World Academy of Science, Engineering and Technology
International Journal of Computer and Systems Engineering
Vol:1, No:1, 2007

# A Taguchi Approach to Investigate Impact of Factors for Reusability of Software Components

Parvinder S. Sandhu, Pavel Blecharz, and Hardeep Singh



***Abstract*—**Quantitative Investigation of impact of the factors' contribution towards measuring the reusability of software components could be helpful in evaluating the quality of developed or developing reusable software components and in identification of reusable component from existing legacy systems; that can save cost of developing the software from scratch. But the issue of the relative significance of contributing factors has remained relatively unexplored. In this paper, we have use the Taguchi's approach in analyzing the significance of different structural attributes or factors in deciding the reusability level of a particular component. The results obtained shows that the complexity is the most important factor in deciding the better Reusability of a function oriented Software. In case of Object Oriented Software, Coupling and Complexity collectively play significant role in high reusability.

***Keywords*—**Taguchi Approach, Reusability, Software Components, Structural Attributes.

## I. Introduction

THE demand for new software applications is currently increasing at the exponential rate, as is the cost to develop them. The numbers of qualified and experienced professionals required for this extra work are not increasing commensurably [1]. Software professionals have recognized reuse as a powerful means of potentially overcoming the above said software crisis [2], [3] and it promises significant improvements in software productivity and quality [4], [5]. There are two approaches for reuse of code: develop the reusable code from scratch or identify and extract the reusable code from already developed code. The organization that has experience in developing software, but not yet used the software reuse concept, there exists extra cost to develop the reusable components from scratch to build and strengthen their reusable software reservoir [4]. The cost of developing the software from scratch can be saved by identifying and extracting the reusable components from already developed and existing software systems or legacy systems [6]. But the issue of how to identify reusable components from existing systems has remained relatively unexplored as the contribution of impact of factors responsible in deciding the reusability has not studied quantitatively. In both the cases, whether we are developing software from scratch or reusing code from already developed projects, there is a need of evaluating the quality of the potentially reusable software.

Tracz observed that for programmers to reuse software they must first find it useful [7]. Experimental results confirm that prediction of reusability is possible but it involves more than the set of metrics that are being used [8]. The contribution of metrics to the overall objective of the software quality is understood and recognized [9]-[11]. But how these metrics collectively determine reusability of a software component is still at its naïve stage and the contribution of each factor towards the reusability of the software components is still not investigated quantitatively. The organization of the paper is as follows:

The second section discusses about Taguchi Approach .in general. Third section given details of the methodology followed. Third and fourth sections describe implementation and results obtained. In Last section conclusion is made.

## II. Taguhi Approach

Design of Experiments (DOE) using Taguchi approach is a standardized form of experimental design technique (referred as classical DOE). DOE is an experimental strategy in which effects of multiple factors are studied simultaneously by running tests at various levels of the factors. But what is a factor and its level? Factor (notation A, B…) is variable (also called parameter) that have direct influence of the output (quality characteristic). Levels (notation $A_1$) are the descriptions that define the condition of the factor held while running the experiments. Quality characteristic (notation Y) is then yardstick of output performance and we distinguish 3 types: B (bigger is better), S (smaller is better) and N (nominal the best).

To study factor influence, we must carry out experiments at least with two levels of the factors. When it is necessary to test more factors (say 5, 10, 15 or more), number of all combinations (full factorial design) is too big. 15 factors at 2 levels require $2^{15}$ trials. To minimize number of trials, Taguchi developed a set of special tables (called orthogonal arrays, OA). Each orthogonal array involves only fraction of all possible combinations. For example, 15 factors at 2 levels

The Manuscript was submitted for review on Sept. 27, 2006.

Parvinder S. Sandhu is Assistant Professor with Computer Science & Engineering Department, Guru Nanak Dev Engineering College, Ludhiana (Punjab)-141006, India (phone: +91-98555-32004; fax: +91161-2490339; e-mail: parvinder.sandhu@gmail.com).

Dr. Pavel Blecharz, Ph.D.(Systems Engineering),Technical University of Ostrava, Sokolská třída 33, Ostrava 1, 701 21, Office : A515 (phone : +420 - 59699 2233, e-mail: pavel.blecharz@vsb.cz).

Dr. Hardeep Singh is Professor and Head with Computer Science & Engineering Department, Guru Nanak Dev University, Amritsar (Punjab), India.

require only 16 runs. Notation of orthogonal array is "L-16". Letter "L" shows mathematical background (latin square), while number "16" means number of trials.

