# CASE 87

# Streamlining of Debugging Software Using an Orthogonal Array

**Abstract:** In the development process of software, debugging before shipping a product is one of the most important and time-consuming processes. Where bugs are found by users after shipment, not only the software per se but also its company's reputation will be damaged. On the other hand, thanks to the widely spreading Internet technology, even if software contains bugs, it is now easy to distribute bug-fix software to users in the market through the Internet. Possibly because of this trend, the issue of whether there are software bugs for a personal computer seems to have become of less interest lately. However, it is still difficult to correct bugs after shipping in the case of software installed in hardware. Thus, we need to establish a method of removing as many bugs as possible within a limited period before shipment. In this study, a debugging method using an orthogonal array is illustrated.

## 1. Introduction

No effective debugging method has been developed to date. Since a debugging procedure or the range depends completely on each debugging engineer, a similar product might be of different quality in some cases. In addition, because it is so labor intensive, debugging is costly. Thus, a more efficient method of finding bugs could lead to significant cost reduction.

In the context addressed above, Genichi Taguchi has proposed an effective debugging method in two articles titled "Evaluation of objective function for signal factor" [1, 2] and "Evaluation of signal factor and functionality for software" [3]. In reference to these, we detail the results of our experiment on debugging using an orthogonal array.

Since this methodology is a technique for evaluating an objective function for multiple signals, it is regarded as related not to quality engineering but rather to the design of experiments, in a sense.

The method that we propose suggests that we allocate items selected by users (signal factors) to an $L_{18}$ or $L_{36}$ orthogonal array, run software in accordance with a combination of signal factors in each row of the orthogonal array, and judge, using 0 and 1, whether or not an output is normal [4]. Subsequently, using the output obtained, we calculated a variance or interaction to identify which combination of factors was most likely to cause bugs. Through these steps we can find almost all bugs caused by each combination of factors, and we only need to check for the remaining bugs based on single factor changes.

## 2. Debugging Experiment Using Orthogonal Array

Using the β version of our company's software, we performed an experiment grounded on an orthog-

onal array. This is because the β version contains numerous bugs, whose existence has been recognized. Therefore, the effectiveness of this experiment can easily be confirmed. However, since bugs detected cannot be corrected, we cannot check whether or not the trend regarding the number of bugs is decreasing.

As signal factors we selected eight items that can frequently be set up by users, allocating them to an $L_{18}$ orthogonal array. When a signal factor has four or more levels—for example, continuous values ranging from 0 to 100—we selected 0, 50, and 100. When dealing with a factor that can be selected, such as patterns 1 to 5, three of the levels that are most commonly used by users were selected. Once we assigned these factors to an orthogonal array, we noticed that there were quite a few two-level factors. In this case we allocated a dummy level to level 3. (Because of our company's confidentiality rule, we have left out the details about signal factors and levels.)

For the output, we used a rule of normal = 0 and abnormal = 1, based on whether the result was what we wanted. In some cases, "no output" is the right output. Therefore, normal or abnormal is determined by referring to the specifications. Signal factors and levels are shown in Table 1.

**Table 1**
Signal factors and levels

| | 1 | 2 | 3 |
|---|---|---|---|
| $A$ | $A_1$ | $A_2$ | — |
| $B$ | $B_1$ | $B_2$ | $B_3$ |
| $C$ | $C_1$ | $C_2$ | $C_3$ |
| $D$ | $D_1$ | $D_2$ | $D_3$ |
| $E$ | $E_1$ | $E_2$ | $E_2'$ |
| $F$ | $F_1$ | $F_2$ | $F_1'$ |
| $G$ | $G_1$ | $G_2$ | $G_1'$ |
| $H$ | $H_1$ | $H_2$ | $H_3$ |

