

An Analysis of Cost-overrun Projects using Financial Data and Software Metrics

Hidetake Uwano

Department of Information Engineering
Nara National College of Technology
Nara, Japan
uwano@info.nara-k.ac.jp

Akito Monden

Nara Institute of Science and Technology
Graduate School of Information Science
Nara, Japan
akito-m@is.naist.jp

Yasutaka Kamei

Graduate School and Faculty of Information Science
and Electrical Engineering
Kyushu University
Fukuoka, Japan
kamei@ait.kyushu-u.ac.jp

Ken-ichi Matsumoto

Nara Institute of Science and Technology
Graduate School of Information Science
Nara, Japan
matumoto@is.naist.jp

Abstract—To clarify the characteristics of cost-overrun software projects, this paper focuses on the cost to sales ratio of software development, computed from financial information of a midsize software company in the embedded systems domain, and analyzes the correlation with outsourcing ratio as well as code reuse ratio and relative effort ratio per development phase. As a result, we found that the lower cost to sales ratio projects had the higher relative effort ratio in external design phase, which indicates that spending less effort in external design can cause decrease of profit. We also found that high outsourcing ratio projects had higher cost to sales ratio, and that projects having moderate code reuse ratio had lower and disperse cost to sales ratio, which suggests troubles in code reuse can damage the profit of a project.

Keywords—Cost overrun project, Cost to sales ratio, Development phase, Outsourcing, Reuse

I. INTRODUCTION

The excess production cost over scheduled cost is commonly seen in software development [1]. Major reasons for such project cost-overrun include insufficient requirement analysis, lack of project management, poor effort estimation, and frequent change requests.

To understand the characteristics of such “failure” projects, case studies and assessments for failure project analysis have been performed [2][3][4]. Also software risk evaluation (SRE) techniques [5][6][7] and estimation method for project failure [8] were proposed. These studies are useful for reducing project failure for future software development.

This paper focuses on the cost-to-sales ratio, which past researches had not focused, to distinguish success and failure of software projects. Although financial information of software development projects is an important source to understand the project results, few studies have been made so far. The cost to sales ratio directly indicates project’s profitability; hence, it is useful to analyze relationships

between the cost to sales ratio and software metrics such as effort in each development phase, to clarify factors of software success/failure in terms of project profit.

In our analysis, we computed the cost to sales ratio from financial data collected in a midsize software development company. This metric indicates how much profit was gained in each project excluding general administrative cost such as office rent cost. The project can be considered as “failure” when the cost to sales ratio was greater than a threshold (90% in this paper.)

To characterize each project, we focus on (1) relative effort ratio in each development phases, (2) outsourcing ratio, and (3) code reuse ratio. These metrics are suitable to our analysis because they are directly connected with project types and/or management strategies.

The rest of the paper is organized as follow: Section 2 and Section 3 describe a project data and metrics used in the analysis. In Section 4, we discuss the result of the analysis. Finally we conclude the paper in Section 5.

II. TARGET PROJECT

In this analysis, we used a dataset consist of 95 projects held in the midsize software development company within two years. The main business domain of the company is embedded software development for wired/wireless communication systems, image processing systems, and public transportation systems.

In this company, most projects are contract-based development; they develop software based on requirements given by other organizations. Hence, most projects consist of development phases after the requirement analysis, i.e. external design, internal design, implementation, unit testing and integration testing. To focus on the main development activity of this company, in our analysis we excluded projects that had spent more than 50 percent effort for requirement analysis or maintenance.

Table 1 show statistics of a dataset we used in the analysis, which include median, average, standard deviation,

TABLE I. STATISTICS OF A DATASET USED IN THE ANALYSIS

		Missing value (%)	Median	Average	Standard deviation
Sales (1,000 JPY)		0	15,574	34,398	46,042
Production cost (1,000 JPY)		0	13,620	30,298	39,986
Effort(Man-Hour)	Requirement analysis	0	0	199	390
	External design	0	845	1,828	2,650
	Internal design	0	359	1,177	2,023
	Implementation	0	530	832	1,158
	Unit testing	0	252	567	817
	Integration testing	0	366	823	1,421
	Other*	0	283	864	1,332
Source Lines of Code (SLOC)	Created lines	21.1	14,354	61,110	181,828
	Reused lines	21.1	88,400	278,153	520,187
	Modified lines	21.1	0	4,096	8,883

*Operations, education, maintenance, etc.

