A Study on Software Development Month Effort

Yinlong Liu, Yong Wang*
Ocean University of China, Qingdao, China

* Corresponding author. Tel.: 13706305013; email: wangy@ieee.org.
Manuscript submitted January 4, 2015; accepted June 8, 2015.
doi: 10.17706/jsw.10.7.798-804

Abstract: In the software development process, effort distribution based on phase is an important but often a neglected aspect of software cost estimation. Insufficient phase effort distribution study is the primary reason of poor software effort estimation. This paper provides the results of an empirical study on phase effort distribution based on 2956 projects. The software development process is divided by month which is different from the phase defined in the Life Cycle. The final results show some consistency features in effort distribution pattern of project development methods and project types, which is useful to provide a good guidance of improving the accuracy of software cost estimation.

Key words: Software cost estimate, software effort distribution, development type, month effort allocation.

1. Introduction

Early in the software industry development, the proportion of software costs in the computer system is small. But with the development of computer and software industry and the enlargement of the software scale and complexity, software costs have become the most expensive in the entire computer system. However, since the last century 60's, typical characteristics of the software crisis appear constantly, such as budget overruns, quality defects and the project delay. In the past few decades, software cost estimation methods and related technologies have made in-depth research and development. Researchers use different methods, such as parametric modeling, knowledge-based modeling, dynamic modeling, fuzzy logic, neural networks and so on [1], [2], and take the COTS (Commercial-off-the-shelf) and open source way to improve the accuracy of estimation and estimation ability [3], [4].

According to the report published by Standish group in 2004 [5], the conclusion is made from the survey of more than 50000 projects, only 29% of the project is success in finishing within budget and time limit, 18% is not completed or cancelled, and 53% is in question, namely, while complete serious cost overruns, budget overruns and even up to 89%.The main cause of a software project failed is out of control, insufficient of software cost estimation or its unstable requirement. As a result, software cost estimation and control will become critical. Due to high software cost estimation and long construction period, the resources of the company will be seriously wasted; from the other side, it will lead to the project budget overruns, out of limited time [6]. Before the software project development begins, therefore, the scale and schedule estimation is very necessary.

Compared with other factors, the software project cost estimation is more complex; because of the complexity of estimation, lack of estimation experience, human errors and poor estimation tools, these factors can lead to a large gap between estimation costs and actual costs. So the software cost estimation error is considered to be one of the four causes of software project failure in this case [7]. Software cost estimation means an activity that estimates and determines the cost of individual activities and total cost of software project through the software project plan or weight information [8]. Cost estimation is the core that estimates and determines the cost by analyzing the data, which provides the foundation and basis for the software project cost estimation and control. In software cost estimation, there are two basic requirements: one is the accuracy, the other is labor savings.

There are many kinds of software cost estimation methods: typical mathematical algorithm model with linear model, analysis model and compound model; as well as analogy method, expert judgment method, function point analysis method [9] and COCOMO model method, etc. But due to many uncertain factors, it is difficult to build a relatively accurate estimation model. In the face of some special project decisions, most software developers do not know how to face, they simply rely on experience or expert judgment, and this is the only feasible way to do[10].As a result, it is very important to provide the accurate cost estimation method to satisfy all kinds of project decision-making.

This paper selects 2956 different projects to analyze with different software development scale, different project types or different development methods. After research and analysis, the paper will take some important factors into account in the development of the effort distribution issues. The results of the study can help software engineers to estimate effort and can guide project managers to do more better allocation of resources in order to achieve the project requirements.

2. Related Work

The data of industrialization software development project has been an indispensable part of effort distribution research and several software cost estimation models which are of great fame were achieved on this basis, Norden[11]has pioneered findings that, in the field of cost estimation, the Rayleigh curve is close to the effort distribution in most development projects. The inapplicability of Rayleigh curve, however, incorporates gradually with the development of industry. The Rayleigh curve has a certain influence on the subsequent models, such as COCOMO model, the SLIM model [12]-[14]. In further study of effort distribution data, Boehm decomposed the COCOMO model on the basis of the phase distribution of the waterfall model [12], [13], and Krutchen also came up with the Life cycle model of the distribution based on RUP [15].

