A Qualitative Method for Mining Open Source Software Repositories

John Noll, Sarah Beecham, and Dominik Seichter

Lero, the Irish Software Engineering Centre,
Department of Computer Science and Information Systems,
University of Limerick, Limerick, Ireland
{john.noll,sarah.beecham,dominik.seichter}@lero.ie

Abstract. The volume of data archived in open source software project repositories makes automated, quantitative techniques attractive for extracting and analyzing information from these archives. However, many kinds of archival data include blocks of natural language text that are difficult to analyze automatically. This paper introduces a qualitative analysis method that is transparent and repeatable, leads to objective findings when dealing with qualitative data, and is efficient enough to be applied to large archives.
The method was applied in a case study of developer and user forum discussions of an open source electronic medical record project. The study demonstrates that the qualitative repository mining method can be employed to derive useful results quickly yet accurately. These results would not be possible using a strictly automated approach.

Key words: Open Source Software, Electronic Medical Record, Qualitative Research

1 Introduction

The sheer volume of data archived in open source software project repositories makes automated, quantitative techniques attractive for extracting and analyzing information from these archives.

However, some kinds of archival data - bug reports, commit log entries, email messages, and forum postings - include large blocks of natural language text that are difficult to analyze automatically. Software development is a human-intensive activity; these qualitative data convey important information about a project that cannot be explained by numbers alone. For example, analyzing project discussion forum postings can help to explain how users are supported, who is reporting and fixing bugs, who actually commits enhancements, and how requirements are elicited.

While qualitative techniques employing human interpretation are necessary to analyze such data, qualitative analysis is a labor-intensive activity; as such, the amount of data that can be analyzed is limited by the capabilities of human researchers.

This paper introduces a hybrid data-mining technique that combines automated data extraction with human qualitative analysis. The approach is transparent and repeatable, produces objective results from qualitative data, and is suitable for a reasonably large project archive.

At the core of the technique is a classification scheme for classifying natural language fragments such as mailing list messages An iterative process is employed to develop a set of categories to classify natural language text, such as discussion forum posts. Inter-rater agreement measures are used to refine the list until a high degree of agreement among researchers is achieved. The resulting categories are then used to classify a representative sample of text artifacts. The results can then be aggregated to provide a quantitative summary of qualitative data.

We describe the method in the next section, including use of inter-rater agreement analysis to refine the coding scheme. The last two sections present related work and conclusions.

2 Method

The method proposed by this study employs *content analysis* Krippendorff [10], a classification technique that is frequently applied to interviews and focus group data. The objective of content analysis is to ask quantitative questions about qualitative data. The approach is similar to the grounded theory method, but differs from grounded theory in that the results are quantitative rather than qualitative: content analysis produces results such as, "49% of messages submitted to project mailing lists were sent by core developers."

Our method is adapted from Burnard [2] and comprises the following specific steps:

Develop Initial Code Set. The first step is to create an initial set of codes by analyzing a small, representative sample of text fragments. Typically these would be elements in the project repository, such as bug reports, discussion forum posts, commit log entries, etc.

This is an inductive step: the researcher reads a fragment and invents a code (word or phrase) that captures the meaning of the fragment. During this step, the list of codes grows and evolves as more fragments are read and the research becomes familiar with the content; the resulting list may be large and therefore require consolidation.

Coalesce Codes into Themes. A good coding scheme has a small set of codes, with clear definitions, so that the scheme is easy to apply and can be performed quickly. As such, the next step is iteratively coalesce codes with similar meaning into a single category, and assign a new code to the category. When the list has coalesced into a handful of categories with distinct meanings, the process ends and the category codes become the codes that are assigned to text fragments during the content analysis phase.

Create Checklist. A checklist describing how to categorize a given text fragment is developed from the set of disjoint codes from the previous step. This checklist guides the coding process, providing a step-by-step decision list for the researcher to use to code the data.

Refine Codes and Checklist. The set of codes and associated checklist are evaluated and refined using a series of trials involving two or more researchers. The goal is to achieve a high degree of agreement among researchers about which code should be assigned to a given text fragment. This is achieved by having two researchers apply the checklist to a small sample of text fragments independently. The results are then compared using *crosstabulation* to see how they agree; disagreements are discussed to determine how the checklist or set of codes could be refined to make the choice of correct code more clear, and the process is repeated until an acceptable level of agreement is achieved.