TABLE II. STATISTICS OF DERIVED METRICS

Metrics		Number of data	Median	Average	Standard deviation
Relative effort ratio	Requirement analysis	68	0.00	5.10	7.86
	External design		32.81	32.45	10.80
	Internal design		17.70	17.49	9.08
	Implementation		15.64	17.40	8.59
	Unit test		10.68	11.16	4.76
	Integration testing		14.46	16.40	7.71
Outsourcing ratio		95	52.54	43.33	27.11
Code reuse ratio		75	75.29	65.48	33.28

and the number of data cases (projects). In this paper, the production cost includes personnel cost, material cost, outsourcing cost, and other costs consumed in a project, while it excludes general administrative cost. Source lines of code (SLOC) is counted as the following three variables:

Created lines

The number of lines newly created in the target project.

Reused lines

The number of lines created in other projects and used in the target project without modification.

Modified lines

The number of lines created in other projects and modified in the target project.

In Table 1, median of effort in the requirement analysis phase is zero because most of projects started from the external design phase. Also, median of modified lines is

zero; many projects had reused lines without modification in the source code.

III. METRICS

This Section describes three metrics that can characterize the cost-overrun projects, by analyzing their relationship with cost to sales ratio of projects, which defines the success/failure of projects. Table 2 shows a list of the metrics and their statistical summary.

A. Cost to Sales Ratio

Cost to sales ratio is a percentage of production cost in sales of a target project; less than 100 percent denotes the project gain a profit by itself. However, we also need to consider general administrative cost such as office rent and/or equipment's upkeep is required to run the company. Hence, cost to sales ratio of each project must be less than a certain threshold smaller than 100.

To determine the threshold for this company, the authors interviewed with two managers. As a result, we confirmed

the average general administrative cost is about 10 percents of sales, which means the threshold of cost to sales ratio in this company is 90. In this paper, a project that has 90 and more cost to sales ratio is labeled as a “failure” project, and a project less than 90 is labeled as a “success” project. We also confirmed the classification of success/failure project by cost to sales ratio meet the manager’s intuition of project success/failure.

Fig. 1 shows a distribution of cost to sales ratio in the dataset. About 70 percent of projects are classified as “Success” project, and 87 percent of projects are in the range from 70 to 100 cost to sales ratio.

B. Relative Effort Ratio

Relative effort ratio is a percentage of effort (man-hours) spent in each development phase to the total man-hours spent on a whole project. For each phase, it can be considered that a project having much smaller or greater relative effort than other projects has a high risk of failure. For example, a project that had spent smaller effort in the requirement analysis and/or design phase can cause excess coding and/or testing effort because of need of rework in requirement analysis and/or design in later phases.

In this analysis, we selected 68 projects as an analysis target, which performed all five development phases (external design, internal design, implementation, unit test and integration test) to remove the effect of unusual projects.

C. Outsourcing Ratio

A lot of software development organization outsources a part of development phase for flexible human resource management and/or reduce the development cost. Preparation of sufficient manpower to each development project is one of the most important issues for managing/administrative person. A proper use of outsourcing in software development increases flexibility and efficiency of management, however, it also increases a risk of project failure.

In this paper, outsourcing ratio in each project is calculated as proportion of outsourcing cost to production cost of a project. 95 projects data were used for this analysis.

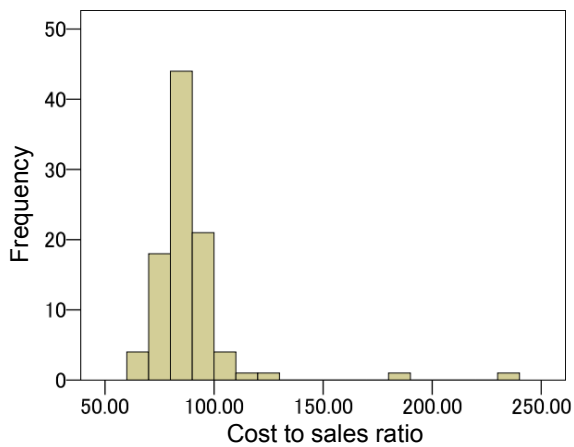


Figure 1. Frequency distribution of cost to sales ratio.

