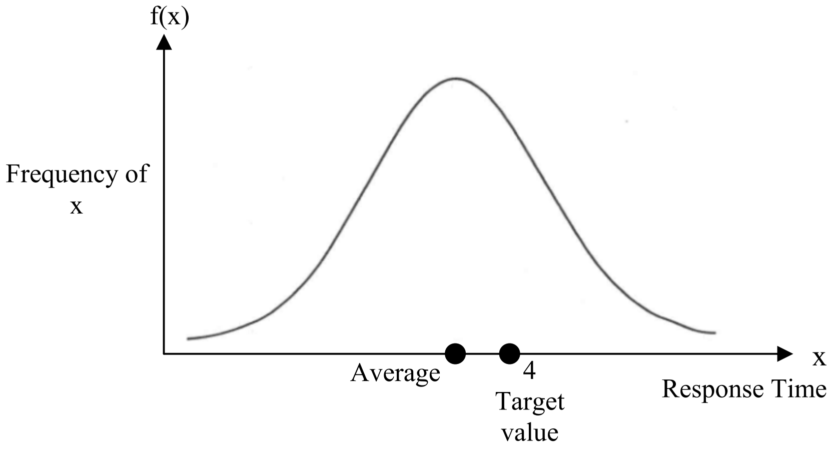
3. The Taguchi Measure of Quality

Taguchi defines quality as the minimum deviation from target value. For example, assume that one of the quality measures of an information system is its response time, with the target value of four seconds. Obviously the response time is not always at four; many factors impact the response time at any one try. It is a random variable with the possible distributions shown in Figure 1. Since there are many factors impacting the response time, one may safely assume that the variable has a normal (bell shape) distribution.

In looking at the distribution of the response time, the average of the distribution may fall well above the target value (Figure 1-a). This difference means a significant quality cost.



(a) Target value is below the distribution average



(b) Target value is around the distribution average

Fig. 1: Target value and the distribution of the information system's response time

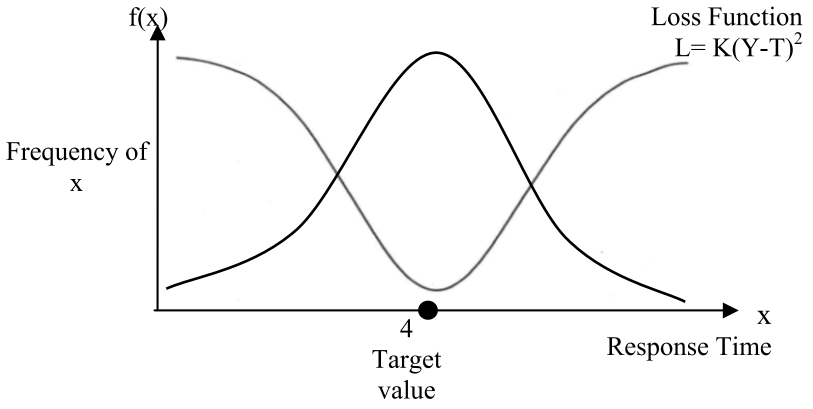
In Figure 1-b, the target value is close to the average of the distribution that reflects lower quality cost. But closeness of the target and average values is not a sufficient measure. There may exist substantial costs because of a large variability in the response time [Card, David, 1992].

In determining the target value for the response time, it is a given fact that the system will not constantly be on the target value. Therefore, we establish the acceptable upper and lower limits for the response time.

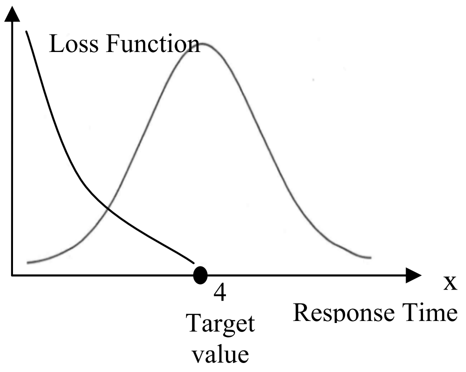
The traditional way of interpreting values of the system's performance measure is that if the distribution of the measure falls outside the upper and lower limits, then the system is out of its allowable range. If a system is within its determined range, then the system's performance is satisfactory. Taguchi disputes this conclusion. He argues that any deviation from the target value is a loss.

