CASE 84

Evaluation of Capability and Error in Programming

Abstract: In assuring software quality, there are few evaluation methods with established grounds. Previously, we have been using a trial-and-error process. One of the reasons is that unlike the development of hardware that machines produce, that of software involves many human beings who are regarded to have a relatively large number of uncertain factors. Therefore, the Sensuous Value Measurement Committee was founded in the Sensory Test Study Group of the Japanese Union of Scientists and Engineers, which has been dealing with the application of quality engineering to the foregoing issues. This research, as part of the achievement, focuses on applying quality engineering to a production line, including human beings with many uncertain factors.

1. Introduction

Because of diversity in production lines of software, it is quite difficult to perform all evaluations at the same time. Figure 1 summarizes the flow of a software production line. We focused on software production that can be relatively easily evaluated. Figure 2 shows the details of software production. By application of the standard SN ratio, we can obtain a good result [4]. On the basis of the experiment, delving more deeply into this issue, we attempted to evaluate the relationship between capability in programming (coding) and errors.

2. SN Ratio for Error Occurrence in Software Production

We created software from specifications. That is, correctness of software in a coding process is our focus. First, we handed specifications and a flowchart to a testee and asked him or her to write a program. When the testee had completed a program that was satisfactory to him or her (or whose framework is completed), he or she saved the program (data 1). Next, compiling this with a compiler for the first time, the testee debugged it by himself or herself. After this program was checked by an examiner, a program with almost no bugs (data 2) was obtained as a final output. Then, by comparing data 1 and 2, we recognized a difference of code between them as an error:

Line number in program:	1	2	3	⋯	N
Representation of correct data:	y_1	y_2	y_3	⋯	y_n

$y = 1$ for a correct line and 0 for an incorrect line.

The analysis procedure (calculation of the SN ratio) is as follows [2]. We compute the fraction of the number of correct codes in data 1 to that of lines in data 2, n:

$$p = \frac{y_1 + y_2 + \cdots + y_n}{n} \tag{1}$$

Because y takes up only 0 and 1, the total variation is

$$S_T = y_1^2 + y_2^2 + \cdots + y_n^2 = np \tag{2}$$

According to the variation in equation (1), the effect by the signal factor results in

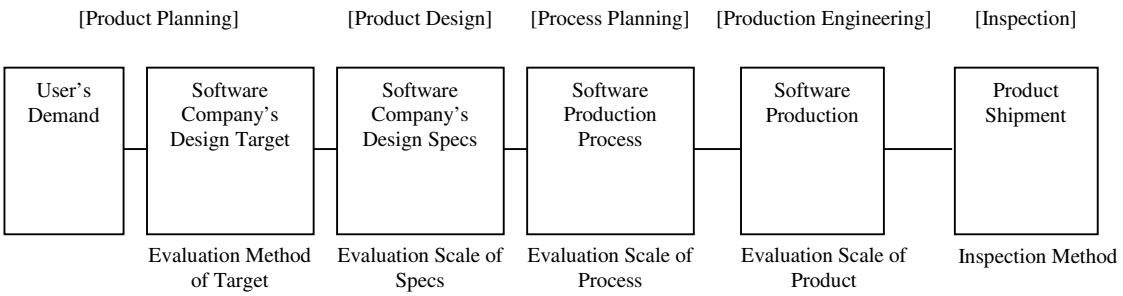

Figure 1
Work flow of software production

$$S_p = \frac{p^2}{\text{total of squares of coefficients}}$$

$$= \frac{[(y_1 + y_2 + \cdots + y_n/n]^2}{(1/n)^2 + (1/n)^2 + \cdots + (1/n)^2}$$

$$= \frac{(y_1 + y_2 + \cdots + y_n)^2}{n} = np^2 = S_m \qquad (3)$$

Then the error variation

$$S_e = S_T - S_\beta = np - np^2 = np\,(1 - p) \qquad (4)$$

An SN ratio for 0/1 data is expressed by a ratio of signal factor variation to error variation. To obtain additivity of measurement in the analysis, we expressed the SN ratio as a decibel value:

