Part VII

Software Testing and Application

21 Application of Taguchi Methods to Software System Testing

21.1. Introduction

System testing is a relatively new area within the Taguchi methods. Systems under testing can be hardware and software or pure software: for example, printer operation, copy machine operation, automotive engine control system, automotive transmission control, automotive antilock brake (ABS) module, computer operating systems such as Windows XP, computer software such as Word and Excel, airline reservation systems, ATMs, and ticket vending machines. The objective of testing is to detect all failures before releasing the design. It is ideal to detect all failures and to fix them before design release. Obviously, you would rather not have customers detect failures in your system.

21.2. Difficulty in System Testing

The difficulty is that there are millions of combinations of customer usage commands. Let's consider a simple system such as printing, for which usage can include the following selections (the number of choices are shown in parentheses):

- ❏ Tray (5)
- ❏ Print range (5)
- ❏ Pages per sheet (6)
- ❏ Duplex options (4)
- ❏ Medium (10)
- ❏ Collate choice (2)

Taguchi's Quality Engineering Handbook. Genichi Taguchi, Subir Chowdhury and Yuin Wu

- Orientation (2)
- Scale to paper (6)

The total number of combinations with these basic parameters alone is

$$5 \times 5 \times 6 \times 4 \times 10 \times 2 \times 2 \times 6 = 144{,}000 \text{ combinations}$$

In addition to these parameters, there are yes/no choices for 11 print options. That would be $2^{11} = 2048$ combinations. Then the total combination will be

$$(2048)(144{,}000) = 294{,}912{,}000 \text{ combinations}$$

Suppose that testing takes 1 minute per combination and assuming that testing can be automated and can be performed 24 hours a day and 365 days per year, it will take 561 years to complete all combinations. If it takes 1 second per combination, it will take 9.35 years.

In addition to these parameters, there are several more user selections with variable input. One such command is the brightness setting, where user input is between −100 and 100 in increments of 1 unit. In other words, this variable has 201 choices; and there are several parameters like this. In other words, there exist literally billions of user input combinations, and it is impossible to complete them all.

21.3. Typical Testing

Testing people analyze the functionality of software and user operations. Then they select common operations that would go under testing. The number of combinations may vary depending on the complexity of the system. But it is typical that such testing requires several weeks to several months to complete, and such testing is not perfect in terms of area coverage or detection rate. Taguchi methods of software testing utilize an orthogonal array that will improve both area coverage and detection rate.

❒ Example

The procedure for system testing using Taguchi methods is explained below using a copy machine operation as an example.

Step 1: Design the test array. Determine the factors and levels. Factors are operation variables and levels are alternatives for operations. Assign them to an orthogonal array.

Factors and Levels. When an operation such as "paper tray selection" has, say, six options, you may just use three options out of six for the first iteration. In the next iteration you can take other levels. Another choice is to use all six options (a six-level factor). It is not difficult to modify an orthogonal array to accommodate many multilevel factors.

If the user gets to input a continuous number, such as "enlargement/shrinkage setting," where a choice can be any percentage between 25 and 256%, you can select some numbers. For example:

A: enlargement
$A_1 = 33\%$ $A_2 = 100\%$ $A_3 = 166\%$

A: enlargement
$A_1 = 25\%$ $A_2 = 66\%$ $A_3 = 100\%$ $A_4 = 166\%$ $A_5 = 216\%$

Basically, you can take as many levels as you like. The idea is to select levels to cover the operation range as much as possible. As you can see in the design of experiment section of this book, you may choose an orthogonal array to cover as many factors and as many levels as you like.

Orthogonal Array. Recall the notation of an orthogonal array (Figure 21.1). Following is a list of some standard orthogonal arrays:

$L_8(2^7)$, $L_{16}(2^{15})$, $L_{32}(2^{31})$, $L_{64}(2^{63})$

$L_9(3^4)$, $L_{27}(3^{13})$, $L_{81}(3^{40})$

$L_{12}(2^{11})$, $L_{18}(2^1 \times 3^7)$, $L_{36}(2^{11} \times 3^{12})$, $L_{54}(2^1 \times 3^{25})$

$L_{32}(2^1 \times 4^9)$, $L_{62}(4^{21})$, $L_{25}(5^6)$, $L_{50}(2^1 \times 5^{11})$

Below is a list of some orthogonal arrays that you can generate from a standard array.