In 1981, Boehm put forward the COCOMO model which divided the types of software development project into the organic, the embedded and the semidetached by development environment (the project scale, the technical complexity and development experience, etc.). If the types of project were different when COCOMO was applied, the parameters of the equation were different too. The COCOMO 81 is a hierarchical model. It has three levels based on the level of details of estimation: Basis COCOMO, Intermediate COCOMO and Advanced COCOMO [13].

SLIM model [14] was proposed by Larry Putnam in 1978, a constraint model was used on productivity level observation data, derived from the Rayleigh curve. The model of software development effort distribution conforms to the Rayleigh curve, all the points corresponding to every main development activities. The basic SLIM model adjusted small projects. Y. Wang and Q. Song et al. used grey relational analysis, to predict software project effort [16]–[19] by integrating the GM (1, 1) and Verhulst Models.

Ye Yang and Mei He et al. studied the effort distribution patterns in [20]. They gave the typical sharps of project effort on the life cycle phase basis. In this paper we discussed the effort distribution employing month as the basis time unit.

Due to the different definition of life cycle phase based on different estimation models [21], phase effort distribution data has been lacking in certain degrees, as a result, the work on integrating the software life cycle, using of the model, data collection and analysis or estimation of the model calibration and related work become more complex.

3. Objective and Experiment

3.1. Objective

The dataset is provided by our partner and contains 14,000 projects at least. The dataset is also one of the largest worldwide datasets of software engineering. The validity and universality of the research method can be sufficient confirmed by enormous and various data, which provides a solid foundation for achieving a reliable and practical research result.

The projects that researched by traditional estimation model are mainly developed in the traditional development mode. However, the purpose of this study, from another point of view, is to analyze the project data based on the iterative development of modern software. We select month effort as metrics to observe the effort distribution trends along with the advancement of software development process, and define the scale of software development by development time required. As a conclusion, we can give some guidance about the effort distribution after having a deep understanding of the results.

Project overall development is divided into two stages: development and management. And through analyzing the project data collected, we finally select the two factors in the development, such as project development methods and project types. We try to get the purpose of our study by analyzing the related effort distribution. The main research questions are as follows:

1) What does the overall phase distribution profile of development effort look like?
2) Does different software development method affect the effort distribution?
3) Does different software project type affect the effort distribution?

3.2. Experiment

The dataset used in the experiment contains an enormous number of projects and deficiency of monthly effort data in each project is controlled within a rational range of dataset integrity, so, when we choose this dataset, we are selective about extraction from dataset. The result of extraction should be relatively integrated and conform to the purpose of the research, then we can analyze the data of it. Table 1 lists the metric and the steps of data cleaning.

Table 1 . The Process of Data Selection

Step ID	# ofproj. excluded	# ofproj. remained	Reason for Exclusion
1		14053	Choose Common development mode of project.
2	5498	8555	Select all stages in the development process and its activities all extracted.
3	545	8010	According to the scale of development, selected 4, 5, 6 three development scale conditions of all projects.
4	4858	3152	Remove the zero value or dirty value of each category.
5	196	2956	

After data cleaning, the dataset is a subset which is consistent with our study condition. We choose some more relative, universal and notable data to study. Specific cleaning steps are as follows: the first step is to develop a method for screening out projects that belong Common, which includes more items and is more universality but other types are relative special, so that it can be ensured the consistency and commonality; the next step is to summarize the project month effort based on the development stage and to summarize the project scale (the time required for project development); and last step is to delete zero value or null value. After these steps, we choose some scales, including 4, 5, 6 months (the selected projects accounted for 9.98%, 11.77%, 10.69% of the total), to set up our dataset. All studies we do are both based on short-cycle development.

