How Does Requirements Quality Relate to Project Success or Failure?

Mayumi Itakura Kamata
Tokyo Research Laboratory, IBM Research
mitakura@jp.ibm.com

Tetsuo Tamai
The University of Tokyo
tamai@acm.org

Abstract

Our research goal is to find relations between requirements quality and project success. To attain the goal, we investigated 32 projects started and completed during the period of 2003–2005 in a large business application software development division of a company in Tokyo. Data of requirements specification quality evaluated by software quality assurance teams as well as overall project performance data in terms of cost and time overrun were available. Requirements specification quality data were first converted into a multiple-dimensional space, each dimension corresponding to an item of the recommended structure of software requirements specifications (SRS) defined in IEEE Std. 830-1998. We applied various statistical analysis techniques over the SRS quality data and project outcomes.

Some interesting relations between requirements quality and project success or failure were found, including: 1) a relatively small set of SRS items have strong impact on project success or failure; 2) descriptions of SRS in normal projects tend to be balanced; 3) SRS descriptions in Section 1, where purpose, overview and general context of SRS are written, are rich in normal projects and poor in overrun projects; 4) when the descriptions of SRS Section 1 are poor while those of functions and product perspective are rich, the project tends to result in a cost overrun.

1. Introduction

The importance of requirements engineering has been gradually recognized by the software industry but still it is hard to say that RE research results are widely in use [1]. One of the major obstacles must be the shortage of evidence that requirements quality really affects outcomes of software projects.

A report of a survey conducted by the Standish Group in 1994 [9] is repeatedly cited, when arguing about software project success and failure. Although some concerns are raised whether the report actually represents reality and is well supported by scientific facts [5, 7], the impact of the report was really strong, because the reported failure rate of software projects was staggering 70% or more. But another reason for its wide circulation might be that the failure rate counted by the number of cost overrun and time overrun projects was quite straightforward and easily accepted by the management of software development organizations.

The Standish report also shows some data on project success and failure factors. They are based on a survey of IT executive managers asking their opinions why projects succeeded or failed. "Clear statement of requirements" was the third in rank (13.0%) of project success factors responded and "incomplete requirements" was the first in rank (13.1%) of project impaired factors responded.

These numbers appear to imply relation between requirements quality and project success/failure but the data only show subjective perception of managers. Moreover, the quality of requirements is not analyzed in detail.

The aim of our research is to fill the gap of evidence concerning the relation between requirements quality and project outcomes. The evidence should be acceptable to practitioners. For that purpose, we make use of data accumulated in a company located in Tokyo, Japan.

Two kinds of data are available covering a sufficient number of projects. One is requirements quality evaluation data judged by software quality assurance (SQA) teams in the company. As the SQA process had been established and being practiced for quite a while in this company, their results can be assumed homogeneous with little fluctuation. The second kind of data is project performance monitored by performance reviewing teams. SQA teams and performance reviewing teams are different and mutually independent as well as independent from development teams. Performance is measured by cost and time comparing the current level with the estimated level prospected in the beginning of the project. Reviews are conducted multiple times separated by a certain interval during the project. Both requirements quality data and performance data were available for 32 projects. Those data were preprocessed and then supplied to a set of statistical analysis methods.

In the following sections, we will report on characteristics of target projects, the way of data preparation and sta-

1090-705X/07 $25.00 © 2007 IEEE
DOI 10.1109/RE.2007.31

tistical analysis, in depth analysis of some typical projects and the lessons learned from this study.

The contributions of this research are as follows.

1. The research is based on requirements quality data of real business projects. Quality is evaluated in real-time of software processes and the data have uniformity as they are recorded by well disciplined review teams following the established procedure. Moreover, they are rearranged to fit into the framework of IEEE Standard for "Recommended Practice for Software Requirements Specifications" [6] so that they are positioned within an objective criterion framework rather than viewed arbitrarily based on local practice of a specific company,

2. Project cost and time overrun data are collected in a consistent manner within a business organization. The data themselves should be of value supplying a different type of data from those of the Standish report. Moreover, they are analyzed in relation to requirements quality so that the results shed new light on the requirements engineering.

3. Interesting lessons are learned including unexpected findings as a result of simple but rigorous statistical analysis and individual project investigations.

2. Characteristics of Target Projects and Available Data

2.1. Target Projects

The projects under investigation of this research were all carried out by a division of a company in Tokyo. The division is in charge of developing business application systems for customers.

Data of 72 projects conducted during the period of 2003–2005 were collected. Among them, 32 projects were completed by the end of 2005 and their performance data showing existence or non-existence of time/cost overrun were available. The other projects were either still in the process of development when we started this study or performance data were not available even though the projects had been completed.