**Table 2**
$L_{18}$ orthogonal array and output

| No. | *A* | *B* | *C* | *D* | *E* | *F* | *G* | *H* | Ouput |
|---|---|---|---|---|---|---|---|---|---|
| 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 | 2′ | 1′ | 1′ | 3 | 1 |
| 4 | 1 | 2 | 1 | 1 | 2 | 2 | 1′ | 3 | 1 |
| 5 | 1 | 2 | 2 | 2 | 2′ | 1′ | 1 | 1 | 0 |
| 6 | 1 | 2 | 3 | 3 | 1 | 1 | 2 | 2 | 0 |
| 7 | 1 | 3 | 1 | 2 | 1 | 1′ | 2 | 3 | 0 |
| 8 | 1 | 3 | 2 | 3 | 2 | 1 | 1′ | 1 | 0 |
| 9 | 1 | 3 | 3 | 1 | 2′ | 2 | 1 | 2 | 0 |
| 10 | 2 | 1 | 1 | 3 | 2′ | 2 | 2 | 1 | 0 |
| 11 | 2 | 1 | 2 | 1 | 1 | 1′ | 1′ | 2 | 0 |
| 12 | 2 | 1 | 3 | 2 | 2 | 1 | 1 | 3 | 1 |
| 13 | 2 | 2 | 1 | 2 | 2′ | 1 | 1′ | 2 | 0 |
| 14 | 2 | 2 | 2 | 3 | 1 | 2 | 1 | 3 | 1 |
| 15 | 2 | 2 | 3 | 1 | 2 | 1′ | 2 | 1 | 0 |
| 16 | 2 | 3 | 1 | 3 | 2 | 1′ | 1 | 2 | 0 |
| 17 | 2 | 3 | 2 | 1 | 2′ | 1 | 2 | 3 | 0 |
| 18 | 2 | 3 | 3 | 2 | 1 | 2 | 1′ | 1 | 0 |

## 3. Analysis of Bugs

From the results of Table 2, we created approximate two-way tables for all combinations. The upper left part of Table 3 shows the number of each combination of $A$ and $B$: $A_1B_1$, $A_1B_2$, $A_1B_3$, $A_2B_1$, $A_2B_2$, and $A_2B_3$. Similarly, we created a whole table for all combinations. A part where many bugs occur on one side of this table was regarded as a location with bugs.

Looking at the overall result, we can see that bugs occur at $H_3$. After investigation it was found that bugs do not occur in the on-factor test of $H$, but occur by its combination with $G_1$ (= $G_1'$, the same level because of the dummy treatment used) and $B_1$ or $B_2$. Since $B_3$ is a factor level whose selection blocks us from choosing (or annuls) factor levels of H and has interactions among signal factors,

**Table 3**
Binary table created from $L_{18}$ orthogonal array

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

it was considered as the reason that this result was obtained.

Now the calculated variance and interaction were as follows.

Variation between $A$ and $B$ combinations:

$$S_{AB} = \frac{1^2 + 1^2 + 0^2 + 1^2 + 1^2 + 0^2}{3} - \frac{4^2}{18} = 0.44 \qquad (f = 5) \tag{1}$$

Variation of $A$:

$$S_A = \frac{2^2 + 2^2}{9} - \frac{4^2}{18} = 0.00 \qquad (f = 1) \tag{2}$$

Variation of $B$:

$$S_B = \frac{2^2 + 2^2 + 0^2}{6} - \frac{4^2}{18} = 0.44 \qquad (f = 2) \tag{3}$$

A summary of all the main effects is shown in Table 4.

Variation of $A$, $B$ interaction:

$$\begin{aligned} S_{A\times B} &= S_{AB} - S_A - S_B \\ &= 0.44 - 0.00 - 0.44 \\ &= 0.00 \qquad (f = 2) \end{aligned} \tag{4}$$

**Table 4**
Main effect

| Factor | Main Effect[a] |
|---|---|
| *A* | 0.0 |
| *B* | **0.44** |
| *C* | 0.11 |
| *D* | 0.11 |
| *E* | 0.03 |
| *F* | 0.03 |
| *G* | **0.44** |
| *H* | **1.77** |

[a] Boldface numbers indicate significant factorial effect.

As the next step, we divided the combinational effect, $S_{AB}$, and interaction effect, $S_{A\times B}$, by each corresponding degree of freedom:

$$\text{combinational effect} = \frac{S_{AB}}{5} = 0.09 \qquad (5)$$

$$\text{interaction effect} = \frac{S_{A\times B}}{2} = 0.00 \qquad (6)$$

Now, since these results are computed from our approximate two-way tables, if the occurrence of bugs is infrequent, as in this example, we should consider such results as a clue for debugging. When there are more bugs or a large-scale orthogonal array is used, we need to use these values for finding bug locations.

