Software Ticket Quality

Andrei-Mihai Nicolae

School of Computing Science
Sir Alwyn Williams Building
University of Glasgow
G12 8QQ

Masters project proposal

December 18th 2017

Contents

1 Introduction

briefly explain the context of the project problem

1.1 A subsection

Please note your proposal need not follow the included section headings; this is only a suggested structure. Also add subsections etc as required

2 Statement of Problem

Clearly state the problem to be addressed in your forthcoming project. Explain why it would be worthwhile to solve this problem.

3 Background Survey

3.1 Introduction

3.2 What was covered and why

3.3 How papers were found; what terms were searched for + snowballing

3.4 Data Quality Metrics

Impact of Poor Data Quality[11]:

- **What were the goals?** The main purpose of the article is to provide some insights into how data quality can have a major economical impact on a company, as well as to raise awareness about this issue.
- **What was the method?**
- **What did they learn?**
- **Relevance to our work**:

Metrics for Measuring Data Quality[5]:

- **What were the goals?** The authors of the paper aimed to answer the two main research questions:
  - how can data quality be measured as metrics?
  - what can be done in regards to data quality and what economic consequences would it bring?
- **What was the method?** They applied the metric for timeliness at a major German mobile services provider. Due to some Data Quality issues the company was having, they had lower mailing campaign success rates, but after applying the metrics, the company was able to establish a direct connection between the results of measuring data quality and the success rates of campaigns.
- **What did they learn?** For some practical problems, the metrics that they applied in regards to data quality proved to be quite appropriate. As they designed the metrics to revolve around interpretability and cardinality, they could quantify data quality, thus they could analyse the economical impact.
- **Relevance to our work**: this paper could possibly help us in measuring data quality. This is an important step in our research as we specifically need to measure quality in software tickets.

3.5 Issue Quality

Bug Report Assignee Recommendation using Activity Profiles[10]:

- **What were the goals?** The main goal of the paper is to automatically detect who should be the developers who are most suitable to solve a specific issue.
- **What was the method?** They employed an algorithm using activity profiles (i.e. assign, review, resolve activity of the developer) such that, after detecting the prior experience, developer's role, and involvement in the project, it could recommend who should fix a specific bug. The average accuracy was around 88%, much higher than the LDA-SVM technique.
- **What did they learn?** They found out that their approach was more accurate than previous work done on assignee recommendation, but they think that it might even be improved further by combining their technique with the classic LDA-SVM one.
- **Relevance to our work**: this work complements the research presented in Anvik et al. [1] and it will provide us with valuable insight into how automatic triaging should be performed.

Modeling Bug Report Quality[6]

- **What were the goals?** The authors' goal is to present a model that would determine if a bug report will be triaged in a given amount of time. Moreover, they are trying to propose features that could help composing a good quality bug report.
- **What was the method?** They run an analysis on over 27000 bug reports from the Mozilla Firefox project, selecting a couple of surface features that could be applied:
  - self-reported severity;
  - readability;
  - daily load (i.e. total number of bug reports that need to be dealt with at that point in time);
  - submitter reputation;
  - changes over time.

  Afterwards, they ran 4 experiments (i.e. validate the assumptions, find how much post-submission data is necessary, find the optimal resolved by cutoff, evaluate the hypothetical benefit) and analyzed the results to see if their model would be beneficial.
- **What did they learn?** They found out that linear model, even a simple one, can have better-than-chance predictive power. Also, they came to the conclusion that attachment and comment counts are really valuable for triaging the bug faster, compared to, for example, patch count. Moreover, the readability of a bug is important as it proved to influence the duration of the triaging process.
- **Relevance to our work**: the authors' goal was quite similar to ours as they are trying to find out which features make for a good quality bug report. Moreover, they are also using programmatic analysis instead of interviews with developers to produce the results, which is the same approach we are aiming for in our research.