$$\eta = 10 \log \frac{S_p}{S_e} = 10 \log \frac{np^2}{np(1 - p)}$$

$$= 10 \log \frac{p}{1 - p}$$

$$= -10 \log \left(\frac{1}{p} - 1\right) \qquad \text{dB} \qquad (5)$$

3. Evaluation Method of Type-by-Type Error with SN Ratio

As a next step, we devised a method of evaluating errors of various types [1]. First, all errors were classified into four categories (Table 1). *Mistake* is an error that can be corrected by a testee if others advise him or her. *Unknown* is an error caused by a testee's ignorance of coding due to insufficient experience in programming or the C language. That is, in this case, even if others give advice, the testee cannot correct the error.

We allocated these error types to an L_8 orthogonal array (Table 2) We defined as level 1 a case of using an error type, and no use as level 2, and at the same time, evaluated an error's significance. The right side of Table 2 shows the number of errors.

As below, we show an example of the calculation for row 2 of the L_8 orthogonal array. We set each number of errors for A', B', C', and D' to 2, 15, 3, and 7, respectively, and the number of lines to 86. Since we use level 1 for the analysis, the only nec-

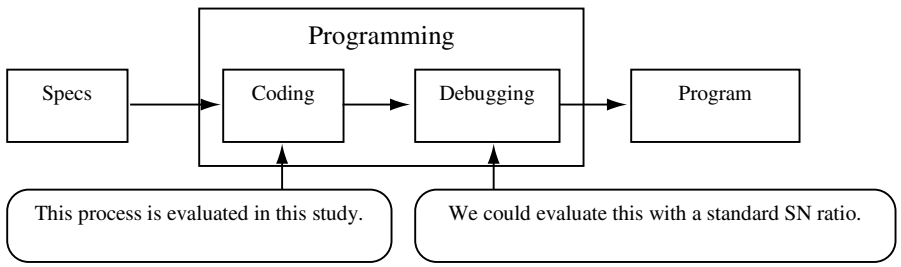

Figure 2
Details of software production

Table 1
Error types and levels

Factor	Level 1	Level 2
Logical error		
A': Mistake	Use	No use
B': Unknown	Use	No use
Syntax error		
C': Mistake	Use	No use
D': Unknown	Use	No use

essary data are A', B', and C; D' is not used for the analysis of row 2 because its number of levels is two.

The sum of errors for experiment 1 in the L_8 orthogonal array is

$$A' + B'' + C' = 2 + 15 + 3 = 20$$

Therefore, the number of lines where correct codes are written is

$$(\text{total number of lines} - \text{number of errors for } D') - \text{number of errors} = 59$$

Then the proportion of correct codes, p, is computed as

$$p = \frac{59}{86 - 7} = 0.747 \tag{6}$$

In this case, because we disregard D' itself, it should be excluded from the total number of lines.

Total variation:

$$S_T = np = (79)(0.747) = 79.0 \tag{7}$$

Effect by a signal factor:

$$S_p = np^2 = (79)(0.747^2) = 32.91 \tag{8}$$

Error variation:

$$S_e = S_T - S_p = 79.0 - 32.91 = 46.09 \tag{9}$$

Thus,

$$\eta = 10 \log \frac{S_p}{S_e} = -10 \log \left(\frac{1}{p} - 1\right) = 4.70 \text{ dB} \tag{10}$$

Using these results we can evaluate the capability of software production for each type of error.

4. Capability Evaluation of Software Production

With an L_{12} orthogonal array, we studied an evaluation method of capability of software production. As factors regarding an instructional method and evaluation, 10 types of factors were selected (Table 3). The C language is used and the program consists of about 100 steps. As testees, six engineering students who have had 10 hours of training in the C language were chosen. Next, we tested these testees according to the levels allocated in the L_{12} orthogonal array.

