Automatic Contextual Bug Report Modelling

Andrei-Mihai Nicolae (2147392)

1 Research Problem

Background

In recent years, software engineering has been shaping many industries that were previously considered unrelated to technology, ranging from healthcare to public transportation and education. As software has grown and inflicted itself into so many fields, it has inherently become more complex and it now involves teams of even hundreds of developers working remotely or in the same office on applications that can shape even the face of a nation, such as Estonia's data exchange layer called X-Road [3].

Therefore, having such a high number of developers working on complex applications, there has been a need for software engineers to plan, track and manage the workload for increased efficiency. Thus, issue tracking systems have been created, which usually incorporate multiple components into a single, unified interface. The main component, however, of any issue tracking system is called a *ticket*. A ticket is comprised of all the necessary information for either fixing a bug (i.e. specific malfunction of the application) or implementing a new feature (i.e. new functionality for the software). However, after an application is released and it starts to get used by actual end users, the most common ticket found in issue tracking systems is a bug report, stating what went wrong inside the application, how it can be reproduced, even possible locations in the source code where the malfunction might originate from.

Typically, the people who file the bug reports are actual end users who go onto the project's issue tracking system website and complete the necessary data required by that system. Once the bug report is filed in the tracking systen, triagers are responsible for assigning that ticket to developers to get fixed. Bettenburg et. al [1] have conducted interviews with developers and they have gathered valuable data related to what makes for a good bug report and what are the key characteristics that help them fix the bug as quickly as possible. One result presented by this research is that having stack traces (i.e. list of methods in the source code which were called when an exception was thrown) in the description field of the bug report helps developers fix the bug quicker. However, even with this information available, the fields displayed in the bug report for the end users to complete (e.g. summary of the bug, description, attachments) are standard across all the different issue tracking systems.

Having the optimal bug report model/design for end users to complete would be beneficial for both the software development team, as well as the overall costs incurred by the company producing the application. Therefore, the main goal of this proposed project is to create a tool that can automatically generate the optimal bug report model for any specific project taking into account the characteristics of that project or the team behind it (e.g. size of the development team, technologies used, mobile/desktop application). Even though there is state-of-the-art research in this field, such as the work of Zimmermann et al.[5], they only look into what can be done and propose only advice to issue tracking systems developers. However, our work goes one step further and we want to conduct the first research in this area that actually gives the community a tool which can automatically fix this issue.

Key Ideas in a Nutshell

"To automatically create a bug report model tailored specifically for a project, taking into account the whole context of the software as well as the developers behind it."

The goal of the project is to create a tool which, given as input details regarding the project and the developers working on it, can automatically generate a bug report model that will increase the efficiency of the team as well as reduce the costs of the organization. Our proposed approach is to, firstly, determine what components determine the time-to-fix period (i.e. time spent between opening a ticket and closing it) for any given project using both statistical analysis as well as the data already collected in a publicly available appendix [2]. Then, in order to make our tool take into account all the complex characteristics of developing software, we want to both look at how the team of developers is managed and how the work is distributed, as well as how the developers interact with the end users reporting the bugs.

The *main benefit* of our project is that, through fixing a still unresolved issue in the software engineering field, it will improve three key aspects:

- reduce the overall costs of the organization developing the software;
- improve the time-to-fix window for all bug reports, thus reducing maintenance effort;
- reduce the communication friction between end users and developers because having a tailored bug report model will not require engineers constantly wasting time on asking for more details from the users reporting the bug.

Example: The UK government's Health Digital Services department (as well as many other departments) is using Jira as an issue tracking system [4]. If our proposed application would be used for their project, the developers could streamline the bug reports in a timely manner and increase their productivity, while also decreasing the overall costs for the government.

Research Objectives
State of the Art
Intended Solutions
Innovative Aspects

2 Methodology

...(WP1)
...(WP2)
...(WP3)

3 Measurable Outcomes

4 Impact and National Importance

References

[1] N. Bettenburg, S. Just, A. Schröter, C. Weiss, R. Premraj, and T. Zimmermann. What makes a good bug report? pages 308–318, 2008.

[2] S. Breu, R. Premraj, J. Sillito, and T. Zimmermann. Appendix to information needs in bug reports technical report. Technical report, October 2009.

[3] Cybernetica. X-road - a secure data exchange layer for building e-governments. Technical report, Mäealuse 2/1, Tallinn 12618, Estonia, 2014.

[4] U. Government. Health digital services jira.

[5] T. Zimmermann, R. Premraj, J. Sillito, and S. Breu. Improving bug tracking systems. In *Software Engineering-Companion Volume, 2009. ICSE-Companion 2009. 31st International Conference on*, pages 247–250. IEEE, 2009.