Secret Life of Bugs[2]:

- What were the goals? The paper tries to understand and analyze common bug fixing coordination activities. Another goal of the paper was to analyze the reliability of repositories in terms of software projects coordination and propose different directions on how to implement proper tools.
- What was the method? They executed a field study which was split into two parts:
  - firstly, they did an exploratory case study of bug repos histories;
  - secondly, they conducted a survey with professionals (i.e. testers, developers).

  All data and interviews were conducted using Microsoft bug repositories and employees.

- What did they learn? They learned that there are multiple factors which influence the coordination activities that revolve around bug fixing, such as organizational, social and technical knowledge, thus one cannot infer any conclusions only by automatic analysis of the bug repositories. Also, through surverying the professionals, they reached the conclusion that there are 8 main goals which can be used for better tools and practices:
  - probing for ownership;
  - summit;
  - probing for expertise;
  - code review;
  - triaging;
  - rapid-fire emailing;
  - infrequent, direct email;
  - shotgun emails.
- Relevance to our work: this paper is one of the key papers for our research paper.

What makes a good bug report[3]:

- **What were the goals?** The main goal of the paper was to investigate the **quality of bug reports** from a developer's point of view, based on the typical information found in such a report (e.g. stack traces, screenshots).
- **What was the method?** The authors conducted a massive survey with over 450 respondents. The survey was online and it targeted developers from Mozilla, Apache and Eclipse.
- **What did they learn?** The main conclusion of this research paper was that well written bug reports will more likely get the attention of the developers. Thus, including steps to reproduce the bugs or stack traces proved to increase the quality of the bug report. Also, an important achievement reached by the authors was the development of a prototype tool called Cuezilla that could estimate, with an accuracy rate of 3148%, the quality of a bug report.
- **Relevance to our work**: it is relevant to our research in that it provides valuable insight into what makes for a good bug report based on actual professionals' opinions.

Who should fix this bug?[1]:

- **What were the goals?** The authors aimed to create a tool that could automatically assign the bug report to a specific developer based on his/her suitability for that specific task.

- **What was the method?** They applied a supervised machine learning algorithm on the repositories to learn which developers were best suited for specific tasks, thus when a new bug report would come in, a small set of people would be selected. In order to train the algorithm, they looked at Bugzilla repositories and selected the free text form of tickets, trying to label similar ones based on textual similarities. Once the tickets were labeled and grouped for specific developers, the algorithm would then be able to present the triager the set of developers suitable to fix the bug.
- **What did they learn?** The most important lesson learned was that collecting data from bug reports and CVS logs was quite challenging. One of the major reasons why they found this aspect hard was that not all CVS comments referenced the specific bug report id.
- **Relevance to our work** The paper taught us that bug triaging is hard and that there is almost no automated tool that can choose the perfect developers to work on the task. Moreover, the method applied by the authors could prove useful as a learning aid when working with the open source repositories chosen as data sets.

Software Quality The Elusive Target[8]:

- **What were the goals?** The main goal of the paper is to determine what makes for a good quality software project, as well as who are the people in charge of this aspect and how should they approach achieving it.
- **What was the method?** They tried to define quality in software projects and analyze techniques that measure such quality by looking at other models proposed in different other papers (e.g. McCall's quality model, ISO 9126).
- **What did they learn?** They learned that quality is very hard to define and there are various factors which need to be taken into consideration, such as the business model of the company, the type of the software project (e.g. safety critical, financial) or the actors which are involved and how they coordinate the software activities.
- **Relevance to our work**: this paper is a key paper on software quality and it can prove beneficial in our research by giving valuable insights into how software quality can be modeled, thus helping us in selecting good quality open source repositories to work with (i.e. ticket selection and analysis).

Code Quality Analysis in OSS[13]:

- **What were the goals?** The article tries to discuss and examine the quality of the source code delivered by open source projects.
- **What was the method?** They used a set of tools that could automatically inspect various aspects of source code. The authors analyzed the 6th release of the OpenSUSE project and examined only the components, which are defined by C functions in the programs.

