Assessing the reliability of developers' classification of change tasks: A field experiment

Technical report 12-2008
Hans Christian Benestad,
Simula Research Laboratory

Abstract. Many software projects routinely record information about *change type* as part of their change management process. Such information can be used to make priorities, to clarify responsibilities, and to monitor and act on trends during software evolution. The objective of this field experiment was to empirically assess the reliability of developers' classifications according to two alternative classification schemes, in the context of a commercial software project. The project members perform classifications of already completed changes according to the dimensions of maintenance by Swanson and according to quality factors included in the ISO9126 standard. The results show that in this project, the inter-rater agreement was low by normal standards, for both the investigated schemes. The conclusion is that projects should take explicit actions to improve and evaluate the reliability of such classification before they are used as input to management decisions.

1 Introduction

Changes analysis can provide valuable insight into the nature of evolution of software artifacts. In an ongoing case study, statistical models that relate characteristics of changes to development effort are constructed. By studying how these models change over time we expect to obtain deeper insight into the maintainability of the software, beyond traditional productivity measures.

Information about *change type* is frequently collected as part of a change management process, and is one of several candidate explanatory variables in the models. Change type is usually a subjective, categorical measure (on the nominal scale), however several different categorization schemes exist. A simple scheme would only differentiate between *defects* and *improvements*. A well known categorization of software maintenance activities was provided by Swanson [1], using the categories of *corrective*, *perfective* and *adaptive* maintenance. In agile software projects, it is customary to differentiate between *fixes*, *enhancements* and *refactorings*. By viewing every change as an enhancements to *some* quality factor, quality models, such as ISO9126 [2] and McCall [3], could be used to categorize changes. The mentioned categorization schemes differ with respect to granularity and focus; however they all express some *purpose of change*.

The choice of categorization scheme is dependent on the goal of the data collection and the nature of changes in the project at hand. However, since the data is subjective, it is also necessary that the data can be reliably collected, meaning that a software project is consistent in their categorization of changes. With inconsistent classification, the value of the collected data is reduced, and the use of such data for management decisions may even be harmful.

In the mentioned project under study, categorization was performed by the reporter of a change request, using the categories *bug* and *enhancement request*. The categorization seemed to be very much dependent on the perspective of the reporter, end users being more inclined to report all operational problems encountered as bugs. Attempts by the author to perform post-hoc categorizations revealed a substantial number of borderline cases: The purpose of many changes was to improve the handling of abnormal situations in the operation environment. Because the requirements were not documented at this level of detail, it was difficult to judge whether the change was corrective, perfective, or even adaptive. Using the ISO9126 scheme, such changes could more naturally be classified as improvements to the quality factor *reliability*. This motivated the use of this quality model as an alternative classification scheme.

In the rest of this paper we describe the experiment that was set up to assess and compare the reliability of classifications according to two alternative categorization schemes. Related work is presented in section 2. Section 3 describes the experimental plan. The results are presented and discussed in section 4, 5, and 6. Section 7 concludes.

2 Related work

We have found no other work that empirically evaluates the reliability of change classifications. The more specific area of *defect classifications* has received more interest: The work by El Emam parallels our work when it comes to the experimental context, i.e. industrial software project, and analyses method, i.e. kappa-statistics to measure inter-rater agreement [4]. Henningson performed a similar experiment in an academic context [5].

Chapin has recognized that due to commonly encountered deviations in classification schemes and their definitions, classifications should be based on objective evidence of people's activities rather than on their intentions [6].

3 Research design

3.1 Goal and research questions

The goal of the experiment was to assess and compare the reliability of classifications according to two alternative categorization schemes in the context of an ongoing software project. The research questions were:

R1. What is the inter-rater agreement using the ISO9126 and Swanson scheme to classify completed changes in the project under study?
R2. Are there significant differences in the reliability using either of the two classification schemes?

3.2 Experiment design

The basis of the experiment design was to instruct the developers use alternative schemes to classify a number of changes, randomly selected from their change management system. Assessment and comparison of reliability when classifying according to the alternative schemes are based on kappa-statistics, which provides standardized measures of inter-judge agreement.

3.2.1 Classification scheme constructs

ISO9126 defines two hierarchical levels of quality factors. The developers were instructed to categorize at the most specific level possible. The original texts were translated to Norwegian by the researcher.

For the Swanson scheme, we used Norwegian translations of the original categories of Corrective, Perfective and Adaptive changes. In order to help explaining these broad categories, we introduced a second level to Corrective and Perfective maintenance. The details can be found in appendix I.

With two-level schemes it is possible to perform analysis and comparison at both levels. Because the second level of the Swanson categories were defined by the researcher, analysis focuses on the first level. Second-level categorization was mapped to its corresponding parent category. A general option “Not possible to decide” was also added to the schemes.

