Application of the Pearson's Method in Model Software Quality Data Analysis

Jie Yang and Deming Zhong* and Zhi Xu
School of Reliability and Systems Engineering, Beihang University, Beijing 100191, China
Dr. Zhong can be reached via E-mail at zhongthomas@qq.com

Abstract. Third-party model software testing involves and generates large amounts of model software quality data. Currently, there is a deficiency in the collection, analysis and application of these data. Due to the characteristics of model software products (e.g., they are complex, uncertain and difficult to measure) as well as the limited and unsystematic understanding of model software quality, model software quality evaluation has always been a challenge in software quality research. Based on the features of third-party model software testing as well as the model software quality data collected during third-party testing, this study focuses on an in-depth analysis of defect data collected during model software testing by using the Pearson product-moment correlation coefficient, presents corresponding conclusions derived from the analysis of model software defect data, presents suggestions for software testing and evaluation organizations to scientifically manage model software quality data and improve software testing efficiency and provides quality control and improvement measures for software developers.

1. Introduction
As the size of software products continues to grow, software quality issues are becoming increasingly prominent, which may cause a series of problems such as delays in the delivery of software products, increases in software development costs, dramatic reductions in the service life of products and decline in user satisfaction. It is against this background that computer scientists and software engineers have sought to more clearly understand the nature of software quality control and attempted to improve software quality using more reasonable and effective methods [1].

Due to the characteristics of model software products (e.g., they are complex, abstract, uncertain and difficult to measure) as well as the limited, unsystematic understanding of model software quality, model software quality evaluation has always been a challenge in software quality research [2] [3]. This study focuses on collecting and analyzing model software quality data using data analysis methods based on the features of third-party model software testing, presents rationalization suggestions for software testing and evaluation organizations to scientifically manage model software quality data and improve software testing efficiency and provides quality control and improvement measures for software developers. This study provides an approach to more objectively evaluate the relationships among software quality data, based on which target-specific measures can be implemented to improve research and development technology and management activity. To date, research in this area is relatively rare in China; thus, the contents of this study are significant.

2. Model Software Data Analysis Process
The model software data analysis process involves several steps [4], as shown in Figure 1.
2.1. Data Selection
Goal-directed data collection is the foundation that ensures effective model software quality data analysis. It is necessary to devise a plan with respect to collected data content as well as data collection channel and method. Based on the specific analysis requirements and the integrity, validity and different classifications of the model software quality data, the collected data are screened and cleansed to extract useful data.

2.2. Data Conversion
A data source often contains various types of data. To meet the requirements of the data analysis technique for model software quality data, it is necessary to use a uniform unit standard for the data, which is used to normalize the model software quality data.

2.3. Data Analysis
During the data analysis step, the processed data are statistically analyzed and evaluated to convert them into information from which to derive conclusions. Generally, various data analysis algorithms are used to analyze the hidden patterns in the data as well as the relationships among the data.

2.4. Evaluation and Suggestions
Based on the analysis results, specific corresponding rationalization suggestions or improvement measures are provided.

Model software quality data analysis is performed according to the aforementioned process. During the data selection and conversion steps, testing data that are easy to manipulate and collect are collected during the third-party testing process, primarily based on the Military Software Quality Metrics (GJB 5236-2004), and suitable model software quality data analysis methods are summarized and used in the data analysis [5]. Defect density is an important index that evaluates model software quality. Therefore, at the data analysis and evaluation and suggestion presentation steps, factors affecting model software quality are primarily analyzed based on defect density [6].

3. Principle of the Pearson Product-Moment Correlation Coefficient
Decision-making problems are often encountered in software engineering. There are myriad factors that can affect the decision-making process, and there are often specific relationships between these factors. It may be impossible to represent a relationship with a specific function. Instead, a quantitative correlation analysis is needed to assist in making the correct decision. The Pearson product-moment correlation coefficient, often denoted by $r$, is an important concept in correlation analysis, which was proposed by British statistician Pearson in the early 1900s to calculate the linear correlation between two variables and is the sample correlation coefficient of a two-dimensional random vector sample $(x, y)$ composed of two examined variables $x$ and $y$ [7] [8].