All the projects have external customers in various industries. Developed systems were middle to large in size. Unfortunately, detailed data about system size, used programming languages, application types etc. cannot be disclosed due to confidentiality but there is no project among them that has a particularly uncommon feature in those factors.

2.2. Requirements Quality Data

In all projects under our study, requirements were provided by or elicited from customers but requirements specification documents were written by the development teams and approved by the customers. During the development process, three independent teams are organized within the organization: the development team, the SQA team and the performance reviewing team. The SQA team is in charge of evaluating quality of artifacts produced at each phase of the process. The evaluation procedure has been practiced for a long time within the company using a fixed check sheet. Among a set of evaluated artifacts, we focus on requirements specifications.

A requirements specification check sheet consists of more than 100 check items. Each item is rated by the SQA team with the score range of 0 to 5. Score 0 means there is no description corresponding to the item. Scores 1 to 5 show the quality grade, the higher the better.

2.3. Project Performance Data

Reviews are held multiple times for one project, the number depending on the project size and risk judgment. Both the development team and the review team participate in a review and the review result is reported by the review team.

Among the items in review reports, we picked up evaluation of cost and time performance. Each review reports performance of the process period between the last review and the current review. And in the final review, total performance is evaluated. The evaluation is ranged between 0 to 4.

We took the final total performance data of cost and time as a project success/failure measure. In the final total review, cost and time are evaluated in comparison with the first estimate of cost and time made at the beginning of the project. A project with cost grade 0 or 1 is identified as a cost overrun project and with time grade 0 or 1 is identified as a time overrun project. Performance evaluation data of reviews during the development process were not used for statistical analysis but referred in the individual case studies.

Using this evaluation, projects are categorized into four categories:

P1 Normal without cost and time overrun,

P2 Cost overrun without time overrun,

P3 Time overrun without cost overrun,

P4 Cost and time overrun.

We decided to characterize project success and failure only by cost and time factors without considering product quality, because data on product quality as reliable as cost and time data were not available. As the 32 projects under analysis were all delivered to customers and no serious problems were reported thereafter, their product quality can be considered roughly equal.

Exact numbers of 32 projects categorized into the four categories cannot be disclosed but each class has at least five members. As it means the project distribution among the four categories is relatively balanced, comparison of the four categories can be considered reasonable.

3. Statistical Analysis

3.1. Conversion of Requirements Quality Data

The obtained requirements quality data are of good quality, because difference by evaluators were minimal due to the long-practiced and stable evaluation procedure. Moreover, there were no missing entries. However, we decided not to use the raw data directly but convert them into another form before analysis. There were two reasons for this decision.

Firstly, the number of check items is too large to manipulate. They should be projected into a lower dimensional space. Secondly, the evaluation was conducted according to local practice of a private company. To make analysis objective, it would be better to convert and interpret the data within a widely accepted standard framework. Then, we can expect our results be more useful to researchers who explore similar studies.

We took IEEE Std. 830-1998 [6] as a reference frame. A recommended structure of software requirements specifications (SRS) given by the standard is shown in Figure 1. The standard provides several alternatives for the contents of Section 3 *Specific requirements* in its appendix but we adopted the subsection structure stated in the text body. Some subsections e.g. 2.1, 3.5 and 3.7 have a finer substructure comprised of subsubsections.

We determined correspondence between the requirements quality check sheet items filled by the SQA teams and the IEEE Standard SRS entries. The mapping was made as follows.

1. The SRS structure of Figure 1 and its definitions were explained to the members of SQA teams.
2. They discussed and decided the mapping between the check sheet items and the SRS entries at the lowest level.

Accordingly, each SRS entry at the lowest level is either related to one or more check sheet items or not related at all.

1. Introduction
   1.1 Purpose
   1.2 Scope
   1.3 Definitions, acronyms and abbreviation
   1.4 References
   1.5 Overview
2. Overall description
   2.1 Product perspective
   2.2 Product functions
   2.3 User characteristics
   2.4 Constraints
   2.5 Assumptions and dependencies
   2.6 Apportioning of requirements
3. Specific requirements
   3.1 External interfaces
   3.2 Functions
   3.3 Performance requirements
   3.4 Logical database requirements
   3.5 Design constraints
   3.6 Software system attributes
   3.7 Organizing the specific requirements

Figure 1. Recommended SRS Structure (IEEE Std. 830-1998)

In the former case, the score for the SRS entry is defined by the average of the related check sheet items' scores. SRS entries in the latter case are just ignored.

Scores of all entries at the second level shown in Figure 1 are defined directly from the above procedure or obtained by taking average of the scores of their subentries. As all entries at the second level has at least one subentry related to check sheet items, ignoring SRS entries with no corresponding check sheet items causes no problem.