Experiments are designed in accordance with appropriate orthogonal array. Experiment results are then recorded into orthogonal array (separate column in the right size). An analysis then utilizes results and OA together.

The analysis has standard steps:

- Average and main effect of factors,/interactions,
- ANOVA (Analysis of Variance),
- Optimum, Y at optimum condition.

More detailed description of Taguchi method is possible to find for example in [12].

## III. Methodology Followed

A two-tier approach is proposed for finding the significance of the factors to evaluate the software component's reusability by analyzing structural properties of the component.

### A. Structural Analysis

A framework of metrics is proposed for structural analysis of procedure or function-oriented [13] and object-oriented [14] query components separately. In [14], Selby has pointed the major desirable attributes in reusable software components and following suits of metrics are able to target those the essential attributes, so we tried analyze, refine and use following metrics of to explore different structural dimensions of an OO component.

The proposed metrics for Function Oriented Paradigm are as follows:

*1) Cyclometric Complexity Using Mc Cabe's Measure*: If the complexity is low then reuse of component will not repay the cost. Otherwise high value of complexity indicates poor quality, high development cost, low readability, poor testability and prone to errors i.e. high rate of failure. Hence, the value of Cyclometric Complexity of a software component should be in between upper and lower bounds as an contribution towards reusability [15] [16].

*2) Regularity Metric:* Regularity is the ratio of estimated length to the actual length [6]. High value of Regularity indicates the high readability, low modification cost and non-redundancy of the component implementation [17]. Hence, there should be some minimum level of Regularity of the component to indicate the reusability of that component.

3) *Halstead Software Science Indicator:* If the volume is high means that software component needs more maintenance cost, correctness cost and modification cost [17]. On the other hand, less volume increases the extraction cost, identification cost from the repository and packaging cost of the component. So the volume of the reusable component should be in between the two extremes.

*4) Reuse Frequency Metric:* "Reuse frequency" is the measure of function usefulness of a component [6]. Hence, there should be some minimum value of "Reuse Frequency" to make software component really reusable.

*5) Coupling Metric:* As coupling increases, there is decrease in understandability and maintainability, so there should be some maximum value of the coupling associated with a software component, beyond which the component becomes non-reusable [11] [15].

The metrics for Object Oriented Paradigm are as follows:

*1) Tuned Weighted methods per class (TWMC):* According to Weighted methods per class (WMC) metric of CK metric suit, if a Class C, has n methods and c1, c2 …cn be the complexity of the methods, then WMC(C)= c1 + c2 +… + cn. Mc Cabe's complexity metric is chosen for calculating the complexity values of the methods of a class, the value is normalized so that nominal complexity for a method takes on a value of 1.0 [18][ 19].

We have used "tuned WMC" (TWMC) measure for class complexity by restricting the WMC value in between 0 and 1 with help of sigmoidal function as shown in (1).

$$f(x,a,c) = \frac{1}{1+e^{-a(x-c)}} \quad (1)$$

Where a=10 and c=0.5.

*2) Lack of Tuned Depth of inheritance tree (LTDIT):* According to DIT metric Depth of inheritance of a class is "the maximum length from the node to the root of the tree". More is the depth of the inheritance tree greater the reusability of the class corresponding to the root of that tree as the class properties are shared by more derived classes under that class. So there too much depth dilutes the abstraction. So there is a need to set the minimum & maximum DIT value for a class as an contribution towards the reusability [18][ 19].

We have used "lack of tuned degree of inheritance" (LTDIT) measure as input in order to restrict the input value between 0 and 1.

*3) Lack of Tuned Number of Children (LTNOC):* According to NOC metric Number of children (NOC) of a class is the number of immediate sub-classes subordinated to a class in the class hierarchy. So greater is the value of NOC greater will be the reusability of the parent class. Hence there should be some minimum value of NOC for a parent class for its reusability [18][19].

In order to restrict the input value between 0 and 1, we have used "lack of tuned Number of Children" (LTNOC) measure as input.

*4) Lack of Coupling Between Object Classes (CBO):* According to CBO metric "Coupling Between Object Classes" (CBO) for a class is a count of the number of other classes to which it is coupled. Theoretical basis of CBO relates to the notion that an object is coupled to another object if one of them acts on the other, i.e. methods of one use methods or instance variables of another. Here we are restricting the unidirectional use of methods or instance variables of another object by the object of the class whose reusability is to be measured. As Coupling between Object classes increases, reusability decreases and it becomes harder to modify and test the software system. So there is the need to set some maximum value of coupling level for its reusability & thief the value if CBO for a class is beyond that maximum value then the class is said to be non-reusable[17][18].