Based on the desirability of reducing the variability, Taguchi draws the loss function as the loss due to deviation from target value, shown in Figure 2.

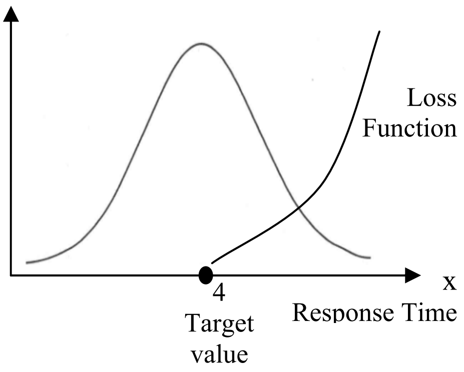
For cases where both positive and negative deviations from the target are considered undesirable, the loss function has a quadratic form, as shown in Figure 2-a. If only positive (or negative) deviation is unwanted, then the loss function consists of half of the quadratic function, as shown in Figure 2-b for the case when the negative deviation from the target is undesirable and 2-c when the positive deviation from the target is undesirable.



a) Loss function when both positive and negative deviations are undesirable



b) Undesirable negative deviation



c) Undesirable positive deviation

Fig. 2: The Quality Loss Function

Taguchi argues that the major focus of the design should be in reduction of the variability of performance. For example, if the reduction in the response time translates into more variability in response time, then the system's performance goes down. This is especially crucial for information systems, where large fluctuations in a system with a good performance record could be considered worse than a fair but consistent and predictable performance. Compare accessing a system that gives a response in one second or in ten minutes as opposed to another system that the responds in five seconds 99 percent of the time. Most people would prefer the system with five-second response because of its predictability.

Thus, in designing a system, it is not enough to consider the average performance parameters of the system; the important factor in the design is the variability of the system performance.

According to Taguchi methods, the loss function of deviating from the target can be measured by:

$$L = C \cdot D^2$$

where C is the cost of unit deviation and D is the amount of deviation from the target.

Furthermore, quality is not what we measure and control after the system is implemented. Instead, quality should be designed into the system. This is a concept that is not commonly addressed in the design of information systems, partly due to the lack of a formal method. The next section discusses using the Taguchi methods in designing systems that have low variability and are robust with respect to uncontrollable environmental factors.

4. Robust Design in Taguchi Methods

The main focus of Taguchi methods is in design that is robust with respect to internal and external disturbances. The disturbances to an information system include:

- Errors by customers who do not follow the instructions for using the system
- Data errors entering the system
- Errors from employees who do not follow the required procedures
- Failure of the network on which the information system operates
- The decision to alter one component of the system, such as the manager of a node in a distributed database deciding to change its database software or add a data item to its database
- The data not being updated on time
- Hardware failure

According to the Taguchi methods, the design of the system should minimize or preferably eliminate the possibility of such disturbances. Taguchi suggests the measure of *signal-to-noise ratio* as a measure for guiding the design [Taguchi, 1990]. Signal is what the system is expected to produce, and noise is an undesirable output. A robust design attempts to maximize signal-to-noise ratio.

For example, the user interface could be designed such that it would not allow the customer to make a mistake—or, if he or she does make a mistake, corrective actions are immediately provided to the customer. Similarly, the intelligent data-entry interface stops the data-entry clerk from entering the wrong data. If the data is not updated on time, the system could send a warning signal to the system manager and in sensitive cases, could even shut down the system. In the case of network problems, such as a failure in a link, one can design alternative routes in case one link in the network shuts down. In the case of a hardware failure, such as a node or a server becoming unoperational, a back-up node or server may take over the function without interrupting the system's operation.

Information systems are among the best candidates for implementing Taguchi's robustness idea. A robust information-system design has an intelligent component for self-control and self-management. Although various fragments of self-management, such as intelligent user interface and intelligent database-entry systems are recognized as necessary modules of a system, there are no common and well-accepted design principles for a self-management unit as a major and necessary component of information systems [Bagchi, 1993].