- **What did they learn?** The research's results show that Linux applications have high quality code standards that one might expect in an open source repository, but the quality is lower than the one implied by the standard. More than half of the components were in a high state of quality, but on the other hand, most lower quality components cannot be improved only by applying some corrective actions. Thus, even though not all the source code was in an industrial standards shape, there is definitely room for further improvement and open source repositories proved to be of good quality.
- **Relevance to our work**: this work completes the previous paper on software quality in general by looking specifically at quality in open source projects, which will be our main points for data collection.

Analysis of Software Cost Estimation[4]:

- **What were the goals?** The authors are trying to show that poor estimation analysis techniques in software projects will lead to wrong conclusions regarding cost estimation accuracy. Moreover, they also propose a framework for better analysis of software cost estimation error.
- **What was the method?** They approached a real-world company where they conducted analysis on their cost estimation techniques.
- **What did they learn?** They learned that regular, straight-forward types of cost estimation analysis techniques error lead them to wrong conclusions.
- **Relevance to our work**: it showed us that we need to be careful when selecting techniques for cost estimation in our own research.

3.6 Measuring Cost and Waste in Software Projects

Waste Identification[9]:

- **What were the goals?** The paper had two main goals:
  - a means to identify communication waste in agile software projects environments;
  - types of communication waste in agile projects.
- **What was the method?** The authors collaborated with a medium-sized American software company and conducted a series of observations, informal discussions, documents provided by the organization, as well as semi-structured interviews. Moreover, the data collection for waste identification was split into 2 parts:
  - **pre-development**: occured before the actual implementation begun (e.g. backlog creation);

  - **development**: happened throughout the implementation process (e.g. throughout sprints, retrospectives, sprint reviews, communication media).

- **What did they learn?** They realized the communication waste can be divided into 5 main categories:

  - lack of involvement;
  - lack of shared understanding;
  - outdated information;
  - restricted access to information;
  - scattered information.

  Also, they learned that their way of identifying these types of waste was quite efficient and they even recommend it to companies if they'd like to conduct such processes internally.

- **Relevance to our work**: the waste identification process can be applied to our work so that we can identify possible causes to poor quality tickets.

Software Development Waste[12]:

- **What were the goals?** The main goal of the paper was to identify main types of waste in software development projects.

- **What was the method?** They conducted a participant-observation study over a long period of time at Pivotal, a consultancy software development company. They also interviewed multiple engineers and balanced theoretical sampling with analysis to achieve the conclusions.

- **What did they learn?** They found out there are nine main types of waste in software projects:

  - building the wrong feature or product;
  - mismanaging backlog;
  - extraneous cognitive load;
  - rework;
  - ineffective communication;
  - waiting/multitasking;
  - solutions too complex;
  - psychological distress.
  - **Relevance to our work**: this paper complements the previous one on waste identification[9].

Waste in Kanban Projects[7]:

- **What were the goals?** The authors are trying to find the main sources of waste in Kanban software development projects and categorize/rank them based on severity.
- **What was the method?** A controlled case study research was employed in a company called Software Factory. They conducted semi-structured interviews with 5 of the team members both in the beginning in order to collect data as well as at the end of the whole process to categorize the seven types of waste found. Moreover, they also measured the overall success of the project based on Shenar's techniques (first-second-third-fourth; project efficiency-imapct on the customer-business success-preparing for the future).
- **What did they learn?** They reached two main findings:
  - they found 7 types of waste throughout the project at various development stages:
    * partially done work;
    * extra processes;
    * extra features;
    * task switching;
    * waiting;
    * motion;
    * defects.
  - they reached the conclusion that they couldn't explain the success of the project even though waste was found.
- **Relevance to our work**: this work completes the findings from the previous work presented as most of the projects we will work with will be Agile, thus Kanban-based in terms of issue management.

3.7 Sentiment Analysis

Thumbs Up or Thumbs Down[14]:

- **What were the goals?** The main goal of the paper is to detect the overall sentiment transmitted through reviews of various types.
- **What was the method?** The author created an unsupervised machine learning algorithm that was evaluated on more than 400 reviews on Epinions on various kinds of markets (e.g. automobiles, movie). The algorithm implementation was divided into three steps:
  - extract phrases containing adjectives or adverbs;

  - estimate the semantic orientation of the phrases;
  - classify the review as recommended or not recommended based on the semantic orientation calculated at previous step.