Table 2
Error types and number of errors assigned to L_8

No.	A'	B'	C'	D'	e	e	e	A'	B'	C'	D'	Total
1	1	1	1	1	1	1	1	2	15	3	7	27
2	1	1	1	2	2	2	2	2	15	3		20
3	1	2	2	1	1	2	2	2			7	9
4	1	2	2	2	2	1	1	2				2
5	2	1	2	1	2	1	2		15		7	22
6	2	1	2	2	1	2	1		15			15
7	2	2	1	1	2	2	1			3	7	10
8	2	2	1	2	1	1	2			3		3

Table 3
Factors and levels

Factor	Level 1	Level 2
A: type of program	Business	Mathematical
B: instructional method	Lecture	Software
C: logical technique	F.C.	PAD
D: period before deadline (days)	3	2
E: coding guideline	Yes	No
F: ability in Japanese language	High	Normal
G: ability for office work	High	Normal
H: number of functions	Small	Large
I: intellectual ability	High	Normal
J: mathematical ability	High	Normal

Because we were dealing with six testees, dividing the L_{12} orthogonal array into the upper half (1 to 6) and the lower half (7 to 12), we first performed the upper-half experiment using the six testees. Then we performed experiments on the lower half conducted on the same testees. That is, each testee needs to write two programs (business and mathematical programs). In this case, although a learning effect was considered to influence them, it hardly has an effect because this level of programming had been completed a few times prior to the training.

The following is an explanation of the factors selected.

- *A: type of program.* Business and mathematical programs were prepared for the experiment.
- *B: instructional method.* A testee may have been trained in the C language by a teacher or may have self-studied it using instructional software.
- *C: logical technique.* A schematic of a program's flow was given along with specifications.
- *D: deadline.* A testee was required to hand in a program within two or three days after the specifications were given.
- *E: coding guideline.* This material states ways of thinking in programming.
- *F: ability in Japanese language.* The ability to understand Japanese words and relevant concepts and to utilize them effectively is judged.
- *G: ability for office work.* The ability to comprehend words, documents, or bills accurately and in detail is judged.
- *H: number of functions.* As long as the difficulty level of each program was not changed, there are two types of specifications, minimal and redundant, in which a testee writes a program.
- *I: intellectual ability.* The ability to understand explanations, instructions, or principles is judged.
- *J: mathematical ability.* The ability to solve mathematical problems is judged.

For human abilities (*F*, *G*, *I*, and *J*), we examined the testees using the general job aptitude test issued by the Labor Ministry. After checking each testee's ability, we classified them into high- and low-ability groups.

5. Experimental Results

We computed interactions between the factors selected and error types. More specifically, an L_8 or-

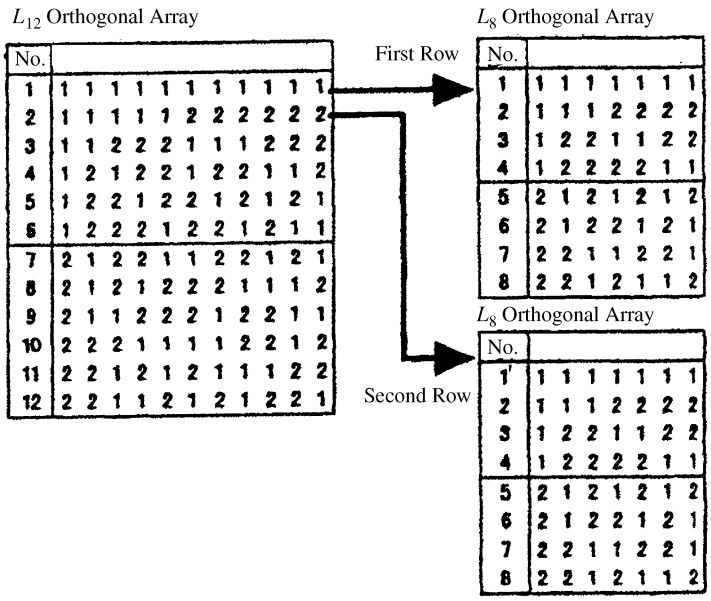

Figure 3
Layout of direct-product experiment using an L_{12} orthogonal array to evaluate the capability of software production and an L_8 orthogonal array to evaluate error types

thogonal array (outer array for error types) was allocated to each row in an L_{12} orthogonal array (inner array for combinations of all factors), as shown in Figure 3. Table 4 shows the data obtained from the experiment, and Table 5 shows the result computed through an analysis of variance with respect to the SN ratio. Boldfaced entries in Table 5 indicate a large factor effect.