3.2. Requirements Quality Characteristics by Project Category

To grasp an overview of SRS quality data distributed by project category, we draw spider charts of the four project categories, as shown in Figures 2 to 5. Each radius corresponds to a SRS item and the score averaged over projects of the project category is plotted on the radius. As the average scores do not exceed 3 in all cases, the outmost circle corresponds to the score 3 in all these Figures.

Comparing the average scores of the four project categories for each SRS item, the following points are observed.

1. SRS items for which difference of scores by 0.5 or

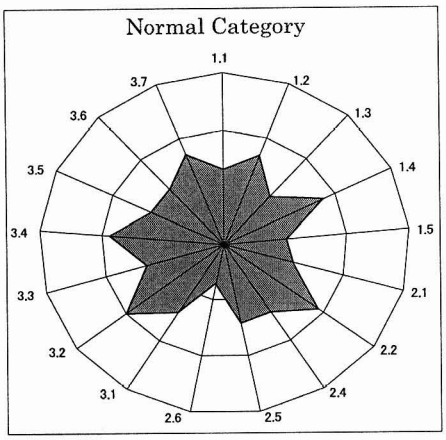

Figure 2. Spider Chart of Normal Projects

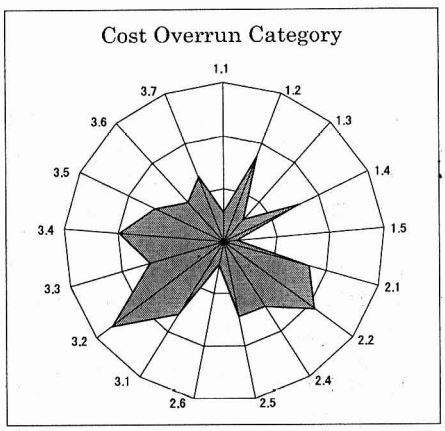

Figure 3. Spider Chart of Cost Overrun Projects

more between at least two project categories is observed are as listed below, where the definition of each SRS item is added.

- 1.1 Purpose: delineate the purpose of the SRS and specify the intended audience
- 1.3 Definitions, acronyms and abbreviations: provide the definitions of all terms, acronyms and abbreviations required to properly interpret the SRS
- 1.4 References: provide a complete list of all documents referenced in the SRS
- 1.5 Overview: describe what the rest of the SRS contains and explain how the SRS is organized
- 2.1 Product perspective: put the product into perspective with other related products and also describe how the software operates inside various constraints such as system interfaces, user interfaces, hardware interfaces, software interfaces, communication interfaces, memory, operations and site adaptation requirements
- 3.2 Functions: define the fundamental actions that must take place in the software in accepting and processing the inputs and in processing and generating the outputs, including validity checks of the inputs, exact sequence of operations, responses to abnormal situations, effect of parameters, and relationship of outputs to inputs
- 3.7 Organizing the specific requirements: specific requirements such as system mode, user class, objects, feature, stimulus, responses, and functional hierarchy.

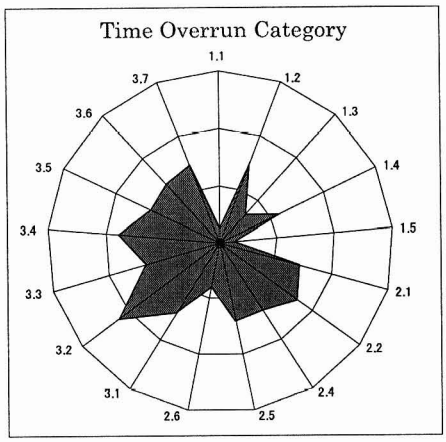

Figure 4. Spider Chart of Time Overrun Projects

2. The score of P1 was the highest among the four project categories in all SRS items except:
   - 2.1 Product perspective
   - 3.2 Functions

As the visual patterns clearly show, the shape of the normal project category P1 is balanced, close to a circle but those of all the other project categories are unbalanced. Particularly, the inner area of the upper right quadrant corresponding to SRS Section 1 is the largest in the normal project pattern among the four.