## 4. Bug Identification

Finally, we succeeded in finding bugs by taking advantage of each combination of factors (Table 5). As below, using the method as described, the bugs can be found from an observation of specific combinations. Following are the differences between our current debugging process and the method using an orthogonal array:

1. Efficiency of finding bugs
   a. *Current process.* What can be found through numerous tests are mainly independent bugs. To find bugs caused by a combination of factors, we need to perform many repeated tests.
   b. *Orthogonal array.* Through a small number of experiments, we can find independent bugs and bugs generated by a combination of factors. However, for a multiple-level factor, we need to conduct one-factor tests later.
2. Combination of signal factors
   a. *Current process.* We tend to check only where the bug may exist and unconsciously neglect the combinations that users probably do not use.
   b. *Orthogonal array.* This method is regarded as systematic. Through nonsubjective combinations that do not include debug engineers' presuppositions, well-balanced and broadband checkup can be performed.
3. Labor required
   a. *Current process.* After preparing a several-dozen-page check sheet, we have to investigate all of its checkpoints.
   b. *Orthogonal array.* The only task that we need to do is to determine signal factors and levels. Each combination is generated automatically. The number of checkups required is much smaller, considering the number of signal factors.
4. Location of bugs
   a. *Current process.* Since we need to change only a single parameter for each test, we can easily notice whether or not changed items or parameters involve bugs.
   b. *Orthogonal array.* Locations of bugs are identified by looking at the numbers after the analysis.
5. Judgment of bugs or normal outputs
   a. *Current process.* We can easily judge whether a certain output is normal or abnormal only by looking at one factor changed for the test.
   b. *Orthogonal array.* Since we need to check the validity for all signal factors for each

**Table 5**
Combinational and interaction effects

| Factor | Combination | Interaction |
|---|---|---|
| *AB* | 0.09 | 0.00 |
| *AC* | 0.09 | 0.17 |
| *AD* | 0.09 | 0.17 |
| *AE* | 0.09 | 0.25 |
| *AF* | 0.04 | 0.00 |
| *AG* | 0.15 | 0.00 |
| *AH* | 0.36 | 0.00 |
| *BC* | 0.26 | 0.39 |
| *BD* | 0.14 | 0.14 |
| *BE* | 0.17 | 0.19 |
| *BF* | 0.42 | 0.78 |
| *BG* | 0.22 | 0.11 |
| *BH* | 0.39 | 0.22 |
| *CD* | 0.14 | 0.22 |
| *CE* | 0.17 | 0.36 |
| *CF* | 0.22 | 0.44 |
| *CG* | 0.12 | 0.03 |
| *CH* | 0.26 | 0.06 |
| *DE* | 0.07 | 0.11 |
| *DF* | 0.12 | 0.19 |
| *DG* | 0.12 | 0.03 |
| *DH* | 0.26 | 0.11 |
| *EF* | 0.12 | 0.22 |
| *EG* | 0.16 | 0.01 |
| *EH* | 0.23 | 0.01 |
| *FG* | 0.20 | 0.06 |
| *FH* | 0.42 | 0.11 |
| *GH* | 0.62 | 0.44 |

output, it is considered cumbersome in some cases.

6. When there are combinational interactions among signal factors
   a. *Current process*. Nothing in particular.
   b. *Orthogonal array*. We cannot perform an experiment

following combinations determined in an orthogonal array.

Although several problems remain before we can conduct actual tests, we believe that through use of our method, the debugging process can be streamlined. In addition, since this method can be employed relatively easily by users, they can assess newly developed software in terms of bugs. In fact, as a result of applying this method to software developed by outside companies, we have found a certain number of bugs. From now on, we will attempt to incorporate this method into our software development process.

## References

1. Genichi Taguchi, 1999. Evaluation of objective function for signal factor (1). *Standardization and Quality Control*, Vol. 52, No. 3, pp. 62–68.
2. Genichi Taguchi, 1999. Evaluation of objective function for signal factor (2). *Standardization and Quality Control*, Vol. 52, No. 4, pp. 97–103.
3. Genichi Taguchi, 1999. Evaluation of signal factor and functionality for software. *Standardization and Quality Control*, Vol. 52, No. 6, pp. 68–74.
4. Kei Takada, Masaru Uchikawa, Kazuhiro Kajimoto, and Jun-ichi Deguchi, 2000. Efficient debugging of a software using an orthogonal array. *Quality Engineering*, Vol. 8, No. 1, pp. 60–69.

*This case study is contributed by Kei Takada, Masaru Uchikawa, Kazuhiro Kajimoto, and Jun-ichi Deguchi.*