Table 1. Crosstabulation table comparing coding of two researchers.

	Sarah					
John	fix	impl	issue	other	prop	Total
fix	3	0	0	3	0	6
impl	0	2	0	2	0	4
issue	1	1	15	0	0	17
other	1	1	2	21	2	27
prop	0	0	0	2	3	5
Total	5	4	17	28	5	59

Table 1 is an example of a crosstabulation created from a trial coding exercise used to refine the coding scheme and checklist for the case study described in [14]. Both the rows and columns are labelled with codes from the coding scheme. The cells show the number of messages coded with the row label by the first researcher (John) that were coded with the column label by the second researcher (Sarah). The diagonal, therefore, represents agreement. The table shows that both John and Sarah assigned seventeen *issue* codes; of these, fifteen were assigned by both researchers to the same text fragments. This table makes clear where disagreements lie: *prop* has only 40% agreement ($frac25$), *impl* has 50% ($\frac{2}{4}$), and *fix* has 60% ($\frac{3}{5}$), while both *issue* ($\frac{15}{17}$) and *other* ($\frac{21}{27}$) have more than 75% agreement.

Assess Inter-rater Agreement. Cohen's kappa [1, 3] is a statistic that attempts to assess the degree of agreement between the codes assigned by two researchers working independently on the same sample. Cohen's kappa accounts for the reality that a certain level of agreement would be achieved even if codes were assigned at random; as such, it is more conservative than simply calculating the percentage of agreement between two researchers, which does not account for randomness.

Cohen's kappa produces values between 0 and 1, where 0 indicates poor agreement, and 1 perfect agreement. Landis and Koch [11] proposed an assessment scheme for determining strength of agreement from Cohen's kappa values: less than .2 represents "slight" agreement; a value between .4 and .6 represents "moderate" agreement; a kappa statistic above .8 is considered a sign of "almost perfect" agreement. Researchers have to balance agreement against the effort required to refine the checklist

and coding scheme in order to achieve high agreement. If "good" agreement has been achieved (kappa value between .6 and .8), and successive refinement attempts produce incremental or no improvement, it may be best to move on to actual coding.

Code the Data. In this phase, several researchers apply the coding checklist to code a large sample extracted from the project archives.

Analyze Coded Data. For example, the case presented in [14], coded discussion forum posts were combined with author IDs extracted from the discussion forums and commit logs, to create a picture of what kinds of activities different groups of project participants were involved in.

2.1 Validation

The method was applied to a case study of an open source software project [14]. The case study results show that useful conclusions can be drawn from minimally structured natural language data that are not easily analyzed using an automated approach. Also, the method for iteratively developing a classification scheme using inter-rater agreement analysis proved to be an effective approach for developing a repeatable yet efficient coding scheme. Finally, we found that qualitative analysis techniques can be employed to derive results quickly and accurately through careful transparent and validated analysis, and dividing the work among several researchers.

3 Related Work

Testing inter-rater reliability is not a new concept, and has been used in software engineering qualitative research.

For example, Henningsson and Wohlin [9], used Cohen's kappa statistic to measure whether eight people could agree on how to classify faults independently. Other researchers used the same method to test the reliability of their fault classifications, with mixed results. However, in contrast to the approach described in this paper, both Henningsson and Wohlin [9], and Hall et al. [8], used the interrater measure to test agreement post hoc, and not as a tool to resolve problems with the coding scheme. However, El Emam and Wieczorek [4] were able to use poor kappa values to go back to look at specific fault types that were causing low repeatability of code defect classifications.

Researchers studying software process assessment have used Cohen kappa statistic to test the external reliability of interrater agreement [5–7, 12, 15]. As software process assessment can be subjective, researchers identify the need to check the reliability of the results. Kohen's kappa has also been applied in fuzzy systems [16], and in the subjective evolvability evaluation of object-oriented software [13]. Also, Vilbergsdttir et al. [17] used kappa statistics over several iterations to revise their scheme defining usability attribute values.

To our knowledge this process has not been used in classifying data mined from software repositories, and therefore as we identify this to be an area that is potentially

high in subjectivity, we think the community can benefit from assessing the quality of classifications prior to making any judgements about what the data are telling us.

4 Conclusions

This paper presented an approach to software repository data mining based on qualitative content analysis, a data analysis technique that is appropriate for situations where data cannot be easily quantified by automated data mining techniques. The approach was successfully applied in a case study of an open source software project.