If $n$ number of observations are made on $(x, y)$ and $n$ number of data pairs $((x_1, y_1),..., (x_n, y_n))$ are obtained, then $r$ is defined as follows:

$$r = \frac{L_{xy}}{\sqrt{L_{xx}L_{yy}}}$$

where

$$L_{xy} = \sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y}), L_{xx} = \sum_{i=1}^{n}(x_i - \bar{x})^2$$

\[1\]

\[2\]
\[ L_{yy} = \sum_{i=1}^{n} (y_i - \bar{y})^2, \bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i, \bar{y} = \frac{1}{n} \sum_{i=1}^{n} y_i \]  

(3)

When \(|r|=1\), there is a completely linear relationship between \(x\) and \(y\). In this case, the smaller the value of \(|r|\), the smaller the degree of linear correlation between \(x\) and \(y\). When \(r=0\), it can be considered that there is no correlation between \(x\) and \(y\) (no linear correlation), but it does not mean that \(x\) and \(y\) are independent of one another because there may be other types of correlation between them. When \(|r|\neq0\) and \(r>0\), it can be considered that there is a positive correlation between \(x\) and \(y\); when \(|r|\neq0\) and \(r<0\), it can be considered that there is a negative correlation between \(x\) and \(y\) [3, 4]. Relationship between \(r\) and correlation is shown in Table 1.

| Correlation             | Negative value | Positive value |
|-------------------------|----------------|----------------|
| Uncorrelated            | -0.09~0.0      | 0.0~0.09       |
| Low correlation         | -0.3~0.1       | 0.1~0.3        |
| Intermediate correlation| -0.5~0.3       | 0.3~0.5        |
| High correlation        | -1.0~0.5       | 0.5~1.0        |

In this study, three model software datasets are selected and analyzed to determine \(r\), based on which software defects are analysed.

4. Correlation Analysis of the Number of Thousands of Lines of Code (KLOC) in a Model Software Program and the Number of Defects Found During Testing (D)

The two variables in the first dataset are the size of each of 112 model software programs, represented by KLOC, and the corresponding D. The correlation between these two variables is analyzed using \(r\). KLOC and D values for the selected model software programs is shown in Table 2.

| No. | 1   | 2   | 3   | 4   | 5   | 6   | 7   | 8   | 9   | 10  | 11  | 12  | 13  | 14  |
|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
|     | KLOC |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0   | 0   | 15.613 | 20.55 | 2.535 | 0.2 | 10 | 3.2 | 3.77 | 3.291 | 17.701 | 2.296 | 5.1 | 1.345 | 21.084 |
| D   | 0   | 61  | 136 | 31 | 16 | 23 | 55 | 32 | 45 | 13 | 52 | 51 | 13 | 117 |
|     | 15  | 16  | 17  | 18  | 19  | 20  | 21  | 22  | 23  | 24  | 25  | 26  | 27  | 28  |
|     | KLOC | 33.939 | 25.373 | 3.756 | 128.835 | 9.916 | 39.854 | 38.29 | 2.791 | 18.431 | 0.201 | 19.924 | 18.728 | 22.994 | 1.797 |
| D   | 21  | 146 | 74  | 137 | 40 | 70 | 225 | 135 | 43 | 0 | 48 | 30 | 83 | 56 |
|     | 29  | 30  | 31  | 32  | 33  | 34  | 35  | 36  | 37  | 38  | 39  | 40  | 41  | 42  |
|     | KLOC | 13.515 | 2.263 | 1 | 0.3 | 0.65 | 107.885 | 94.143 | 3.126 | 47.474 | 4.641 | 6.104 | 0.1 | 6.471 | 87.086 |
| D   | 83  | 41  | 2  | 2 | 6 | 77 | 78 | 34 | 144 | 1 | 36 | 0 | 21 | 308 |
|     | 43  | 44  | 45  | 46  | 47  | 48  | 49  | 50  | 51  | 52  | 53  | 54  | 55  | 56  |
|     | KLOC | 19.206 | 0.659 | 9.49 | 3.382 | 0.673 | 7.928 | 3.261 | 14.089 | 3  | 0.5 | 1.123 | 2.997 | 0.45 | 10.57 |
| D   | 37  | 25  | 76  | 74  | 24  | 28 | 39 | 37 | 8 | 5 | 5 | 12 | 8 | 10 |
|     | 57  | 58  | 59  | 60  | 61  | 62  | 63  | 64  | 65  | 66  | 67  | 68  | 69  | 70  |
|     | KLOC | 3.424 | 0.86 | 13.626 | 13.082 | 3.091 | 61.417 | 10.705 | 2 | 37.735 | 8.521 | 55.366 | 2.889 | 5.491 | 2.787 |
| D   | 3   | 0   | 10  | 0  | 60 | 48 | 46 | 17 | 14 | 18 | 19 | 27 | 31 | 45 |
|     | 71  | 72  | 73  | 74  | 75  | 76  | 77  | 78  | 79  | 80  | 81  | 82  | 83  | 84  |
|     | KLOC | 44.038 | 44.13 | 6.534 | 42.672 | 149.264 | 115.717 | 23.062 | 2.401 | 20.18 | 186.494 | 56.127 | 9.75 | 0.8 | 31.901 |
| D   | 89  | 88  | 34  | 160 | 111 | 227 | 144 | 69 | 70 | 343 | 887 | 10 | 0 | 42 |
|     | 85  | 86  | 87  | 88  | 89  | 90  | 91  | 92  | 93  | 94  | 95  | 96  | 97  | 98  |
|     | KLOC | 1.439 | 21.593 | 9.119 | 41.205 | 0.45 | 5.072 | 9.62 | 5.835 | 2.553 | 8.104 | 7.829 | 41.553 | 14.992 | 25.449 |
Based on the equation provided in Section 2, the following values are obtained:

\[
\begin{align*}
\bar{x} &= 21.118125000000003, \\
\bar{y} &= 74.973214285714292, \\
L_{xy} &= 11.759104343750001e+05, \\
L_{xx} &= 1.128130025902500e+05, \\
L_{yy} &= 1.386614919642856e+06, \\
r &= 0.444768567534013
\end{align*}
\]

The \( r \) between the two variables is calculated to be 0.444768567534013, which is greater than 0.3 but smaller than 0.5, thereby indicating an intermediate correlation. Thus, it can be concluded that for model software, there is a certain correlation between its size (represented by KLOC) and D.

The relationship between D and KLOC is shown in Figure 2.

![Figure 2. D–KLOC relationship.](image)

Next, a linear equation is used to analyze the data, which is generally based on the principle of least squares. Let us set \( y = a + bx \) as the fitted linear equation. The \( a \) and \( b \) that allow the following function to reach the minimum are those being sought.

\[
\phi(a, b) = \sum_{i=1}^{112} (y_i - a - bx_i)^2
\]  

(4)
\( \varphi(a, b) \) is calculated using the method used in mathematical analysis to find extreme values. Next, partial derivatives for \( a \) and \( b \) are found. Then, by setting the partial derivatives to 0, a set of simultaneous equations that \( a \) and \( b \) should satisfy is obtained:

\[
\begin{align*}
\frac{\partial \varphi}{\partial a} &= -2 \sum_{i=1}^{112} (y_i - a - bx_i) = 0 \\
\frac{\partial \varphi}{\partial b} &= -2 \sum_{i=1}^{112} (y_i - a - bx_i)x_i = 0
\end{align*}
\] (5)

By substituting the data from Table 2 into this set of simultaneous equations, \( a \) and \( b \) can be determined: \( a = -42.043512419445442 \) and \( b = 1.559309922934391 \). Thus, the linear equation can also be determined: \( y = 1.559309922934391x - 42.043512419445442 \). This equation suggests that for model software, \( D \) will increase as its size (i.e., KLOC) increases.

The approximate linear relationship between \( D \) and KLOC is shown in Figure 3.

Figure 3. Approximate linear relationship between KLOC and \( D \).

5. Correlation Analysis of the Number of Test Cases (N) Written to Test a Model Software Program and \( D \)

The two variables in the second dataset are the \( N \) and \( D \) for each of the 112 model software programs. The correlation between these two variables is analyzed using \( r \). \( N \) and \( D \) values for the selected 112 model software programs is shown in Table 3.

| No. | 1   | 2   | 3   | 4   | 5   | 6   | 7   | 8   | 9   | 10  | 11  | 12  | 13  | 14  | 15  | 16  | 17  | 18  | 19  | 20  | 21  | 22  | 23  | 24  | 25  | 26  | 27  | 28  |
|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| N   | 56  | 694 | 918 | 123 | 46  | 61  | 72  | 47  | 22  | 251 | 47  | 236 | 137 | 459 |
| D   | 0   | 61  | 136 | 31  | 16  | 23  | 55  | 32  | 45  | 13  | 52  | 51  | 13  | 117 |
Based on the equation provided in Section 2, the following values are obtained:

\[
\begin{align*}
\bar{x} &= 5.580803571428571e+02, \\
\bar{y} &= 74.973214285714292, \\
L_{xy} &= 6.650647241071427e+06, \\
L_{xx} &= 7.040735827678569e+07, \\
L_{yy} &= 1.386614919642856e+06, \\
r &= 0.673095913198346
\end{align*}
\]

The \( r \) between the two variables is calculated to be 0.673095913198346, which is greater than 0.5, thereby indicating a significantly positive correlation. Thus, it can be concluded that for model software, there is a certain correlation between \( D \) and \( N \).