D. Code Reuse Ratio

Code reuse ratio depicts how many lines of source code were reused from past software. Reuse of source code or design document from past similar software is essential to efficient and speedy development. Reused source code has a better quality than new source code in general because it was already tested when the source code was created. Therefore, higher code reuse ratio will decrease the risk of excess test effort for unpredictable defects correction. On the other hand, understanding of the past project for correct reuse of source code is time-consuming and difficult task especially when the project has poor documentation. Code reuse without correct understanding will increase the cost of defect correction and testing.

Many recent software products were developed as maintenance or enhancement projects, hence, to understand the effect of code reuse to the project result is essential. In this paper, code reuse ratio is calculated as proportion of reused lines to total lines of code (sum of created lines, reused lines, and modified lines.) In the analysis, we used 75 projects that had no missing value in code reuse ratio.

IV. RESULTS AND DISCUSSION

A. Relative Effort Ratio

Table 3 shows relative effort ratio in each development phase. The table shows that success projects tend to have higher relative effort ratio in external design phase and lower relative effort ratio in requirement analysis phase. There is no tendency at internal design, implementation, unit testing, and integration testing phases. Fig. 2 shows a box-plot of relative effort ratio in external design phase. Each box and whiskers describe a range of relative effort ratio in external design phase. The figure shows failure projects have larger box (i.e. disperse relative effort ratio) than success projects. The result of Mann-Whitney U Test shows significant difference ($p=0.015$) between success and failure projects.

TABLE III. RELATIVE EFFORT RATIO IN EACH PHASE

	Project result	Median	p-value
Requirement analysis (%)	Failure	3.37	0.103
	Success	0.00	
External design (%)	Failure	28.18	0.015
	Success	34.31	
Internal design (%)	Failure	19.86	0.396
	Success	17.22	
Implementation (%)	Failure	16.66	0.264
	Success	15.41	
Unit testing (%)	Failure	10.29	0.545
	Success	10.68	
Integration testing (%)	Failure	13.67	0.501
	Success	15.21	

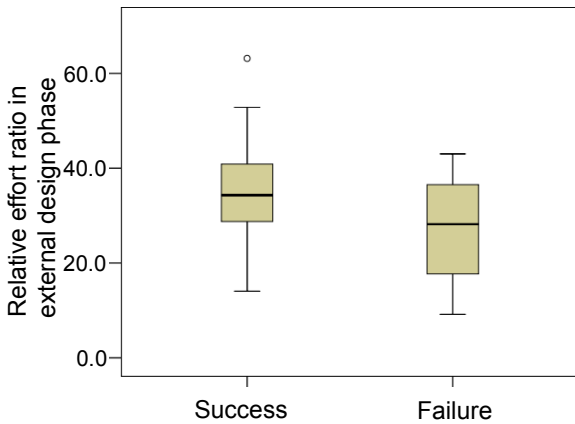


Figure 2. Relative effort ratio in external design phase of success/failure project.

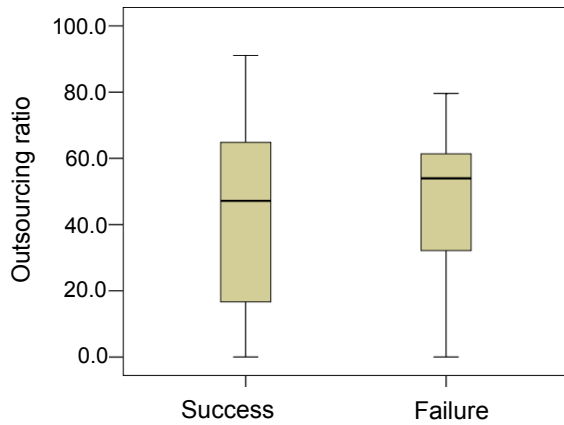


Figure 3. Outsourcing ratio of success/failure project.

The result suggests failure projects spend insufficient man-hour in external design phase, and caused more reworks and defect corrections. On the other hand, success projects could avoid reworks and defect corrections by the proper external design with sufficient effort.

B. Outsourcing Ratio

Median value of outsourcing ratio in success projects and failure projects were 47.2 percent and 54.3 percent respectively. Fig. 3 shows that outsourcing ratio in both groups were greatly dispersed, and there is no significant difference ($p=0.501$.)

We also investigated the correlation between cost to sales ratio and outsourcing ratio for more detailed understandings. We divided projects into three groups:

- 1) Largely
Project having 50 percent or more outsourcing ratio.
- 2) Partly
Project having greater than 0 percent and below 50 percent outsourcing ratio.