The fact that the scores of two SRS items, *Product perspective* and *Functions*, for P1 are not the highest, actually they are the lowest among the four, is somewhat counter-intuitive and thus interesting. Figure 6 and 7 show the score distribution of these items for the four categories. Compared to other items, these patterns indicate it is not that the

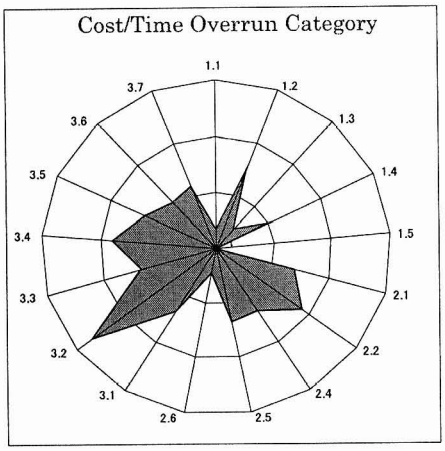

Figure 5. Spider Chart of Cost and Time Overrun Projects

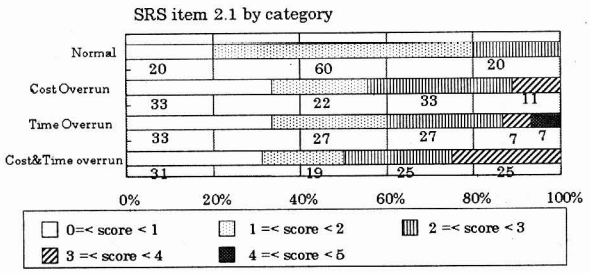

Figure 6. Score Distribution of SRS 2.1 by Category

score of normal projects is low but the scores of overrun projects are particularly high for these items.

But these observations are intuitive, not based on statistical significance analysis. Let us proceed to more rigorous analysis of relations between SRS quality factors and project success/failure.

3.3. Analysis of Variance

We conducted a typical analysis of variance to see how SRS item scores affect project outcomes. Three cases of analysis each dividing the set of projects into two groups were tested.

1. Normal project group vs. cost or time overrun project group

2. Cost overrun project group vs. cost within range project group

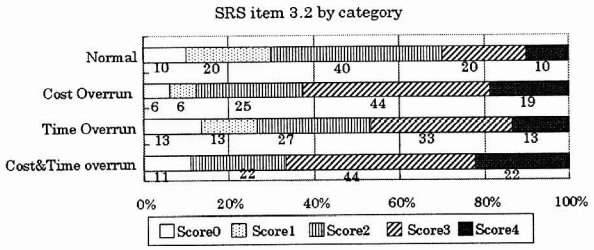

Figure 7. Score Distribution of SRS 3.2 by Category

3. Time overrun project group vs. time within range project group

For each case, a null hypothesis takes a form that the score distribution of the given SRS item of one group and that of the other group are the same. F-test and t-test are tried and when the hypothesis is refuted, it implies that the corresponding SRS item can be a factor that affects the project performance.

Normal vs. cost or time overrun Normal projects P1 and cost or time overrun projects, i.e. the union of P2, P3 and P4, are compared. In this analysis, the item 1.1 *Purpose* was found statistically significant by F-test with 95% confidence range as shown in Table 1.

Table 1. Statistically significant SRS items for normal vs. overrun

No.	Item Name	test	conf.	+/−
1.1	Purpose	F-test	95%	+

conf. = confidence

The last column of the table designates whether the factor affects positively to the first group, i.e. the normal project group, or negatively. In this case, it is positive, which means normal projects had better score for the description of SRS *Purpose* than overrun projects.

The hypothesis is not refuted for all the other SRS items either by F-test or by t-test.

Cost overrun vs. cost within range Cost overrun projects are by definition the union P2 and P4 and cost within range projects are the union of P1 and P3. Four items were found significant between these two groups as shown in Table 2. Both F-test and t-test are tried to all SRS items but only 3.2 *Functions* is found statistically significant by t-test and the other three items are found significant by F-test.

Table 2. Statistically significant SRS items for cost overrun vs. cost within range

No.	Item Name	test	conf.	+/−
1.5	Overview	F-test	95%	−
2.1	Product perspective	F-test	95%	+
2.6	Apportioning of req.	F-test	95%	−
3.2	Functions	t-test	95%	+

SRS item 2.6 *Apportioning of requirements* is defined in the IEEE Standard as "identify requirements that may be delayed until future versions of the system." It is comprehensible that such recognition will reduce the risk of cost overrun.

A particularly interesting result is that statistical significance is confirmed for cost overrun projects getting higher scores in *Product perspective* and *Functions*.

Time overrun vs. time within range Time overrun projects are by definition the union P3 and P4 and time within range projects are the union of P1 and P2. Two items were found significant as shown in Table 3.

Table 3. Statistically significant SRS items for cost overrun vs. cost within estimate

No.	Item Name	test	conf.	+/−
1.1	Purpose	F-test	99%	−
1.5	Overview	F-test	95%	−

The interesting point here is that *Purpose* is a strong negative factor to characterize the time overrun projects. The confidence range for this item is 99%.

3.4. Multivariate Analysis

The analysis so far is single variate. It is highly plausible that the factors represented by SRS items are mutually related. Thus, some type of multivariate analysis should be applied as well.