3.2.2 Assignment of change tasks to developers

60 change tasks were selected by random from the change management database. The design elements below describe the specifics in how these changes were assigned to the developers:

- Each of the eight developer classified 30 changes according to the ISO scheme and 30 changes according to the Swanson scheme. This design element eliminates the possibility that the results for each schema were confounded by individual differences, which would have been a threat if one developer used one classification scheme only.
- For each developer, the 60 change tasks where allocated randomly and in random order to classification scheme, under the restriction that every change should be classified four times according to the ISO scheme, and four times according to the Swanson scheme. This design element eliminates the potential effect that sequence of change tasks influence differently for the two schemes.
- Half of the developers were allowed 30 minutes for classifying according to the ISO scheme, then 30 minutes for classifying according to the Swanson scheme. The other group used the reverse sequence. This design element eliminates the threat that the results for each schema were confounded by time pressure or fatigue.
- The two groups were balanced with respect to the familiarity with the selected changes. The familiarity was measured by counting the tracks that each developer had left in the change management system for the 60 selected changes. This

design element reduces a potential interaction effect between individual differences and the sequence in which the classification schemes were applied.

3.3 Experiment operation

The eight participating developers were co-located for the duration of the experiment, and were under the supervision of two researchers. Two spreadsheets where distributed by email to each developer, each containing:

- Definition of the current categorization scheme and categories
- Instructions for the current part of the experiment
- The identifier of 30 changes, hyperlinked into the web-based change management system
- Drop down-menus from which the developers could select a change type
- Open text fields were developers were allowed to write any comment
- Closed questions about the developers' perception of the classification scheme

The developers were instructed to follow the hyperlinks into the change management system, read the information about the change in this system, and then select the best matching classification from the predefined drop-down menus in the spreadsheet. Timing information was provided 10 minutes and 5 minutes before the expiry of the 30-minute sessions. The spreadsheets were e-mailed back to the researcher at the end of the second session.

Few anomalies were recorded during the operation. All developers finished within the scheduled time. One developer arrived 10 minutes late, but was allowed to finish 10 minutes late. One developer signalled a time squeeze, but claimed that his judgments were not affected by this.

3.4 Analyses method

The original kappa-statistics by Cohen provides a measure of agreement between two judges on a nominal scale [7]. If the raters are in complete agreement, kappa = 1. If the agreement corresponds to what could be expected by pure chance, kappa = 0. Fleiss generalized this measure to the case with more than two judges [8]. The analysis in this study is based on the measure by Fleiss, calculated using a SAS-macro available from SAS institute [9]. A review of the literature in various disciplines provides guidelines for interpreting Kappa, as well as interpretation guidelines for using Kappa in evaluating the reliability of software process assessments is provided in [10]. In general, Kappa values less than 0.45 indicate inadequate agreement. Values above 0.62 indicate good agreement, and values above 0.78 indicate excellent agreement. We will use these guidelines in order to answer the research question.

To be able to judge whether there are significant differences between the schemes, we need to be able to judge whether differences are more extreme than what can be expected from normal variation when conducting repeated measurements. A jack-knifing procedure is used to calculate a value of the standard error for the kappa values of the two schemes. Assuming that the kappa-values are normally distributed when performing repeated measurements (according to the central limit theorem), a 95% confidence intervals can be established by subtracting and adding two times the standard error to the obtained kappa-value. With non-overlapping confidence intervals we can conclude that the kappa-values are significantly different.

For assessment of the absolute value of the kappa-values, that is, whether the agreement can be said to be high, a similar calculation can be performed: It is the lower end of the confidence interval that should be tested against the cited threshold for inadequate, good and excellent agreement.

4 Results

4.1 Agreement on first level

In Table 1, the values and standard error are given for Fleiss' kappa-statistic.

Table 1; First-level agreement

	Kappa	Kappa Standard Error
ISO9126	0.37693	0.031540
Swanson	0.35503	0.040255

Research question 1
The 95% confidence interval can be approximated to 0.31-0.43 for the ISO9126 scheme, and to 0.27-0.43 for the Swanson scheme. Since both these interval falls beneath the limit of high and moderate agreement, we can conclude that the agreement is low.

Research question 2
The confidence intervals are overlapping, and we can therefore not conclude that there is a difference between the classification schemes.

4.2 Agreement on second level

In Table 2, the values and standard error are given for Fleiss' kappa-statistic

Table2: Second-level agreement

	Kappa	Kappa Standard Error
ISO9126	0.27584	0.021538
Swanson	0.28914	0.025969

Research question 1
The 95% confidence interval can be approximated to 0.23-0.31 for the ISO9126 scheme, and to 0.23-0.34 for the Swanson scheme. Since both these interval falls beneath the limit of high and moderate agreement, we can conclude that the agreement is low.