In order to restrict the input value between 0 and 1, we have used "lack of CBO" (LCBO) measure as input.

*5) Lack of Cohesion in Methods (LCOM):* Consider a Class $C_1$ with *n* methods $M_1$ , $M_2$ ..., $M_n$ . Let $\{I_j\}$ = set of instance variables used by method $M_i$ .There are *n* such sets $\{I_1\},\{I_2\}...$ $\{I_n\}$. Let $P = \{ (I_i, I_j) \mid I_i \cap I_j = \varnothing \}$ and $Q = \{ (I_i, I_j) \mid I_i \cap I_j \neq \varnothing \}$. If all n sets $\{I_1\},\{I_2\}...$ $\{I_n\}$. are $\varnothing$ then let $P = \varnothing$ [4]. Lack of Cohesion in Methods (LCOM) of a class can be defined as:

$$LCOM = |P| - |Q|, \text{ if } |P| > |Q|$$
$$LCOM = 0 \text{ otherwise}$$

The high value of LCOM indicates that the methods in the class are not really related to each other and vice versa means less reusability otherwise low value of LCOM depicts high internal strength of the class which results into high reusability. So there should be some maximum value of LCOM after that class becomes non-reusable [18][19].

We have used "tuned LCOM" (LCOM) measure as input to the neuro-fuzzy inference engine by restricting the LCOM value in between 0 and 1 with help of sigmoidal function as shown in (2).

$$f(x,a,c) = \frac{1}{1+e^{-a(x-c)}} \quad (2)$$

Where a=4 and c=1.5.

Values to the linguistic variables of all inputs are assigned in terms of three linguistic variables "LOW", "MEDIUM" and "HIGH" in the range of 0 to 1 as discussed with the experts in the domain.

Values to the linguistic variables of Reusability are assigned in terms of "how reusable the software module is?" Reusability is assigned six linguistic variables PERFECT, HIGH, MEDIUM, LOW, VERY-LOW and NIL as constants in the range of 0-1.

### *B. Taguchi Analysis*

As there are three levels for each factor, L-27 Orthogonal Array was selected for the experimental design and Taguchi analysis of the data collected from the last stage is performed. The analysis is showed in the form of a software output of Qualitek-4 Software for Automatic Design and Analysis of Taguchi Experiments developed by R.K. Roy. Beyond this, there are some notes and commentary to understand better each software output (tables).

## IV. Experimentation And Results

Structural analysis of 109 function oriented software components and 87 object oriented components is performed. The meta information is extracted and metric values are calculated.

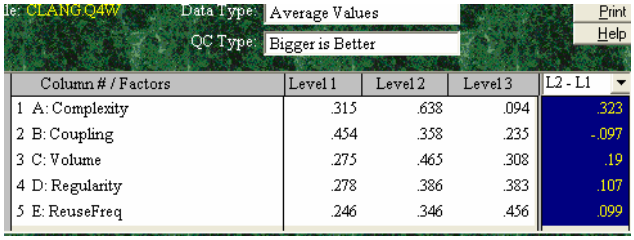
### *A. Analysis of Function based Reusability Data*

The Quality characteristic, Y, reusability is derived with five Factors (inputs), that are designated as:

A: complexity
B: coupling
C: volume,
D: regularity
E: reuse frequency

As there is need of maximization of the Y so "bigger is better" option is most suitable for the analysis. The steps performed in the analysis are as under:

*1) Average and Main effects:* The average effect of the factors is studied and Factor A has the strongest effect (0.323) shown in Fig. 1.




| Column # / Factors | Level 1 | Level 2 | Level 3 | L2 - L1 |
|---|---|---|---|---|
| 1 A: Complexity | .315 | .638 | .094 | .323 |
| 2 B: Coupling | .454 | .358 | .235 | -.097 |
| 3 C: Volume | .275 | .465 | .308 | .19 |
| 4 D: Regularity | .278 | .386 | .383 | .107 |
| 5 E: ReuseFreq | .246 | .346 | .456 | .099 |