4. Experimental Results

In the research process, we select the data that take most priority principle and then analyze data by average number, namely the data we choose several include relatively large types, at the same time, we choose the calculation based on the sample average number to be the analysis basis. So we can get more universal law in the study, and we are more likely to find the overall distribution trend:

4.1. The Effort Distribution under Different Methodology and Scale

According to the summary of different project development methodology in dataset, we found that the four methodology like the System Life Cycle 3 (SLC3), the 1.6 Development GSMS + (GSMS1.6), the Standard Systems Process (SSP), GSMS Development + (GSMS), which contains the largest number of projects, as the research objects, and the other two project types included the relatively small number, not be as the research objects. The four project types marked with the serial number, respectively M19, M24, M26, M33, the numbers of projects accounted for the overall ratio were 12.07%, 19.89%, 25.23% and 18.85%. On this basis, we study the effort distribution trend of project that the development scale is 4, 5, 6 months (Fig. 1).

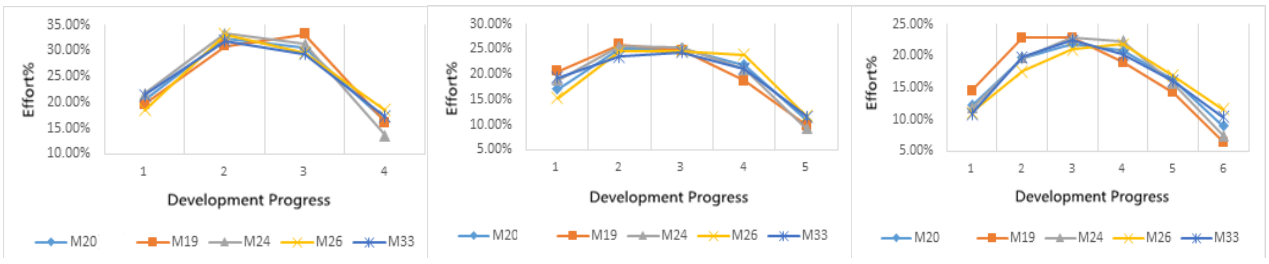

Fig. 1. Four months develop (Left), Five months develop (Middle), Six months develop (Right).

Fig. 1 gives the results of the experiment. In the Four Months Develop, only the M19 (said the curve of the box, namely SLC3) is distinguished from the distribution curve of the overall. The major difference is that effort in the third development month increases obviously. We find that the projects belong to the M19 are in the code development phase in the middle two months by returning to analyze the dataset, and the effort of the third month has a relatively higher proportion about 60 percent which may explain why the effort increase significantly in the third month that relative to the overall; For the Five Months Develop , the M19 still is more than 3.6% than the overall average in the first month but less than 3.12% than the overall average in the fourth month. It shows that more investment is needed in the early phase of the development in the M19 method. The M33 (GSMS) is same to the M19, which is higher than 2.26% than the overall average in the first month, and may be put more attention on the first month. The other two methods M24 (GSMS1.6) and M26 (SSP) is roughly same to the overall distribution trend, but the SSP is higher than 2.06% than the overall average in the fourth month. After analyzing the dataset, we found that there are about 66.7% projects which invest a lot of effort in implementation and maintenance in the last months; In the Six Months Develop, the M19 and the M26 are still different from other two methods. The former is higher than the overall average in the previous months (maximum gap is up to 3.16%), which is lower than the overall average in the later months at the same time (maximum is up to 2.5%). On the contrary, the M26 is higher than the overall average in the later months but lower than overall average in the previous months. From this results, we can find that these two methods respectively focus on the early stage and late stage, but both are roughly consistent with the overall distribution trend. Meanwhile, the projects belong to six months scale are also concentrated in the intermediate stage, which also be in line with the traditional distribution trend, but the effort of begin stage is significantly higher than that of end stage.

4.2. The Effort Distribution under Different Project Type and Scale

According to the summary of different project types in dataset, we found that the four project types like the New Development, the Enhancement, the Maintenance and the Other Projects & Services, which contains the largest number of projects, as the research objects, and the other two project types included the relatively small number, not be as the research objects. The four project types marked with the serial number, respectively 1, 2, 4, 5, the number of projects accounted for the overall ratio were 17.12%, 55.82%, 7.42% and 18.26%. On this basis, we study the effort distribution trend of project that the development scale is 4, 5, 6 months (see Fig. 2).