Next we inquired into the main effect of the factor shown in Figure 4. The factor with the greatest main effect is the number of functions (H). A small number of functions leads to a small number of

Table 4
Number of errors obtained from experiment

	L_{12}											
L_8	1	2	3	4	5	6	7	8	9	10	11	12
A'	2	7	3	0	6	6	4	4	6	4	4	5
B'	15	51	7	75	50	13	40	12	160	47	2	35
C'	3	2	57	8	2	0	2	0	6	1	10	5
D'	7	12	0	3	2	0	15	4	12	12	2	0
Total number of lines	86	96	266	113	146	60	104	84	300	94	103	116

Table 5
ANOVA with respect to SN ratio based on capability of software production and error types[a] (dB)

Factor	*f*	*S*	Factor	*f*	*S*	Factor	*f*	*S*
A	1	22.9	$A \times A'$	1	11.0	$A \times C'$	1	12.8
B	1	41.8	$B \times A'$	1	1.5	$B \times C'$	1	0.0
C	**1**	**178.5**	$C \times A'$	1	90.5	$C \times C'$	1	109.5
D	1	31.1	$D \times A'$	1	2.4	$D \times C'$	1	70.3
E	1	77.2	$E \times A'$	1	85.2	$E \times C'$	1	79.7
F	1	106.8	$\boldsymbol{F} \times A'$	**1**	**141.3**	$\boldsymbol{F} \times \boldsymbol{C'}$	**1**	**218.4**
G	1	17.5	$G \times A'$	1	47.5	$\boldsymbol{G} \times \boldsymbol{C}$	**1**	**146.7**
H	**1**	**929.8**	$H \times A'$	1	123.1	$H \times C'$	1	11.4
I	1	33.7	$I \times A'$	1	50.1	$I \times C$	1	21.4
J	1	22.9	$J \times A'$	1	27.5	$J \times C'$	1	79.2
A'	1	66.4	$A \times B'$	1	29.6	$A \times D'$	1	7.6
$\boldsymbol{B'}$	**1**	**2408.7**	$B \times B'$	1	44.4	$B \times D'$	1	8.1
C'	1	6.5	$C \times B'$	1	23.5	$C \times D'$	1	0.1
$\boldsymbol{D'}$	**1**	**236.7**	$D \times B'$	1	1.5	$D \times D'$	1	9.1
e	41	938.4	$E \times B'$	1	48.1	$E \times D'$	1	0.1
(*e*)	**86**	**(2466.8)**	$F \times B'$	1	21.1	$F \times D'$	1	0.9
ST	95	6980.9	$\boldsymbol{G \times B'}$	**1**	**147.1**	$G \times D'$	1	9.7
			$H \times B'$	1	77.0	$H \times D'$	1	14.7
			$I \times B'$	1	0.6	$I \times D'$	1	32.2
			$J \times B'$	1	39.3	$J \times D'$	1	17.8

[a] Parentheses indicate pooled error.
$V_e = 938.4/41 = 22.9$. $\boldsymbol{(V_e)} = (2466.8)/86 = (28.7)$.

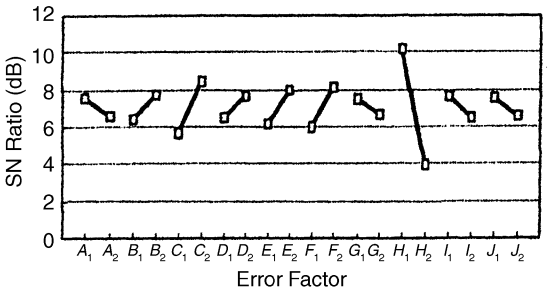

Figure 4
Response graphs of the SN ratio for capability of software production

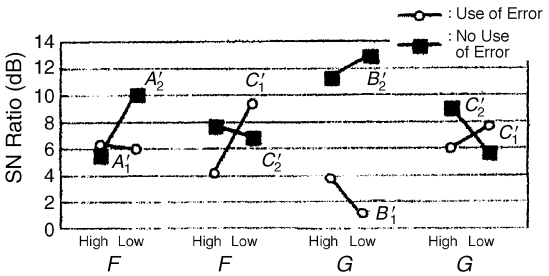

Figure 6
Interaction between SN ratios for capability of software production and SN ratios for error type