Fig. 1 Average effect of factors

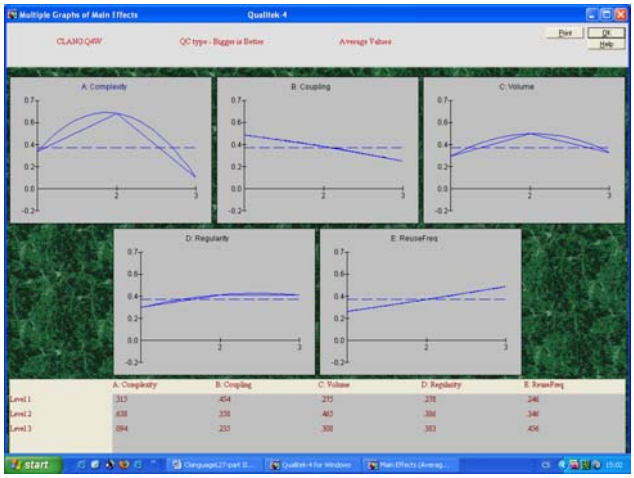



Fig. 2 Graph of Main effects of factors

We can see non linear effect of factors A (Complexity) and C (Volume) in the Main effect analysis of factors as shown in Fig. 2. If we wanted further improvement, follow-up experiment for these 2 factors could find, if level 2 (Medium) is real optimum.

Further, we can make test for presence of interactions as shown in the interaction table shown in Fig. 3. Interactive analysis can ensure qualitative analysis of any pair of factors.

It is clear from the interaction table that the strongest interaction (severity index) is in between factor A (Complexity and factor B (Coupling), then between factor C (Volume) and factor E(Reuse Frequency) and so on.

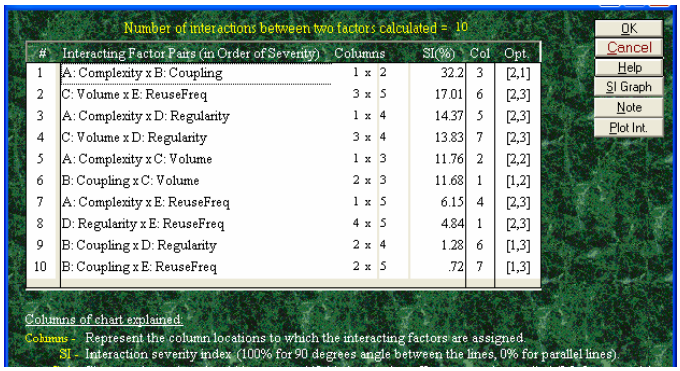



Fig. 3 Interaction table

*2) ANOVA:* ANOVA analysis of the function oriented reusability data is performed as shown in Fig. 4. It is interpreted from the obtained results that factor A (Complexity) is extremely important (key factor, 63.9%). Factors B (Coupling), C (Volume) and E (reuse Frequency) have standard impact.

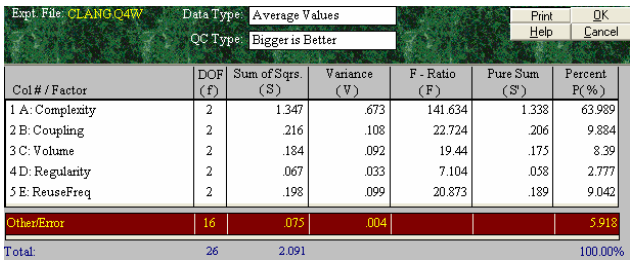



Fig. 4 ANOVA Analysis of Function Based Data

But now the question is - what about factor D? Generally, factors with less than 5% influence we consider as non significant. It means, is it really factorial effect or an error? Should we pool this factor?

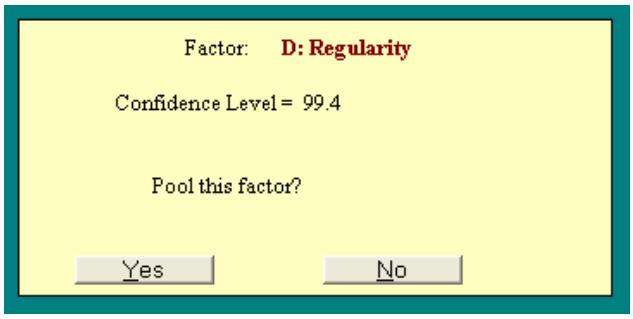



Fig. 5 Test of significance

As shown in Fig. 5, the confidence level is found equal to 99.4. So, test of significance says that factor D should not be pooled. But anyhow, factor D (Regularity) does have very poor effect to output.

*3) Optimum conditions:* In order to find the optimum conditions of factors, "Bigger is Better" analysis is performs as shown in Fig. 6.