As a multivariate analysis method, we chose a tree model, because the method is intuitively comprehensible and is fitted for interpreting data that are inherently discrete, not obtained by measuring physical phenomena.

A decision tree is suitable for representing classification-type knowledge. A variable is assigned to each node and the outgoing edges from that node correspond to possible value ranges the variable can take so that the selection of an edge at each node, starting from the root node, guides the classification process. The selection path ends at a leaf that determines a group the object is to be classified in [2].

In this case, we use a binary tree, where two edges from a node correspond to two intervals of the value range separated by a threshold value. The threshold is determined statistically to maximize the deviance. To bifurcate a node, a variable that brings the greatest deviance is chosen. When a node sufficiently represents either of the groups, then the node will not be bifurcated further.

We constructed three tree models, each classifying two groups just as the analysis of variance: i.e. normal vs. cost/time overrun, cost overrun vs. cost within range and time overrun vs. time within range. We used a statistical analysis tool *R* [10] for generating tree models. As the result of cost overrun vs. cost within range is the most conspicuous, we explain that case first.

Cost overrun vs. cost within range The result is as shown in Figure 8. A node with label "xn.m" denotes a

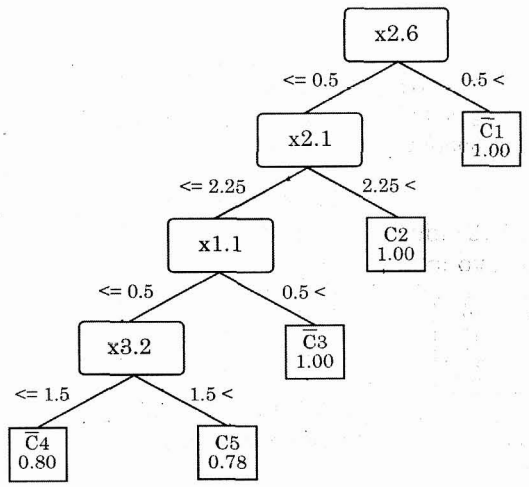

Figure 8. Decision Tree of Cost Overrun Projects

decision by the score of SRS item "n.m". The left edge of each node leads to a subset with the score smaller than or equal to the threshold, the right edge the score greater than the threshold. A leaf node with label "C" represents a subset sorted as cost overrun and label "$\bar{C}$" represents a subset sorted as no cost overrun. The number in the box denotes the fraction (1.0 means 100%) the projects in the sample of 32 classified into this leaf are actually cost overrun (or no cost overrun for the $\bar{C}$ leaf).

For example, projects with SRS 2.6 score greater than 0.5 are classified as no cost overrun by 100%. We can summarize the results as follows.

1. cost overrun projects

- SRS 2.6 score no greater than 0.5 and SRS 2.1 score greater than 2.25 (100%)
- SRS 2.6 score no greater than 0.5 and SRS 2.1 score no greater than 2.25 and SRS 1.1 score no greater than 0.5 and SRS 3.2 score greater than 1.5 (78%)

2. cost within range projects
   - SRS 2.6 greater than 0.5 (100%)
   - SRS 2.6 score no greater than 0.5 and SRS 2.1 score no greater than 2.25 and SRS 1.1 score greater than 0.5 (100%)
   - SRS 2.6 score no greater than 0.5 and SRS 2.1 score no greater than 2.25 and SRS 1.1 score no greater than 0.5 and SRS 3.2 score no greater than 1.5 (80%)

The negative effect of SRS 2.6 *Apportioning of requirements* to cost overrun can be interpreted straightforwardly as mentioned before but it is still surprising that the item is selected as the first factor and the score just exceeding the value of 0.5 identifies cost within range projects out of the sample by 100%.

The last case of cost within range is interesting to note. In this case, all factors 2.6, 2.1, 1.1 and 3.2 are in the negative side. We will come back to the issue of how to interpret this rather counter-intuitive phenomenon later.

Time overrun vs. time within range Figure 9 shows the tree model that classifies time overrun projects. In this case,

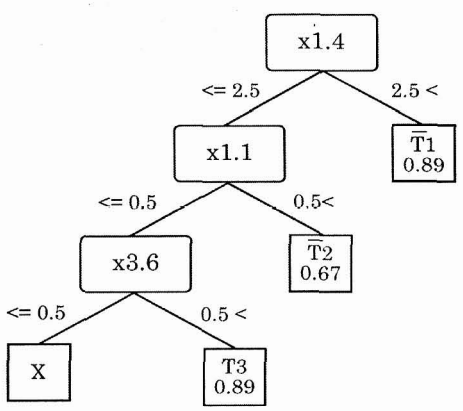

Figure 9. Decision Tree of Time Overrun Projects

SRS 1.4 *Reference* is chosen first as giving a negative effect, i.e. if references are well described, the project tends not to result in time overrun. The next factor is SRS 1.1 *Purpose* that also affects negatively. Under the condition that these two scores are low, SRS 3.6 *Software attributes* plays a positive effect, i.e. if the item has good score, the project tends to delay.