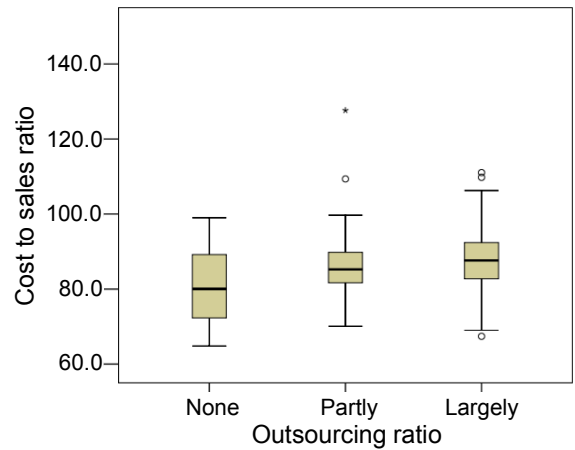


Figure 4. Box plot of cost to sales ratio in different outsourcing ratio projects.

TABLE IV. COST TO SALES RATIO IN DIFFERENT OUTSOURCING RATIO PROJECTS

	# project	Outsourcing ratio	Cost to sales ratio
Largely	49	64.3%	87.6%
Partly	29	31.2%	85.2%
None	17	0.0%	80.1%

- 3) None
Project of zero (0 percent) outsourcing ratio.

Fig. 4 describes a box-plot of cost to sales ratio in each group. The figure shows higher outsourcing projects tend to have higher cost to sales ratio. Median values of outsourcing ratio and cost to sales ratio in each group are shown in Table 4. The result of Mann-Whitney U Test showed significant differences ($p=0.034$) between “Largely” outsourcing projects and “None” outsourcing projects. This result can be interpreted as follow: largely outsourcing projects need additional efforts for meetings with contractor and/or acceptance test to deliverables. In addition to this, defect correction of deliverables created by the contractor tends to take longer time than that of in-house documents. Hence in total the project will delay and consumes unscheduled resources.

C. Code Reuse Ratio

Code reuse ratio in success/failure projects are shown in Fig. 5. Median values of “success” and “failure” projects were 66.7 percent and 87.0 percent respectively. However, in both group, large variance of code reuse ratio and projects that had very high code reuse ratio were observed. As a result, there is no significant differences ($p=0.139$) between them.

More detailed analysis of code reuse ratio is described in Fig. 6. We hypothesize that a difference of code reuse ratio represents different types of project. Here, projects were divided into three groups:

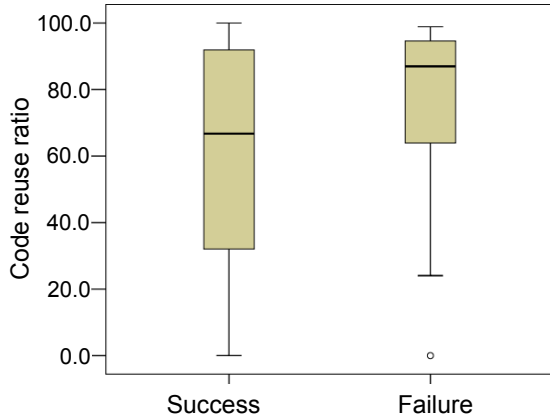


Figure 5. Code reuse ratio of success/failure project.

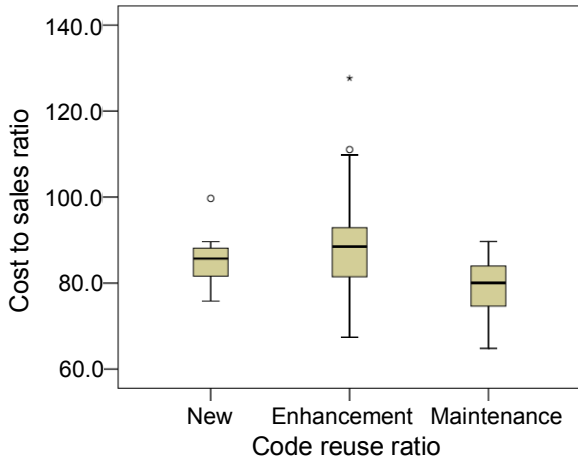


Figure 6. Box plot of cost to sales ratio in different code reuse ratio projects.

- 1) New
Project having zero (0 percent) code reuse ratio.
- 2) Enhancement
Project having greater than 0 percent and below 99 percent code reuse ratio.
- 3) Maintenance
Project having 99 percent or more code reuse ratio.