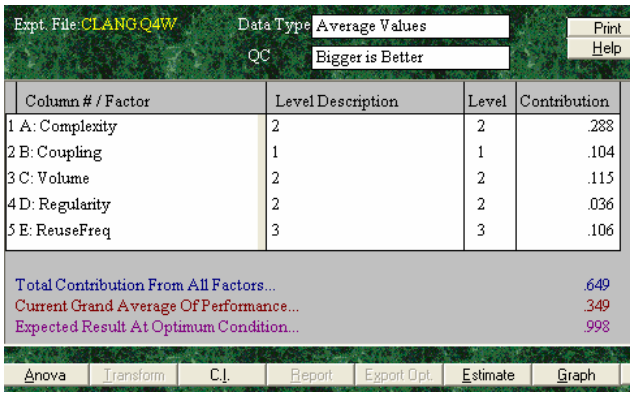



Fig. 6 Bigger- the -Better Analysis

Optimum condition found is: $A_2\ B_1\ C_2\ D_2\ E_3$
At this optimum conditions: $Y_{OPT} = 0.998$.

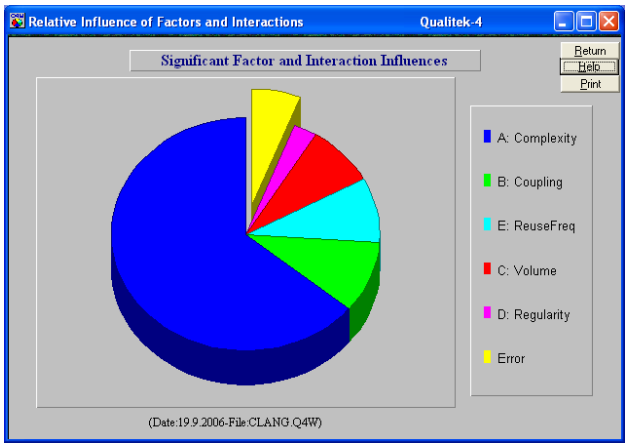



Fig. 7 Relative Influence of factors and interactions

The relative influence of factors and interactions is shown in Fig. 7. Since experimental error is only 6% and it has been reached significant improvement here, there is no need to make additional experiments or look for better combination. Although it is possible to carry out an L-9 experiments with factors A and C and slightly increase optimization effect.

If validation test will confirm the results above, this factor combination (optimum) could be used as a standard.

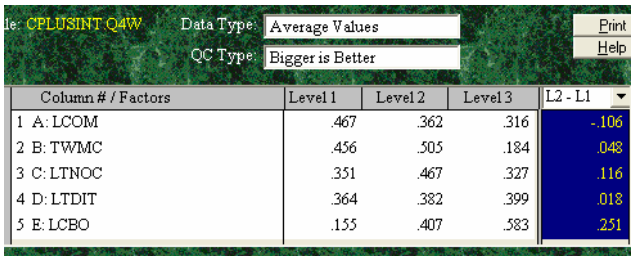
*B. Analysis for Object Oriented Reusability Data*

The Quality characteristic, Y, reusability is derived with five Factors (inputs), that are designated as:

A: LCOM
B: TWCM
C: LTNOC
D: LTDIT
E: LCBO

The three steps of Taguchi analysis are performed as follows:

*1) Average and Main effects:* The average effect of the factors is studied and Factor E has the strongest effect (0.251) shown in Fig. 8.




| Column # / Factors | Level 1 | Level 2 | Level 3 | L2 - L1 |
|---|---|---|---|---|
| 1 A: LCOM | .467 | .362 | .316 | -.106 |
| 2 B: TWMC | .456 | .505 | .184 | .048 |
| 3 C: LTNOC | .351 | .467 | .327 | .116 |
| 4 D: LTDIT | .364 | .382 | .399 | .018 |
| 5 E: LCBO | .155 | .407 | .583 | .251 |