The last leaf labeled by "X" in the tree denotes a case where the decision is inconclusive, i.e. it is hard to decide to which group members in this leaf belong.

Normal vs. cost or time overrun The decision tree for normal projects is shown in Figure 10.

It is surprising that SRS 2.1 *Product perspective* is selected first and affects negatively. It will be discussed in the next section. Otherwise, the condition that high rating of both SRS 3.7 *Organizing the specific requirements* and SRS 1.1 *Purpose* is required is comprehensible.

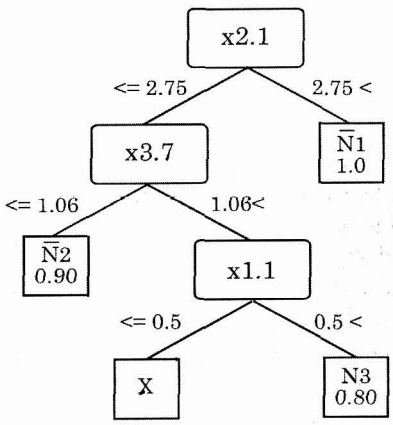

Figure 10. Decision Tree of Normal Projects

4. Closer Look at Individual Projects

Using multiple statistical analysis methods, we have found that there are relations between SRS quality and project success/failure. The general trend confirms the widely accepted assumption that higher requirements quality brings project success but some apparently opposing phenomena are also found.

To see how requirements factors actually affect software processes and outcomes, we investigated each individual project of the 32 samples in detail. Here, we pick up some typical cases, particularly focusing on the probable reasons that might have caused the *counter-intuitive* results. For the analysis of individual projects, we used not only the quantitative data treated in the preceding sections but also qualitative data available as comments made by the SQA teams and the review teams.

First, we pick up project #1[1], which is a cost overrun

[1]The 32 projects under study were taken out of a set of 72 projects and thus the projects are numbered from 1 to 72.

project. Figure 11 shows the spider chart of SRS quality of this project. The outmost circle of all the spider charts in this section corresponds to score 4, in contrast with those in Section 3 where the outmost circle corresponds to score 3.

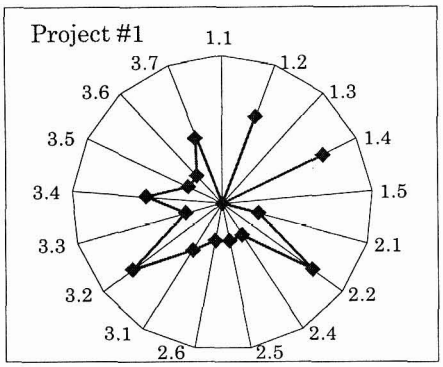

Figure 11. Spider Chart of Project #1

In Section 1 of SSR, 1.1 *Purpose*, 1.3 *Definitions, acronyms, and abbreviations* and 1.5 *Overview* are totally missing, while 1.2 *Scope* and 1.4 *References* are well written. In Section 2 and 3, 3.2 *Functions*, 2.2 *Project functions* (summary of major functions) and 3.7 *Organizing the specific requirements* have high scores.

Actually, the requirements analysis task was not conducted independently in this project but merged into the design phase and SRS and design specifications were written at the same time. That is the reason the descriptions in SRS of detailed functions are rich compared to other items. But a number of missing requirements and errors were found in the testing phase, which caused much rework to push up the cost.

Another cost overrun project, #66, goes to extreme even farther as Figure 12 shows. Description of Section 1 is nil and the same pattern of 3.2, 2.2 and 3.7 can be seen. The requirements definition task of this project was not completed during the requirements phase and the task was moved to the design phase. The situation is quite similar to #1. Detailed functional descriptions without grasping conceptual and essential requirements tend to cause much specification change and needs of rework.

However, there is a case that a cost overrun project has a balanced and beautiful shape of the SRS evaluation spider chart as shown in Figure 13. This project #19 did well in the RE phase and no problem was detected during the design and implementation phase. But it turned out that the load of the test phase was higher than projected and extra testing staff had to be added. According to the postmortem, the cause was simply a wrong estimate of testing load.

The next case is one of the time overrun projects.