Research question 2
The confidence intervals are overlapping, and we therefore not conclude that there is a difference between the classification schemes.

5 Threats to validity

Psychological factors. The company was paid on an hourly basis to participate in the experiment. The subjects explicitly volunteered to participate. Nevertheless, we cannot rule out the possibility that some developers might have a negative attitude towards this kind of data collection. However, the observations and discussions with the developers gave no indication in this direction.

Understanding of categories. The possibility of every developer to consistently assign a change to the same category presumes that the developers have understood the category in the same manner. This requires that the categories are defined in an unambiguous, easy-to-understand manner. This again, requires that the underlying domain - the nature of software change - is unambiguous and easy to understand.

The documentation of changes. Changes may have been insufficiently documented in the change management system. However, because the classifications were performed post-hoc, the level of documentation was equal or better than expected in a realistic classification situation.

From the perspective of the project the results are strong and conclusive, with only limited threats to validity (see above). However, from a general perspective, this research should be viewed as a case study. Nevertheless, it can be assumed that the situation is similar other software projects as well: The investigated project followed a well-organized change process. The experience level of the developers was high, general reputation of the software company studied was high. The project was of medium size (16 000 hours of development, so far. It is not possible, at least from this study, to precisely describe when similar situations might occur. One important factor can be the level of “agility” in the project. With few written requirements, constant change, and no explicit separation between development and maintenance, it is possible that traditional categorization schemes become less appropriate.

6 Discussion

6.1 Implications for research and practice

Based on the above, we recommend that similar investigations should be made in projects planning to record change type for managerial use. The limited resources required to conduct such an experiment (90 minutes in our case) make this recommendation viable. Tasks for the experiment are immediately available in change management systems. Two weeks of research time was used for the planning, but we consider that this can be significantly reduced if experiments are conducted along the lines of the one reported in this paper. Sound analysis requires knowledge of inter-rater agreement statistics, but simple calculations of proportions of agreement can provide enough insight.

What actions should a project take if reliability of classifications is shown to be low? First, it must be established that the disagreement was not due to simple misunderstanding of categories. A summary session together with the participants would be appropriate to reveal this, using the collected data as a basis for creating improved definitions that are less ambiguous in the context at hand. Second, by finding patterns of typical disagreements, categories could be merged or split. Another approach would be to perform a qualitative analysis of the contents of the change management database, using a grounded theory-like approach to extract commonalities among changes. This again could form the basis for a custom categorization scheme. This is also in parallels with the recommendations in [11], which states that a quality model is created from the requirements of the project at hand, since no static quality model can fulfil the requirements of all software projects. This paper has described relations to such quality model, and the same conclusion may be valid when the quality models are used as change classification schema.

7 Conclusion and further work

The experiment has provided our case study with conclusive evidence that the inter-rater reliability of change classifications is low when classifying according to the dimensions of maintenance by Swanson and the ISO9126 quality model. Because we assume the situation to be similar in other projects as well, we recommend that a verification of the reliability of change characterizations is adopted as a standard procedure before such information is collected and used as input to managerial decisions.

References

1. Swanson EB. The dimensions of maintenance. *Proceedings of the Proceedings of the 2nd international conference on Software engineering.* IEEE Computer Society Press, 1976; 492-497.
2. ISO/IEC, "Software engineering — product quality — part 1: Quality model," 2001.
3. McCall J, Richards P, and Walters G. Factors in software quality. General Electric Command & Information Systems Technical Report 77CIS02 to Rome Air Development Center, 1977.
4. El Emam K and Wieczorek I. The repeatability of code defect classifications. *Proceedings of the The Ninth International Symposium on Software Reliability Engineering.*1998; 322-333.
5. Henningsson K and Wohlin C. Assuring fault classification agreement - an empirical evaluation. *Proceedings of the International Symposium on Empirical Software Engineering.*2004; 95- 104.
6. Chapin N, Hale JE, Khan KM, Ramil JF, and Tan W-G. Types of software evolution and software maintenance. *Journal of Software Maintenance and Evolution: Research and Practice* 2001; **13**(1):3-30.
7. Cohen J. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement* 1960; **20**(37-46.
8. J.Fleiss. Measuring nominal scale agreement among many raters. *Psychological Bulletin* 1971; **76**(5):378-382.
9. SAS-Institute. Compute estimates and tests of agreement among multiple raters, http://support.Sas.Com/ctx/samples/index.Jsp?Sid=507. Retrieved May 2006;
10. Emam KE. Benchmarking kappa: Interrater agreement in software process assessments. *Empirical Software Engineering* 1999; **4**(2):113-133.
11. Kitchenham B, Linkman S, Pasquini A, and Nanni V. The squid approach to defining a quality model. *Software Quality Journal* 1997; **6**(3):211-233.