Fig. 8 Average effect of factors of OO based Data

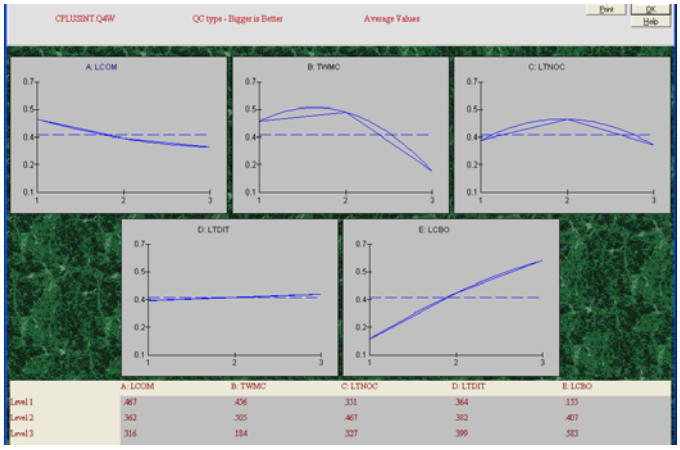


Fig. 9 Graph of Main effects of factors

We can see non-linear effect of factors B (TWMC) and C (LTNOC) in the Main effect analysis of factors as shown in Fig. 9. If we wanted further improvement, follow-up experiment for these 2 factors could find, if level 2 (Medium) is real optimum.

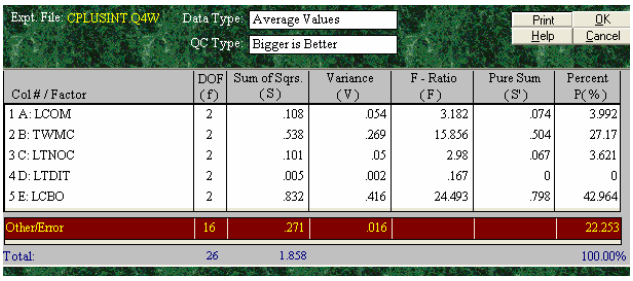
Further, we can make test for presence of interactions as shown in the interaction table shown in Fig. 10. It is clear from the interaction table that the strongest interaction (severity index) is in between factor C (LTNOC) and factor D (LTDIT).

Number of interactions between two factors calculated = 10

| # | Interacting Factor Pairs (in Order of Severity) | Columns | SI(%) | Col | Opt. |
|---|---|---|---|---|---|
| 1 | C: LTNOC x D: LTDIT | 3 x 4 | 78.22 | 7 | [2,2] |
| 2 | A: LCOM x D: LTDIT | 1 x 4 | 42.63 | 5 | [1,2] |
| 3 | B: TWMC x D: LTDIT | 2 x 4 | 38.65 | 6 | [2,2] |
| 4 | A: LCOM x B: TWMC | 1 x 2 | 28.52 | 3 | [1,2] |
| 5 | A: LCOM x C: LTNOC | 1 x 3 | 20.55 | 2 | [1,2] |
| 6 | B: TWMC x C: LTNOC | 2 x 3 | 18.71 | 1 | [2,2] |
| 7 | D: LTDIT x E: LCBO | 4 x 5 | 8.33 | 1 | [3,3] |
| 8 | C: LTNOC x E: LCBO | 3 x 5 | 7.14 | 6 | [2,3] |
| 9 | B: TWMC x E: LCBO | 2 x 5 | 5.52 | 7 | [1,3] |
| 10 | A: LCOM x E: LCBO | 1 x 5 | 3.91 | 4 | [1,3] |

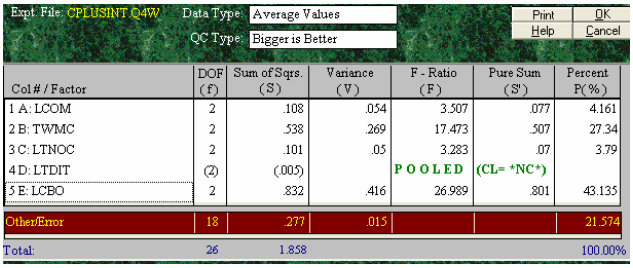
Fig. 10 Interaction Table of Factors

*2) ANOVA:* The ANOVA analysis of Object Oriented Reusability data is performed and the results are shown in Fig. 11. Since factor D has 0% influence, it should be pooled.




| Col # / Factor | DOF (f) | Sum of Sqrs. (S) | Variance (V) | F - Ratio (F) | Pure Sum (S') | Percent P(%) |
|---|---|---|---|---|---|---|
| 1 A: LCOM | 2 | .108 | .054 | 3.182 | .074 | 3.992 |
| 2 B: TWMC | 2 | .538 | .269 | 15.856 | .504 | 27.17 |
| 3 C: LTNOC | 2 | .101 | .05 | 2.98 | .067 | 3.621 |
| 4 D: LTDIT | 2 | .005 | .002 | .167 | 0 | 0 |
| 5 E: LCBO | 2 | .832 | .416 | 24.493 | .798 | 42.964 |
| Other/Error | 16 | .271 | .016 | | | 22.253 |
| Total: | 26 | 1.858 | | | | 100.00% |