Fig. 6 shows low cost to sales ratio in “maintenance” projects and “new” projects. On the other hand, “enhancement” projects had higher (and also disperse) cost to sales ratio than others. Basically, sales price of software is determined from production cost estimated at the beginning of the project. Therefore, this result suggests estimation of production cost in enhancement projects is inaccurate. Table 5 shows median of code reuse ratio and cost to sales ratio in each group. Statistical test revealed a significant difference between “enhancement” and “maintenance” ($p=0.033$.)

TABLE V. COST TO SALES RATIO OF DIFFERENT CODE REUSE RATIO PROJECTS

	# project	Code reuse ratio	Cost to sales ratio
Maintenance	7	99.8%	80.1%
Enhancement	58	77.2%	88.5%
New	10	0.0%	85.7%

In “new” and “maintenance” projects, additional work to combine new code with existing code (i.e. understanding the existing code or testing) is relatively small, i.e. risk of unexpected additional work is low. Hence, projects finish within scheduled cost to sales ratio - less than 90 percent. In “enhancement” project, developer must understand wide range of the existing code to combine with new codes. It is difficult to accurately predict effort, therefore variance of cost to sales ratio is disperse in “enhancement” project.

For more understanding of “enhancement” projects, we divided the group into three subgroups by the cost to sales ratio. Table 6 shows median of cost to sales ratio in the three subgroups. The table describes projects that reused more than 90 percent and below 99 percent had worst cost to sales ratio. This subgroup had significant differences with “new” and “maintenance” projects. The result suggests enhancement project that had high code reuse ratio (more than 90 percent and below 99 percent) was the most risky in this company.

V. SUMMARY

This paper focused on the cost-to-sales ratio to distinguish success and failure of software projects in terms of project profit. Statistical analysis with financial data and software metrics suggested that financially “success” projects had higher effort rate in external design phase than “failure” projects. Also the result showed a tendency that high outsourcing ratio projects had higher cost to sales ratio than low outsourcing ratio projects, and middle code reuse ratio projects had higher and disperse cost to sales ratio than others.

Our analysis is based on a dataset from a midsize software company; hence supplementary analysis with other datasets is crucial to generalize the result. However, the result must be a valuable for software development organizations in similar business domain.

We used software metrics measured at the end of projects. In our future work, we plan to analyze the gap between planned metrics values and the resultant values to clarify the root causes of project success/failure.

REFERENCES

- [1] E. H. Conrow and P. S. Shishido, “Implementing Risk Management on Software Intensive Projects,” *IEEE Software*, Vol.14, No.3, pp.83-89, 1997.
- [2] B. W. Boehm, “Industrial Software Metrics Top 10 List,” *IEEE Software*, Vol.4, No.5, pp.84-85, 1987.

TABLE VI. MEDIAN OF COST TO SALES RATIO IN "ENHANCEMENT" PROJECT

	# project	Code reuse ratio	Cost to sales ratio
More than 90% and below 99%	17	97.0%	91.0%
More than 80% and below 90%	11	87.0%	89.3%
Below 80%	30	57.3%	84.9%

- [3] C. Wohlin and A. A. Andrews, "Prioritizing and Assessing Software Project Success Factors and Project Characteristics using Subjective Data," *Empirical Software Engineering*, Vol.8, pp.285-303, 2003.
- [4] A. Avritzer and E. J. Weyuker, "Metrics to Assess the Likelihood of Project Success Based on Architecture Reviews," *Empirical Software Engineering*, Vol.4, pp.199-215, 1999.
- [5] D. J. Procaccino, J. M. Verner, S. P. Overmyer, and M. E. Darter, "Case study: factors for early prediction of software development success," *Information and Software Technology*, Vol.44, No.1, pp.53-62, 2002.
- [6] R. C. Williams, G. J. Pandelios, and S. G. Behrens, "Software risk evaluation (SRE) Method Description (Version 2.0)," *Software Engineering Institute Technical Report*, CMU/SEI99TR029, 1999.
- [7] J. M. Verner, W. M. Evanco, and N. Cerpa, "State of the practice: An exploratory analysis of schedule estimation and software project success prediction," *Information and Software Technology*, Vol.49, No.2, pp.181-193, 2007.
- [8] Y. Takagi, O. Mizuno, and T. Kikuno, "An empirical approach to characterizing risky software projects based on logistic regression analysis," *Empirical Software Engineering*, Vol.10, No.4, pp.495-515, 2005.