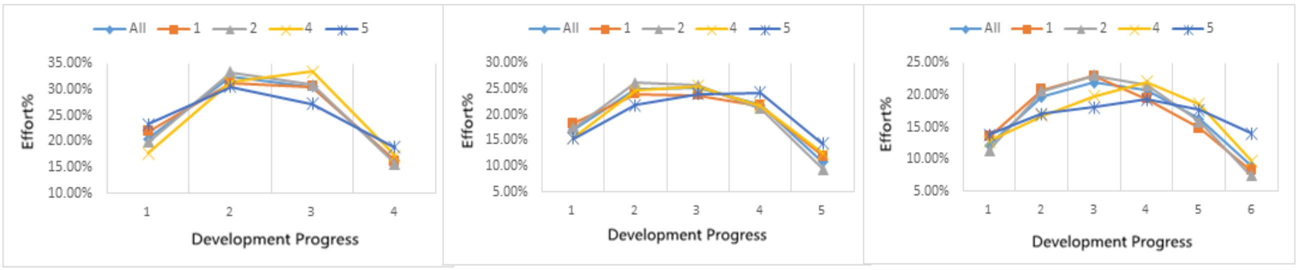

Fig. 2. Four months develop (Left), Five months develop (Middle), Six months develop (Right).

Fig. 2 gives the results of the experiment more clearly. In the Four Months Develop, the effort distribution of the projects belong to type 4 (Maintenance) and type 5 (Other Projects & Services) has a little difference from the overall, especially in the third month (type 4 is up to 2.85% and type 5 is -3.30%), and the effort distribution of the type4 in the first month is lower than 2.81% than overall average. So the type of maintenance should invest more effort in the later stage. While type 5 needs less effort in the later months in the Four Months Develop, this type needs more effort in the later months in the Five Months Develop and has an average distribution on the whole in each month of the Six Months Develop. It is a special project type. In the Six Months Develop, type 4 and type 5 are also very different from other two types.

5. Conclusions and Future Work

5.1. Conclusions

According to the experimental results, we can obtain the answers to the questions at the beginning:

1) In view of the software development process of projects that have different scale, the effort distribution trends of modern software development and traditional software development which scale are within one years(Due to space constraints, the biggest scale we listed is six months) are doing much the same thing. Namely the proportion of effort at the intermediary phase is significantly higher than other phases;
2) According to the development methods of different software projects, the GSMS 1.6 Development (M19) approach indicates that, with the increase of project development scale, the effort distribution trend of software project changes too. That is to say, the proportion of early phase of the software project is higher and the later phase seems lower. However the other three methods we applied suffered less impact of development scale relatively and had less errors compared with the entire effort distribution trend;
3) According to the types of different software projects, the type which includes maintenance and Other Projects & Services put more effort into the later phase with the increase of development scale. The

software projects of other two types we analyzed suffered less impact of development scale relatively and had fewer errors compared with the entire effort distribution trend.

Through the analysis of the results in this dataset, we summarized the effort of modern development by the month as a cycle, finally find that the effort distribution trend primarily is that the effort of beginning and ending phases is lower than the intermediate phase and the effort proportion of beginning phase is slightly higher than the ending phase. It is consistent with the curve of Rayleigh distribution.

5.2. Further Work

Due to a variety of projects attribute data, we choose a classification based on the scale of the project and focus on the effort of the development phase of the project. We study the effort distribution percentages in different project scale between development method and project type.

Future research may be the perspective of the following:

1) The project development scale we choose is relatively small and medium (development process takes 4 to 6 months), in the future we are likely to be focused on large-scale;
2) Project management also requires a certain amount of work, so we willanalyze the project effort distribution trend from the perspective of managing project;
3) From the perspective of traditional software development, we follow the traditional software development life cycle to collect effort and do some research on the effort distribution;
4) Find out a model to simulate the trend of effort distribution, then based on this, we do some simulation and optimization to find a general iterative development effort distribution rule.

Acknowledgment

This work was supported by the grants of the National Natural Science Foundation of China (No.61170312).