Fig. 11 ANOVA Analysis of OO Data




| Col # / Factor | DOF (f) | Sum of Sqrs. (S) | Variance (V) | F - Ratio (F) | Pure Sum (S') | Percent P(%) |
|---|---|---|---|---|---|---|
| 1 A: LCOM | 2 | .108 | .054 | 3.507 | .077 | 4.161 |
| 2 B: TWMC | 2 | .538 | .269 | 17.473 | .507 | 27.34 |
| 3 C: LTNOC | 2 | .101 | .05 | 3.283 | .07 | 3.79 |
| 4 D: LTDIT | (2) | (.005) | | POOLED | (CL= *NC*) | |
| 5 E: LCBO | 2 | .832 | .416 | 26.989 | .801 | 43.135 |
| Other/Error | 18 | .277 | .015 | | | 21.574 |
| Total: | 26 | 1.858 | | | | 100.00% |

Fig. 12 ANOVA after Pooling

When ANOVA after Pooling is performed, the Interpretation of results tells that factor E is extremely important (key factor). Factor B is also important. As shown in Fig. 12, factor B and C have together 70% impact to quality!

Generally, factors with less than 5% influence we consider as non significant. Should we pool factor C and then factor A?

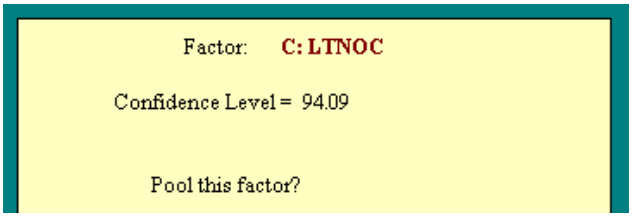



Fig. 13 (a) Pooling of factor C

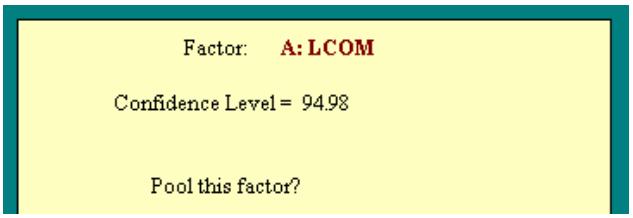



Fig. 13 (b) Pooling of factor A

The confidence level is 94.09 and 94.98 respectively. The choice of 90% confidence level is enough. So test of significance says – factor C and E should not be pooled. But anyhow factors have quite poor effect to output.

*3) Optimum conditions:* In order to find the optimum conditions of factors, "Bigger is Better" analysis is performs as shown in Fig. 14.

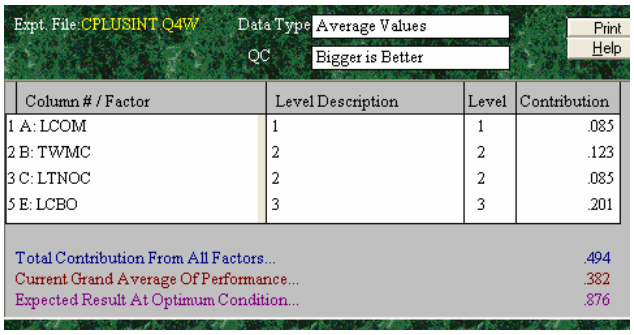



Fig. 14 Bigger- the- Better Analysis for OO Based Data

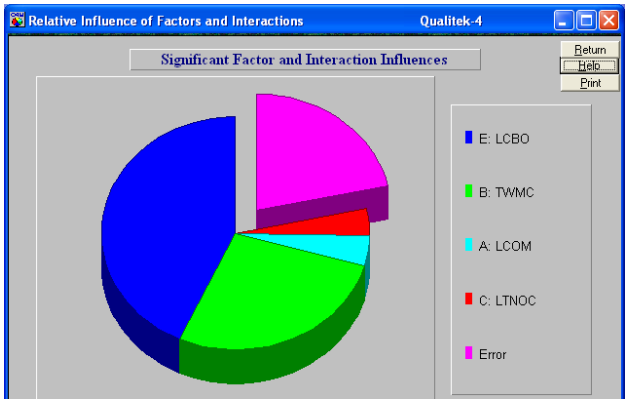



Fig. 15 Relative Influence of factors and interactions

Optimum condition obtained is: Optimum $A_1 B_2 C_2 E_3$

Based on factors effects, D has any chosen level (the most cost effective). But in accordance with interaction table (the strongest interactions CxD, AxD and BxD), level of factor D, based on interactions, is always 2. At this optimum conditions: $Y_{OPT} = 0.876$

Although we do not know if some of these interactions are significant, we can recommend level 2 for factor D (namely if confirmation tests will not be satisfactory).