The purpose of the self-management unit is to increase the system robustness with respect to environmental and external factors, and to self-evaluate the system performance and reliability well before the customers become dissatisfied with the system.

Measuring Signal-to-Noise Ratio of Design Factors. The signal-to-noise ratio is a powerful metric for designing various components of the system. If we could identify the value of this metric for each system module, the design decisions would become straightforward and simple. The field of information systems has not begun to address the measurement issues related to quality metrics. Therefore, for the time being we need to use the signal-to-noise ratio as a metaphor to compare various design options. Although we cannot measure the signal-to-noise ratio, we can identify design components that increase its value.

The following design principles could contribute to the enhancement of the signal-to-noise ratio:

- reduce the interaction of components;
- encapsulate each component;
- create interface standards;
- include safeguards against faulty components;
- do not implement a system that barely meets the set standards or goals;
- use lowered performance variability as a metric for robustness;
- incorporate a self-manager module into the design of information systems;

Encapsulating each component of the information systems and reducing the interactions among them helps reduce the chance of a faulty component contaminating the performance of others. Therefore, the signal-to-noise ratio of each component will be determined by the design of that component, rather than the value of the signal-to-noise ratios of others.

The standard interaction makes it possible to change a component without affecting the design of others. Therefore, if the signal-to-noise ratio of a component is low, one can redesign that component without affecting the signal-to-noise ratio of other components.

Including safeguards for a faulty component allows the system to switch to a backup component if the signal-to-noise ratio of a component falls below a predetermined level, hence maintaining the overall performance of the system.

The design should not barely meet a minimum acceptable level of signal-to-noise ratio because a small variation in the performance of one component pushes the system performance below the acceptable level. Among the major goals of the system should be the reduction of variability in the signal-to-noise ratio and staying on the target level all the time. Similarly, designing a self-manager module into the system increases the signal-to-noise ratio of the system.

Orthogonal Arrays. Selecting the best and most expensive alternative for each component of the system may not guarantee the highest and least variable signal-to-noise ratio for the system. For example, a system with almost unlimited capacity for data may make the retrieval and access to data least efficient and highly variable. An elaborate self-manager unit that sends warnings and signals for the slightest deviation may make the task of data entry very cumbersome to the point that the data-entry personnel and system manager may ignore all warnings of the system and find a way to bypass the system checks. This defeats the purpose for which die self-manager unit is designed. Furthermore, since the system is designed to rely on the self-manager for error checks, there may not be any manual checks. Therefore, bypassing the self-manager unit may lead to a drastic reduction in the signal-to-noise ratio of the system. Thus, it is essential to measure the impact of various design options on the overall signal-to-noise ratio of the system.

One way to measure the impact of various design options is to create prototypes of the system in which various combinations of design elements are tested. Alternatively, one can simulate the system and test the combination of various design options. In order to reduce the number of required prototypes at the same time, to test all major design options in prototypes or the simulation of the system, one can use Taguchi's orthogonal arrays.

The use of orthogonal arrays in the design of information systems would be for the design of complex systems in which either simulation or prototyping of the system is in order. In such cases, one can use orthogonal arrays to find the optimal design diat maximizes the signal-to-noise ratio of the system.

5. Conclusions

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.

REFERENCES

1. Bagchi, T. P. (1993) - *Taguchi methods Explained: Practical step to Robust Design*, Prentice-Hall;

2. Card, David and Glass, Robert L. (1990) - Measuring Software Design Quality, *Prentice Hall*, Englewood Cliffs, NJ;

3. Kan, S. H. (1995) - *Metrics and Models in Software*, Massachusetts: Addison-Quality Engineering, Wesley;

4. Roy, Ranjit. (1990) - *A Primer on the Taguchi Methods*, Van Nostrand Reinhold, New York, NY;

5. Taguchi, Genichi and Clausing, Don. (1990) - "Robust Quality," *Harvard Business Review*, January-February, Vol. 68, No. 1;

6. Zahedi, Fatemeh (1995) - *Quality Information Systems*, Thomson Publishing, USA.