errors and subsequently, to a high SN ratio. In addition, logical technique (C) and ability in the Japanese language (F) have a large effect. Since factors other than these have a small effect, good reproducibility cannot be expected. On the other hand, since we observed no significant effect for intellectual ability, even in the experiment on error-finding ability, this factor's effect in programming was considered trivial.

In the response graphs for error types shown in Figure 5, level 1 represents error used for analysis, and level 2, when it was not used. Since the SN ratio when not using error is high, all the errors in this study should be included in an analysis. Unknown error in logical error (B') turned out to be particularly important.

Furthermore, according to the analysis of variance table in Figure 6, while some factors generate significant interactions with A', B', and C', interactions with D' had only trivial interactions with other factors, but D' per se was relatively large compared with other error types. By combining large-interaction errors into one error and making a different classification, we are able to categorize errors into one with clear interactions and one without interactions in the next experiment.

It was observed that an interaction occurs more frequently in the factors regarding ability in the Japanese language and ability for office work. Considering that a similar result was obtained in a confirmatory experiment on error finding [3], we supposed that the reason was a problem relating to the ability test itself.

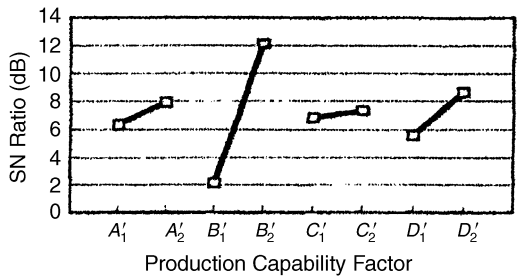

Figure 5
Response graphs of the SN ratio for error type

6. Conclusions

As a result of classifying capability of software production by each error type and evaluating it with an SN ratio for the number of errors, we concluded that our new approach was highly likely to evaluate the true capability of software production. We are currently performing a confirmatory experiment on its reliability.

Since students have been used as testees in this study, we cannot necessarily say that our results would hold true for actual programmers. Therefore, we need to test this further in the future.

The second problem is that a commercial ability test has been used to quantify each programmer's ability in our experiment. Whether this ability test is itself reproducible is questionable. An alternative method of ability evaluation needs to be developed.

On the other hand, while we have classified errors into four groups, we should prepare a different error classification method in accordance with an

experimental objective. For example, if we wish to check only main effects, we need to classify the error without the occurrence of interaction. On the contrary, if interactions between error types and each factor are more focused, a more mechanical classification that tends to trigger the occurrence of interactions is necessary.

References

1. Kei Takada, Muneo Takahashi, Narushi Yamanouchi, and Hiroshi Yano, 1997. The study of evaluation of capability and errors in programming. *Quality Engineering,* Vol. 5, No. 6, pp. 38–44.
2. Genichi Taguchi and Seiso Konishi, 1988. *Signal-to-Noise Ratio for Quality Evaluation,* Vol. 3 in Tokyo: *Quality Engineering Series.* Japanese Standards Association, and ASI Press.
3. Norihiko Ishitani, Muneo Takahashi, Narushi Yamanouchi, and Hiroshi Yano, 1996. A study of evaluation of capability of programmers considering time changes. *Proceedings of the 26th Sensory Test Symposium,* Japanese Union of Scientists and Engineers.
4. Norihiko Ishitani, Muneo Takahashi, Kei Takada, Narushi Yamanouchi, and Hiroshi Yano, 1996. A basic study on the evaluation of software error detecting capability. *Quality Engineering,* Vol. 4, No. 3, pp. 45–53.

This case study is contributed by Kei Takada, Muneo Takahashi, Narushi Yamanouchi, and Hiroshi Yano.