Optimum: $A_1 B_2 C_2 D_2 E_3$

This combination is in trial number 6. The result is 0.92. The Error is 21%. We can rely on results, but confidence interval for factor effects and Y at optimum will be broader.

## VI. Conclusion

The above studies show well designed experiments. The first study (C language) gives a little bit better results, including small experimental error (5%). The second study about the OO based software reusability, has bigger error (21%), so there could be some other influences not involved in the study. We can rely on results, but confidence interval for factor effects and Y at optimum will be broader.

It is found that the complexity is the most important factor in deciding the better Reusability of a function oriented Software. In case of Object Oriented Software, Coupling and Complexity collectively play significant role in high reusability. The results are stimulating for telling the best condition of all factors in obtaining high reusability index of a software component and identification of the reusable components from the legacy systems.

## References

[1] E. Smith, A. Al-Yasiri, and M. Merabti, "A Multi-Tiered Classification Scheme For Component Retrieval," *Proc. Euromicro Conference*, 1998, 24th Volume 2, 25-27 Aug. 1998, pp. 882 – 889.

[2] V.R. Basili, "Software Development: A Paradigm for the Future," Proc. COMPAC '89, Los Alamitos, Calif.: IEEE CS Press, 1989, pp. 471-485.

[3] B.W. Boehm and R. Ross, "Theory-W Software Project Management: Principles and Examples," *IEEE Trans. Software Eng.*, vol.15, no. 7, 1989, pp. 902.

[4] W. Lim, "Effects of Reuse on Quality, Productivity, and Economics," *IEEE Software*, vol. 11, no. 5, Oct. 1994, pp. 23-30.

[5] H. Mili, F. Mili and A. Mili, "Reusing Software: Issues And Research Directions," *IEEE Transactions on Software Engineering*, Volume 21, Issue 6, June 1995, pp. 528 - 562.

[6] G. Caldiera and V. R. Basili, "Identifying and Qualifying Reusable Software Components", *IEEE Computer*, February 1991, pp. 61-70.

[7] W. Tracz, "A Conceptual Model for Megaprogramming," *SIGSOFT Software Engineering Notes*, Vol. 16, No. 3, July 1991, pp. 36-45.

[8] Stephen R. Schach and X. Yang, "Metrics for targeting candidates for reuse: an experimental approach," ACM, SAC 1995, pp. 379-383.

[9] W. Humphrey, Managing the Software Process, SEI Series in Software Engineering, Addison-Wesley, 1989.

[10] L. Sommerville, Software Engineering, Addision-Wesley, 4th Edition, 1992.

[11] R. S. Pressman, Software Engineering: A Practitioner's Approach, McGraw-Hill Publications, 5th edition, 2005.

[12] R.K. Roy, Design of Experiments Using the Taguchi Approach, John Willey & Sons Inc., 2001.

[13] Parvinder Singh Sandhu and Hardeep Singh, "Automatic Quality Appraisal of Domain-Specific Reusable Software Components", Journal of Electronics & Computer Science, Vol. 8, No. 1, June 2006, pp. 1- 8.

[14] Parvinder Singh Sandhu and Hardeep Singh, "A Reusability Evaluation Model for OO-Based Software Components", *International Journal of Computer Science*, Volume 1, Number 4, 2006, pp. 259-264.

[15] Richard W. Selby, "Enabling Reuse-Based Software Development of Large-Scale Systems", *IEEE Trans. Software Engineering*, VOL. 31, NO. 6, June 2005, pp. 495-510.

[16] T. MaCabe, "A Software Complexity measure," *IEEE Trans. Software Engineering*, vol. SE-2, December 1976, pp. 308-320.

[17] Maurice H. Halstead, Elements of Software Science, Elsevier North-Holland, New York, 1977.

[18] S.R. Chidamber and C.F. Kemerer, "Towards a Metrics Suite for Object Oriented Design," Proc. Conf. Object Oriented Programming Systems, Languages, and Applications (OOPSLA'91), vol. 26, no. 11, pp. 197-211, 1991.

[19] Chidamber, S.R. and Kemerer, C.F., " A Metric Suite for Object Oriented Design", IEEE Trans. Software Engineering , Vol. 20, pp. 476-493,1994.