The relationship between \( D \) and \( N \) is shown in Figure 4.

Next, a linear equation is used to analyze the data, which is generally based on the principle of least squares. Let us set \( y = a + bx \) as the fitted linear equation. The \( a \) and \( b \) that allow the following function to reach the minimum are those being sought. The same method used in Section 3.3.1 is used to calculate \( a \) and \( b \).
By substituting the data from Table 3 into this set of simultaneous equations, $a$ and $b$ can be determined: $a = 22.257195983518375$ and $b = 0.094459548033692$. Thus, the linear equation can be determined: $y = 0.094459548033692x + 22.257195983518375$. This equation, together with the one determined in Section 3.3.1, indicates that for model software, $D$ will increase as its size and $N$ increase.

The approximate linear relationship between $D$ and $N$ is shown in Figure 5.
6. Correlation analysis of the number of test cases written to test a model software program per thousand lines of code (NKLOC) and D

The two variables in the third dataset are the NKLOC and D for each of the 112 model software programs. The correlation between the two variables is analyzed using r. NKLOC and D values for the selected 112 model software programs is shown in Table 4.

Based on the equation provided in Section 2, the following values are obtained:

\[ x = 642, \bar{y} = 67.9, L_{xy} = 504220 \]

\[ L_{xx} = 16279928, L_{yy} = 60288, r = 0.509 \]

The r between the two variables is calculated to be 0.052921370 546483, which is smaller than 0.1, thereby indicating no correlation. Thus, it can be concluded that for model software, there is no correlation between D and NKLOC.

The relationship between D and NKLOC is shown in Figure 6.

![Figure 6. Relationship between D and NKLOC.](image)

As demonstrated in Figure 6, D varies significantly when NKLOC is in the range of 0~50, but decreases as NKLOC continues to increase after exceeding 50.

7. Conclusions

In this study, by considering the integrity of information of model software (size, N and D), the quality data of 112 model software programs obtained from testing are selected and analyzed. The analysis results show that for model software, there is a significant correlation (a positive linear relationship) between D and its size, a significant correlation (a positive linear relationship) between D and N but no correlation between D and NKLOC.

These findings indicate that for model software, D will increase as its size and N increase. In addition, D varies significantly when NKLOC is in the range of 0~50, but decreases as NKLOC continues to increase after exceeding 50.

Based on these findings, the following rationalization suggestions are provided:

- During the software testing process, when NKLOC<50, the testing personnel should increase N to increase the probability of finding software defects.
During the software testing process, when NKLOC\geq 50, the testing personnel do not need to increase N to improve software testing efficiency.

In this study, defect data included in model software quality data are analyzed based on the principle of the Pearson product-moment correlation coefficient. Two rationalization suggestions are provided for in-depth analysis of model software quality data. In future work, software quality data will be collected, classified and analyzed to provide further rationalization suggestions.

Acknowledgments
This research was supported by grants from the Civil Aviation Joint Funds established by the National Nature Science Foundation of China and the Civil Aviation Administration of China (No. U1533201), a project of Ministry of Industry and Information Technology of China (No.JSZL2015601C008) and the Major State Basic Research Development Program of China (973 Program) (No. 2014CB744904).

References
[1] S. Wei, L. Meihong, and Z. Guang, “Application of ODC in software process measurements and analysis,” Computer Engineering and Applications, vol. 39, no. 23, pp. 98–101, 2003.
[2] J. Lian and H. Zhu, “Software defect metric based on orthogonal defect classification,” Journal of Jianghan University, vol. 35, no. 1, pp. 62–66, 2007.
[3] Grbac T G, Željka Car, Huljenić D. A quality cost reduction model for large-scale software development[J]. Software Quality Journal, 2015, 23(2):363-390.
[4] Mejia J, Iñiguez F, Muñoz M. SMART-SPI: A data analysis model proposal for software process improvement[C]// Information Systems and Technologies. IEEE, 2017:1-7.
[5] Jung H J. The Analysis of Data on the basis of Software Test Data[J]. 2015, 13(10):1-7.
[6] Niu H, Keivanloo I, Zou Y. Learning to rank code examples for code search engines[J]. Empirical Software Engineering, 2017, 22(1):259-291.
[7] Pearson K, Galton F. Pearson product-moment correlation coefficient[J]. Covariance, 2012.
[8] Kornbrot D. Pearson Product Moment Correlation[J]. 2014:4670-4670.