In general, time overrun projects show a similar pat-

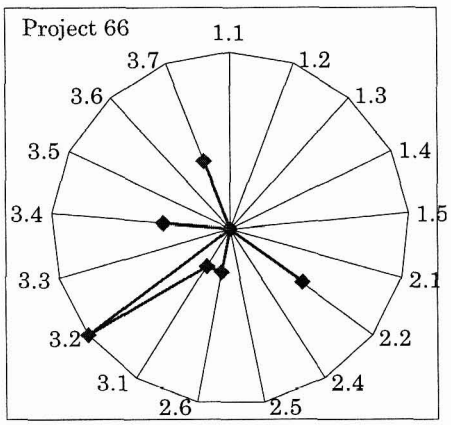

Figure 12. Spider Chart of Project #66

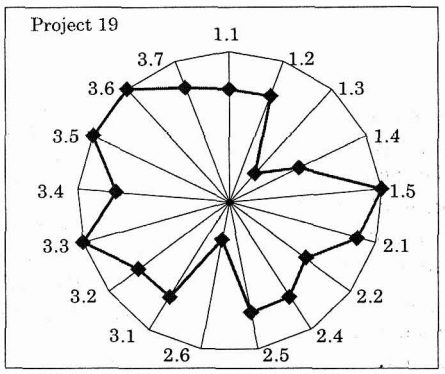

Figure 13. Spider Chart of Project #19

tern as cost overrun projects such as #1, i.e. relatively poor scores in SRS Section 1 items. The project #39, its spider chart given in Figure 14, shows a different shape. It has particularly high scores in SRS section 1. With a closer look at the documents and through interviews, we found that the cause of project delay was not poor requirements specification but use of an inappropriate tool in the implementation phase. The functions of the tool were not sufficient for the purpose and the development team had poor experience of using it.

Our last example is a cost and time overrun project, #12. In this case, the pattern of Section 1 is also similar to #1. Scores of Section 2 and 3 are higher. Particularly, high score of 2.1 is worthy to note. This project failed in completing the current system study during the requirements analysis phase. That caused frequent specification changes in the design phase, resulting in cost and time overrun. It also implies interface to the current system was complicated and as a result, SRS 2.1 *Product perspective* which is supposed to describe interfaces to other systems among others turned

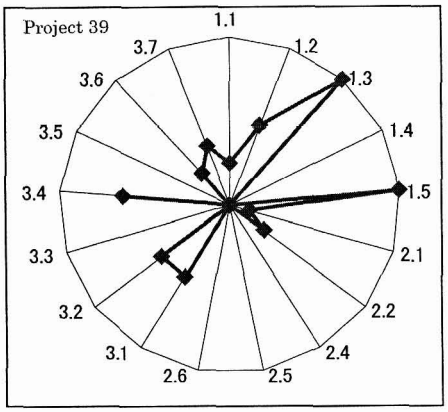

Figure 14. Spider Chart of Project #39

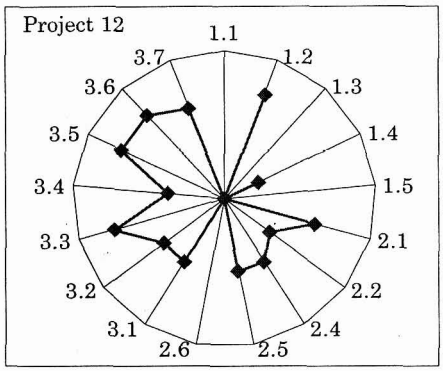

Figure 15. Spider Chart of Project #12

out to be relatively highly rated. This case partly addresses the question raised in the last section.

5. Threats to Validity

The first probable threat to validity might be reliability of SRS quality evaluation data. Admittedly, there is a risk of dispersion between evaluators and room for human errors. However, compared to other similar work, we believe that the dataset used in this study has an advantage for the following reasons.

1. Evaluation was made and recorded in real time as projects proceeded. The data are different in nature from those collected through questionnaires or interviews after projects are over.

2. The procedure for evaluation is established, having been practiced for a long time and the fixed check sheet is used. Data are different from those obtained through asking responders' or interviewees' subjective views.

3. Data have been kept through daily practices by the dedicated SQA teams. They were not collected newly by researchers for the sake of research.

The second probable threat to validity might be its limited data source. Data were collected from a single organization and characteristics of projects are basically uniform. This is an advantage in the sense that data are coherent and well suited for statistical analysis. However, comparison with other kinds of data covering different applications and organization types will be fruitful.

6. Related Work

One of the most important studies related to our work is the one reported by Damian & Chisan [3]. They conducted an intensive study of a large software development project that had newly introduced a requirements process improvement program. The case study extended 30 months to follow the RE process improvement activities while the research process was divided into three stages. The approach can be characterized by: 1) targeting a single large project; 2) using questionnaires prepared by the researchers to collect data from managers, team-leaders and senior engineers; and 3) focusing on RE processes rather than quality of RE products. In this sense, their work and ours are complementary to each other.