$L_8(4^1 \times 2^4)$

$L_{16}(4^1 \times 2^{12})$ $L_{16}(4^2 \times 2^9)$, $L_{16}(4^3 \times 2^6)$, $L_{16}(4^4 \times 2^3)$, $L_{16}(4^5)$, $L_{16}(8^1 \times 2^8)$

$L_{32}(4^1 \times 2^{28})$, $L_{32}(4^4 \times 2^{19})$, $L_{32}(4^8 \times 2^7)$, $L_{32}(8^1 \times 2^{24})$, $L_{32}(8^1 \times 4^6 \times 2^{12})$

$L_{64}(4^1 \times 2^{60})$, $L_{64}(4^{12} \times 2^{27})$, $L_{64}(8^8 \times 2^7)$, $L_{64}(8^4 \times 4^5 \times 2^{20})$, $L_{64}(8^9)$

L_{27} $(9^1 \times 3^9)$

L_{81} $(9^1 \times 3^{32})$, $L_{81}(9^6 \times 3^{16})$, $L_{81}(27^1 \times 3^{27})$, $L_{81}(9^{10})$

$L_{18}(6^1 \times 3^6)$

$L_{36}(6^1 \times 3^{13} \times 2^2)$

$L_{54}(6^1 \times 3^{24})$

For example, $L_{81}(9^6 \times 3^{16})$ can handle up to six nine-level factors and 16 three-level factors in orthogonal 81 runs.

In the design of experiments section of the book a technique called *dummy treatment* is described. This technique allows you to assign a factor that has fewer

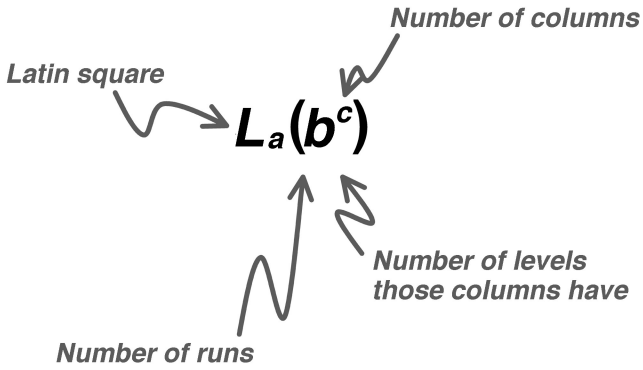

Figure 21.1
Orthogonal array notation

levels than those of a column. Using the dummy treatment technique, you can assign:

- ❑ two-level factor to a three-level column
- ❑ two-level factor to a four-level column
- ❑ two-level factor to a five-level column etc.
- ❑ three-level factor to a four-level column
- ❑ three-level factor to a five-level column
- ❑ three-level factor to a six-level column etc.
- ❑ four-level factor to a five-level column
- ❑ four-level factor to a six-level column
- ❑ four-level factor to an eight-level column etc.
- ❑ seven-level factor to a nine-level column etc.

For example, using dummy treatment, $L_{81}(9^6 \times 3^{16})$ can handle:

- ❑ Five two-level factors
- ❑ Ten three-level factors
- ❑ One five-level factor
- ❑ Three seven-level factors
- ❑ Two eight-level factors

By having all these orthogonal arrays and dummy treatment techniques, you can assign just about any number of factors and levels. A Japanese company, Fuji Xerox, uses an L_{128} array for system testing. An L_{128} can handle just about any situation as long as the number of factors is on the order of tens, up to 80 or 90.

To return to our example of copy machine operation system, factors and levels are shown in Table 21.1. They can be assigned to an L_{18} $(2^1 \times 3^7)$.

Table 21.1
Factors and levels

	Factor	Level 1	Level 2	Level 3
A	Staple	No	Yes	
B	Side	2 to 1	1 to 2	2 to 2
C	No. copies	3	20	50
D	No. pages	2	20	50
E	Paper tray	Tray 6	Tray 5 (LS)	Tray 3 (OHP)
F	Darkness	Normal	Light	Dark
G	Enlarge	100%	78%	128%
H	Execution	From PC	At machine	Memory

Step 2: Conduct system testing. Table 21.2 shows the L_{18} test array. Eighteen tests are conducted according to each row of the L_{18}. The response will simply be a 0–1 response where

$$y = \begin{cases} 0 & \text{if no problem} \\ 1 & \text{if problem} \end{cases}$$

It is okay to have some infeasible factor-level combinations. For example, when OHP is selected, users should not be able to perform copying "2 to 1 side" or "2 to 2 sides." In that case, the system should provide a proper response. If the system provides the proper output, the data is 0; the data is 1 otherwise.

Step 3: Construct two-way response tables showing the failure rate. Once all tests are run and 0–1 data corrected, a two-way response table is constructed for every two-factor combination. An $A \times B$ table, $A \times C$ table, and so on, up to an $G \times H$ table is then constructed. The entry of each combination is the sum of the 1's of the responses. The data are then converted into the percentage of failure (Table 21.3). For instance, since there are nine combinations of B_iC_j (for $i = 1, 2, 3$ and $j = 1, 2, 3$), there are two runs of B_2C_3 in an L_{18}. The result is two under B_2C_3, indicating 100% failure for B_2C_3. Similarly, A_1B_2 results in two failures in three runs, indicating a 66.7% failure rate.