References

[1] Boehm, B., & C. Abts. (2000). Software development cost estimation approaches-A survey. *Annals of Software Engineering*, *10(1-4)*, 177-205.

[2] Jorgensen, M., & Shepperd, M. (2007). A system review of software development cost estimation studies. *IEEE Transaction on Software Engineering*, *33(1)*, 33-53.

[3] Basili, V. R. (1989). Software development: A paradigm for the future. *Proceedings of the 13th Annual International Computer Software and Application Conference* (pp. 471-485).

[4] Feller, J., & Fitzgerald, B. (2000). A framework analysis of the open source software development paradigm. *Proceedings of the 21st International Conference in Information Systems* (pp. 58-69).

[5] The Standish Group. 2004 the 3rd Quarter Research Report. Retrieved 2004, from http://www.standishgroup.com.

[6] Guo, N., & Zhou, X. H. (2007). *Software Project Management.* Beijing: Tsinghua University Press, Beijing Jiaotong University Press.

[7] Qi, Z. C., Tan, Q. P., & Ning, H. (2001). *Software Engineering*. Beijing: Higher Educationg Press.

[8] Swapna, K., & Rajesh, N. (2004). *Software Requirements and Estimate*. Beijing: Mechanical Industy Press.

[9] Li, M. S., He, M., & Yang, D. (2007). Software cost estimation method and application. *Journal of Software*, *4*, 775-795.

[10] Wohlin, C. (2004). Distribution patterns of effort estimation. *Proceedings of the International Conference on Scientific Organization Advancing Sciences and Applications of Information Technology and Microelectronics*.

[11] Norden, P. V. (1958). Curve fitting for a model of applied research and development scheduling. *IBM Journal Research and Development*, *3(2)*, 232-248.

[12] Boehm, B. W., *et al*. (2000). *Software Cost Estimation with COCOMO II*. Prentice Hall, NY.

[13] Boehm, B. W. (1981). *Software Engineering Economics*. Prentice Hall.

[14] Putnam, L., & Myers, W. (1992). *Measures for Excellence*. Yourdon Press Computing Series.

[15] Kruchten, P. (2003). *The Rational Unified Process: An Introduction*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

[16] Wang, Y., Song, Q., MacDonell, S., Shepperd, M., & Shen, J. (2009). Integrate the GM (1, 1) and verhulst models to predict software stage effort. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, *39(6)*, pp. 647-658.

[17] Wang, Y., Song, Q., & Shen, J. (2007). Grey learning based software stage-effort estimation. *Proceedings of the 6th International Conference on Machine Learning and Cybernetics* (pp. 1470-1475).

[18] Wang, Y., Song, Q., & Shen, J. (2007). Grey prediction based software stage-effort estimation. *Wuhan University Journal of Natural Sciences*, *12 (5)*, 927-931.

[19] Song, Q., Shepperd, M., & Mair, C. (2005). Using grey relational analysis to predict software effort with small data sets. *Proceedings of the 11th IEEE International Software Metrics Symposium* (pp. 318-327).

[20] Ye, Y., Mei, H., Li, M., *et al*. (2008). Phase distribution of software development effort. *Proceedings of the 2008 ACM-IEEE International Symposium on Empirical Software and Measurement* (pp. 61-69).

[21] Boehm, B. (2004). *Software Engineering Economics*. Beijing: Mechanical Industry Press.

Yong Wang was born in Jinan, China, in 1971. He received the Ph.D. degree in computer science from Xi'an Jiaotong University, Xi'an, China, in 2009.

He is currently an associate professor of software technology in the Department of Computer Science and Technology, Ocean University of China, Qingdao, China. His current research interests include empirical software engineering, machine learning, and data mining.

Yinlong Liu was born in Yantai, China, in 1989. He received the bachelor degree in computer science and technology from University of Jinan, Jinan, China, in 2012.

He is currently studying for master degree in Ocean University of China, Qingdao, China. His major is software engineering. His research interests include empirical software engineering, software development such as Java, C#, cloud computing, and big data.