- **What did they learn?** One thing the author learned that different categories will yield different results. For example, the automobile section on Epinions ranked much higher, 84%, compared to movie reviews, which had an accuracy of 65.83%. Moreover, most pitfalls of the algorithm could be attributed to multiple factors, such as not using a supervised learning system or limitations of PMI-IR.
- **Relevance to our work**: the method can be applied for extracting sentiments from the tickets (description and comments) we will use in our own research.

Recognizing Contextual Polarity[15]:

- **What were the goals?** The paper's main goal is to find efficient ways to distinguish between contextual and prior polarity.
- **What was the method?** They used a two step method that used machine learning and a variety of features. The first step classified each phrase which had a clue as either neutral or polar, followed by taking all phrases marked in the previous step and giving them a contextual polarity (e.g. positive, negative, both, neutral).
- **What did they learn?** Through the method the authors employed, they managed to automatically identify the contextual polarity. As most papers were only looking at the sentiment extracted from the overall document, they managed to get valuable results from looking at specific words and phrases.
- **Relevance to our work**: when analyzing the description and comments of the ticket, we can use their method for infering the sentiment transmitted probably more accurately than the technique used in Turney's paper[14].

3.8 Conclusion

4 Proposed Approach

state how you propose to solve the software development problem. Show that your proposed approach is feasible, but identify any risks.

5 Work Plan

show how you plan to organize your work, identifying intermediate deliverables and dates.

References

[1] John Anvik, Lyndon Hiew, and Gail C Murphy. Who should fix this bug? pages 361–370, 2006.

[2] Jorge Aranda and Gina Venolia. The secret life of bugs: Going past the errors and omissions in software repositories. pages 298–308, 2009.

[3] Nicolas Bettenburg, Sascha Just, Adrian Schröter, Cathrin Weiss, Rahul Premraj, and Thomas Zimmermann. What makes a good bug report? pages 308–318, 2008.

[4] Stein Grimstad and Magne Jørgensen. A framework for the analysis of software cost estimation accuracy. pages 58–65, 2006.

[5] Bernd Heinrich, Marcus Kaiser, and Mathias Klier. Metrics for measuring data quality foundations for an economic data quality management. pages 87–94, 2007.

[6] Pieter Hooimeijer and Westley Weimer. Modeling bug report quality. pages 34–43, 2007.

[7] Marko Ikonen, Petri Kettunen, Nilay Oza, and Pekka Abrahamsson. Exploring the sources of waste in kanban software development projects. pages 376–381, 2010.

[8] Barbara Kitchenham and Shari Lawrence Pfleeger. Software quality: the elusive target [special issues section]. *IEEE software*, 13(1):12–21, 1996.

[9] Mikko Korkala and Frank Maurer. Waste identification as the means for improving communication in globally distributed agile software development. *The Journal of Systems and Software*, 95:122–140, 2014.

[10] Hoda Naguib, Nitesh Narayan, Bernd Br"ugge, and Dina Helal. Bug report assignee recommendation using activity profiles. pages 22–30, 2013.

[11] Thomas C Redman. The impact of poor data quality on the typical enterprise. *Communications of the ACM*, 41(2):79–82, 1998.

[12] Todd Sedano, Paul Ralph, and Cécile Péraire. Software development waste. pages 130–140, 2017.

[13] Ioannis Stamelos, Lefteris Angelis, Apostolos Oikonomou, and Georgios L Bleris. Code quality analysis in open source software development. *Information Systems Journal*, 12(1):43–60, 2002.

[14] Peter D Turney. Thumbs up or thumbs down?: semantic orientation applied to unsupervised classification of reviews. pages 417–424, 2002.

[15] Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. Recognizing contextual polarity in phrase-level sentiment analysis. pages 347–354, 2005.