Another example of an in-depth case study on a single company can be found in [12]. It is interesting to note that they adopted a single criterion of whether software delivery was on time to judge project success.

Sommerville & Ransom [8] also focused on RE process assessment and improvement. Nine companies were contacted in this case and the RE process maturity model proposed by the authors was used to assess their processes. Then, process improvements were recommended and practiced. The eventual goal of the research was to correlate improvements in RE processes to business performance, which is challenging but hard to achieve as the authors admit.

The goal of the work by Verner et al. [11] is closer to ours. They tried to find relationships between requirements practices and software project outcomes. The approach they took was to distribute questionnaires to practitioners in U.S. and Australia. The respondents answered to questions that characterized RE practice of the projects they knew, which they considered either successful or a failure. The authors admit that "surveys are of course based on self-reported data which reflects what people say happened, not what they actually did or experienced."

A research on measuring the success of RE processes is reported by Emam & Madhavji [4]. They listed up 32 indicators for measurement and classified them into two major

dimensions, quality of RE products and quality of RE service. Their analysis is closed within RE processes and not related to project success or final product quality.

7. Conclusions and Future Work

What we found through our study can be summarized as follows.

1. Data indicate there is relationship between SRS quality and project outcomes. Moreover, a relatively small set of SRS items have strong impact.
2. Descriptions of SRS in normal projects tend to be balanced. When the SRS item evaluation rating is plotted on the spider chart, the pattern shows a figure close to a circle.
3. SRS descriptions in Section 1, where purpose, overview and general context of SRS are written, are rich in normal projects and poor in overrun projects. Particularly, when references or purpose in Section 1 are well written, the project tends to finish on time.
4. When the descriptions of SRS Section 1 are poor while those of functions and product perspective are rich, the project tends to result in a cost overrun, because such characteristics often indicate that the RE phase has been neglected or absorbed in the design phase.
5. Identifying requirements whose implementation may be delayed can be a good way for preventing cost overrun.

In the near future, we plan to collect more data from the same organization and corroborate or enrich the current findings. It will also be interesting to try other kind of multivariate analysis such as principal components analysis.

In the long term, it will be valuable to apply a similar approach to other areas, including embedded software systems and COTS products.

Acknowledgment

The authors would like to thank Takehiko Yasukawa for his kind support and advice in conducting statistical analysis.

References

[1] D. Berry, D. Damian, A. Finkelstein, D. Gause, R. Hall, E. Simmons, and A. Wassyng. To do or not to do: If the requirements engineering payoff is so good, why aren't more companies doing it? In *Proc. 13th International Requirements Engineering Conference (RE'05)*, page 447. IEEE, 2005.

[2] L. Breiman, J. H. Friedman, R. A. Ohshen, and C. J. Stone. *Classification and Regression Trees*. Wadsworth, Belmont, CA, 1984.

[3] D. Damian and J. Chisan. An emipirical study of the complex relationships between requirements engineering processes and other processes that lead to payoffs in productivity, quality, and risk management. *IEEE Transactions on Software Engineering*, 32(7):433–453, July 2006.

[4] K. E. Emam and N. H. Madhavji. Measuring the success of requirements engineering processes. In *Second IEEE International Symposium on Requirements Engineering*, pages 204–211, 1995.

[5] R. L. Glass. The standish report: Does it really describe a software crisis? *Communications of the ACM*, 49(8):15–16, Aug. 2006.

[6] IEEE. Recommended practice for software requirements specifications. Technical report, IEEE, 1998. IEEE Std 830-1998.

[7] M. Jorgensen and K. Molokken-Ostvold. How large are software cost overruns? A review of the 1994 chaos report. *Information and Software Technology*, 48(4):297–301, Apr. 1997.

[8] I. Sommerville and J. Ransom. An empirical study of industrial requirements engineering process assessment and improvement. *ACM Transactions on Software Engineering and Methodology*, 14(1):85–117, Jan. 2005.

[9] Standish Group International. The chaos report (1994). http://www.standishgroup.com/sample-research/chaos_1994_1.php, 1994.

[10] The R Foundation for Statistical Computing. The R project for statistical computing. http://www.r-project.org/index.html.

[11] J. Verner, K. Cox, S. Bleistein, and N. Cerpa. Requirements engineering and software project success: An industrial survey in Australia and the U.S. 13:225–238, 2005.

[12] H. Wohlwend and S. Rosenbaum. Software improvements in an international company. In *15th International Conference on Software Engineerig (ICSE'93)*, Baltimore, MD, USA, 1993.