In general, for a combination A_iB_j to generate 100% failure, the total number of failures must equal the size of the array ÷ the number of combinations A_iB_j. For this example, for A_1B_1 to become 100% failure, the total must be 3 (18/6 = 3).

Step 4: Investigate 100% combinations. By observing two-way tables, 100% failure occurred for B_2C_3, B_2F_3, B_2G_2, C_3G_2, and H_1F_3. Now we need to investigate those combinations for how failures occurred, and then fix the problem.

Table 21.2
L_{18} array

	A_1	A_2																	
B_1	0	0																	
B_2	2	1																	
B_3	0	0																	
			B_1	B_2	B_3														
C_1	0	0	0	0	0														
C_2	1	0	0	1	0														
C_3	1	1	0	**2**	0														
						C_1	C_2	C_3											
D_1	0	1	0	1	0	0	0	1											
D_2	1	0	0	1	0	0	1	0											
D_3	1	0	0	1	0	0	0	1											
									D_1	D_2	D_3								
E_1	1	0	0	1	0	0	0	1	0	0	1								
E_2	0	1	0	1	0	0	0	1	1	0	0								
E_3	1	0	0	1	0	0	1	0	0	1	0								
												E_1	E_2	E_3					
F_1	1	0	0	1	0	0	0	1	0	0	1	1	0	0					
F_2	0	0	0	0	0	0	0	0	0	0	0	0	0	0					
F_3	1	1	0	2	0	0	1	1	1	1	0	0	1	1					
															F_1	F_2	F_3		
G_1	1	0	0	1	0	0	1	0	0	1	0	0	0	1	0	0	1		
G_2	1	1	0	**2**	0	0	0	**2**	1	0	1	1	1	0	1	0	1		
G_3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
																		G_1	G_2	G_3
H_1	1	1	0	2	0	0	1	1	1	1	0	0	1	1	0	0	**2**	1	1	0
H_2	1	0	0	1	0	0	0	1	0	0	1	1	0	0	1	0	0	0	1	0
H_3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 21.2
(*Continued*)

	A 1	*B* 2	*C* 3	*D* 4	*E* 5	*F* 6	*G* 7	*H* 8	1 = Problem
1	1	1	1	1	1	1	1	1	0
2	1	1	2	2	2	2	2	2	0
3	1	1	3	3	3	3	3	3	0
4	1	2	1	1	2	2	3	3	0
5	1	2	2	2	3	3	1	1	1
6	1	2	3	3	1	1	2	2	1
7	1	3	1	2	1	3	2	3	0
8	1	3	2	3	2	1	3	1	0
9	1	3	3	1	3	2	1	2	0
10	2	1	1	3	3	2	2	1	0
11	2	1	2	1	1	3	3	2	0
12	2	1	3	2	2	1	1	3	0
13	2	2	1	2	3	1	3	2	0
14	2	2	2	3	1	2	1	3	0
15	2	2	3	1	2	3	2	1	1
16	2	3	1	3	2	3	1	2	0
17	2	3	2	1	3	1	2	3	0
18	2	3	3	2	1	2	3	1	0

Table 21.3
Percentage of failure

	A_1	A_2																	
B_1	0	0																	
B_2	67	33																	
B_3	0	0																	
			B_1	B_2	B_3														
C_1	0	0	0	0	0														
C_2	33	0	0	50	0														
C_3	33	33	0	**100**	0														
						C_1	C_2	C_3											
D_1	0	33	0	50	0	0	0	50											
D_2	33	0	0	50	0	0	50	0											
D_3	33	0	0	50	0	0	0	50											
									D_1	D_2	D_3								
E_1	33	0	0	50	0	0	0	50	0	0	50								
E_2	0	33	0	50	0	0	0	50	50	0	0								
E_3	33	0	0	50	0	0	50	0	0	50	0								
												E_1	E_2	E_3					
F_1	33	0	0	50	0	0	0	50	0	0	50	50	0	0					
F_2	0	0	0	0	0	0	0	0	0	0	0	0	0	0					
F_3	33	33	0	**100**	0	0	50	50	50	50	0	0	50	50					
															F_1	F_2	F_3		
G_1	33	0	0	50	0	0	50	0	0	50	0	0	0	50	0	0	50		
G_2	33	33	0	**100**	0	0	0	**100**	50	0	50	50	50	0	50	0	50		
G_3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
																		G_1	G_2	G_3
H_1	33	33	0	**100**	0	0	50	50	50	50	0	0	50	50	0	0	**100**	50	50	0
H_2	33	0	0	50	0	0	0	50	0	0	50	50	0	0	50	0	0	0	50	0
H_3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

It is important to recognize that this system test method will lead us to where problems may reside, but it does not pinpoint the problem or provide solutions to solve problems.

Xerox, Seiko Epson, and ITT Defense Electronics are three companies that presented applications for this method in public conferences in the late 1990s. All three companies report that they achieved 400% improvement in both area coverage and detection rate. For more case studies, see Section 2 in this book.