The results of the case study show that useful conclusions can be drawn from minimally structured natural language data that are not easily analyzed using an automated approach. Further, the application of inter-rater agreement analysis in the case study demonstrates how an effective coding scheme can be created that is transparent, repeatable, and consistent. This allows several researchers to work independently on content analysis, while still producing results that can be reliably aggregated. Finally, our experience shows the method is efficient as well as effective: researchers were able to code more than 60 messages per hour, meaning we were able to complete the coding and analysis in a week working part-time.

5 Acknowledgments

This work was supported, in part, by Science Foundation Ireland grant 03/CE2/I303_1 to Lero - the Irish Software Engineering Research Centre (`www.lero.ie`).

References

[1] R. Bakeman. Behavioral observation and coding. In H. T. Reis and C. M. Judge, editors, *Handbook of research methods in social and personality psychology*, pages 138–159. Cambridge University Press, 2000.

[2] P. Burnard. A method of analysing interview transcripts in qualitative research. *Nurse Education Today*, 11:461–466, 1991.

[3] M. E. Dewey. Coefficients of agreement. *British Journal of Psychiatry*, 143: 487–489, 1983.

[4] K. El Emam and I. Wieczorek. The repeatability of code defect classifications. In *Proceedings, Ninth International Symposium on Software Reliability Engineering*, Nov. 1998.

[5] K. El Emam, D. Goldenson, L. Briand, and P. Marshall. Interrater agreement in SPICE-based assessments: some preliminary results. In *Proceedings, Fourth International Conference on the Software Process*, Dec. 1996.

[6] K. El Emam, J.-M. Simon, S. Rousseau, and E. Jacquet. Cost implications of interrater agreement for software process assessments. In *Proceedings, Fifth International Software Metrics Symposium*, Nov. 1998.

[7] P. Fusaro, K. El Emam, and B. Smith. Evaluating the interrater agreement of process capability ratings. In *Proceedings, Fourth International Software Metrics Symposium*, Nov. 1997.

[8] T. Hall, D. Bowes, G. Leibchen, and P. Wernick. Evaluating three approaches to extracting fault data from software change repositories. In M. A. Babar, M. Vierimaa, and M. Oivo, editors, *Product-Focused Software Process Improvement*, volume 6156 of *Lecture Notes in Computer Science*. Springer Berlin/Heidelberg, 2010.

[9] K. Henningsson and C. Wohlin. Assuring fault classification agreement - an empirical evaluation. In *International Symposium on Empirical Software Engineering (ISESE '04)*, Aug. 2004.

[10] K. Krippendorff. *Content Analysis: An Introduction to Its Methodology*. Sage Publications, 2nd edition, 2004.

[11] J. R. Landis and G. G. Koch. An application of hierarchical kappa-type statistics in the assessment of majority agreement among multiple observers. *Biometrics*, 33(2):363–374, June 1977.

[12] H.-Y. Lee, H.-W. Jung, C.-S. Chung, J. M. Lee, K. W. Lee, and H. J. Jeong. Analysis of interrater agreement in ISO/IEC 15504-based software process assessment. In *Proceedings Second Asia-Pacific Conference on Quality Software*, 2001.

[13] M. V. Mantyla. An experiment on subjective evolvability evaluation of object-oriented software: explaining factors and interrater agreement. In *International Symposium on Empirical Software Engineering*, Nov. 2005.

[14] J. Noll, S. Beecham, and D. Seichter. A qualitative study of open source software development: the OpenEMR project. In *5th International Symposium on Empirical Software Engineering and Measurement (ESEM '11)*, Banff, Alberta, Canada, Sept. 2011.

[15] H.-M. Park and H.-W. Jung. Evaluating interrater agreement with intraclass correlation coefficient in SPICE-based software process assessment. In *Proceedings, Third International Conference on Quality Software*, Nov. 2003.

[16] S. Vieira, U. Kaymak, and J. Sousa. Cohen's kappa coefficient as a performance measure for feature selection. In *2010 IEEE International Conference on Fuzzy Systems (FUZZ)*, July 2010.

[17] S. G. Vilbergsdttir, E. T. Hvannberg, and L.-C. Law. Classification of usability problems (CUP) scheme. In *Proceedings of the 4th Nordic conference on Human-computer interaction (NordiCHI '06)*, 2003.