Assessing the Quality of Software Requirements Specifications

Conference Paper · September 2008

DOI: 10.1109/RE.2008.29 · Source: DBLP

2 authors, including:

Eric Knauss
University of Gothenburg
166 PUBLICATIONS **1,515** CITATIONS

SEE PROFILE

Assessing the Quality of Software Requirements Specifications

Eric Knauss and Christian El Boustani
FG Software Engineering, Leibniz Universität Hannover
Welfengarten 1, 30167 Hannover, Germany
eric.knauss@inf.uni-hannover.de, info@christian-eb.de

Abstract

Software Requirements Specifications (SRS) are hard to compare due to the uniqueness of the projects they were created in. In practice this means that it is not possible to objectively determine if a projects SRS fails to reach a certain quality threshold. Therefore, a commonly agreed-on quality model is needed. Additionally, a large set of empirical data is needed to establish a correlation between project success and quality levels.

As there is no such quality model, we had to define our own based on the Goal-Question-Metric (GQM) method. Based on this we analyzed more than 40 software projects (student projects in undergraduate software engineering classes), in order to contribute to the empirical part.

This paper contributes in three areas: Firstly, we outline our GQM plan and our set of metrics. They were derived from widespread literature, and hence could lead to a discussion of how to measure requirements quality. Practitioners and researchers can profit from our experience, when measuring the quality of their requirements. Secondly, we present our findings. We hope that others find these valuable when comparing them to their own results. Finally, we show that the results of our quality assessment correlate to project success. Thus, we give an empirical indication for the correlation of requirements engineering and project success.

1 Study Goals

Based on the GQM method [**?**] our main goal was to assess the quality of a Software Requirements Specifications (SRS) and to connect it to project success (as defined in section 2). Figure 1 outlines our systematical measurement of SRS quality. The term *formal requirements quality* refers to verbalization rules as in [**?**] (e.g. completely specified process words, avoidance of incomplete comparisons, etc.). These metrics are objective and can be easily measured. However they reflect the quality of the SRS's content only indirectly. In contrast, *content related requirements quality* refers to goals that need interpretation to some extend. Having to judge, for example, whether the SRS contains a quality model or not, the assessor has to search for quality aspects and decide, if they are sufficiantly detailed.

Figure 1. Measurement goals.

2 Hypotheses

In order to judge a project's success we interviewed each customer and asked him to rate the success based on the following scale:

++ The project's results (i.e. software) are used in the intended way.

+ The project's results could be used in the intended way, but there are better solutions available to reach the customer's goals.

- Projects in this category failed to reach some of the customer's goals. The customer believes that these goals are reachable within a month of rework.

-- This category consists of projects that failed to deliver working software. These projects failed the acceptance tests and the customer does not believe that it would pay off to continue the project.

Note that our definition of project success differs from [**?**]: Our projects cannot overrun time and budget, because they are stopped at the end of term. If they cannot deliver, they have failed (category --). Only projects in category ++ are considered successful.

Concerning the relationship between SRS quality and project success we have the following hypotheses:

Hypothesis 1 Projects with a high quality-score are more likely to succeed (category ++).

Influencing factors: Relationship between formal quality aspects and quality of the SRS's content as reported in [**?**]. A high quality SRS might also be a sign of well-organized teams, more likely to succeed in delivering valuable software.

Hypothesis holds if we find an upper threshold with more than 75% of the projects scoring above, fall in category ++ or +.

Hypothesis 2 Projects with very low quality-score are likely to fail (category --).

Influencing factors A low-quality SRS is bad enough. But teams that produce a bad SRS might have additional problems. E.g., team members may work against each other or may have a bad time-managing. These difficulties may multiply as the project proceeds.

Hypothesis holds if a lower quality-threshold can be found, with more than 75% of projects scoring below, fail (- or --).

3 Results and Conclusion

Figure 2 shows an excerpt from the data we obtained. This data covers the 16 projects conducted in the last two years. To get a quick overview of the SRS, we summed up the weighted measurement results. Weights were choosen according to [**?**]. The results support our hypotheses:

Hypothesis 1: As shown in figure 2, 87% of the projects that score more than 44 on the quality-assessment fall into category ++.

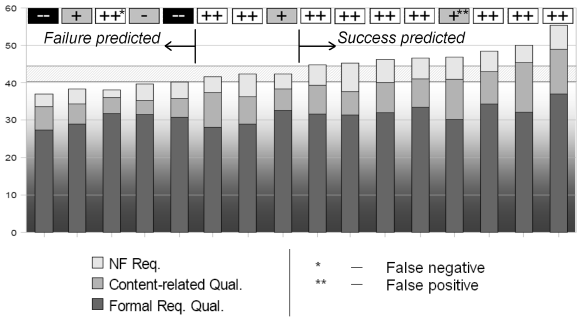

Figure 2. Results and Thresholds

Hypothesis 2: All projects that failed (- and --) scored below 40 points and we found only one project from category ++ below this threshold. Therefore, 80% of the projects below the lower threshold were not satisfactory.

Investigations of 24 additional projects support this. Based on these additional results we do not suppose that the upper threshold could be set lower. To sum up, our results suggest a strong relationship between quality of requirements engineering and project success.

Future work needs to accumulate more empirical data and eliminate or reduce the various threats to validity we identified. Especially the variations in the measurement result depending on the assessors person need to be adressed. We identified three sources for this derivation: False positives (e.g. req. wrongly classified as passive voice), scope of interpretation (e.g. identifying process words), and generally subjective measures (e.g. understandability). The first two classes describe systematic errors that can be addressed by training, keyword-lists, and heuristic tools. However, there will allways remain a certain degree of subjectivity, despite all these efforts. Further investigation needs to quantify these effects and ensure that this does not lead to wrong conclusions.