A Project on Software Defect Prevention at Commit-Time: A Success Story of University-Industry Research Collaboration

Wahab Hamou-Lhadj
ECE, Concordia University
Montréal, QC, Canada
abdelw@ece.concordia.ca

Mathieu Nayrolles
La Forge Research Lab, Ubisoft
Montréal, QC, Canada
mathieu.nayrolles@ubisoft.com

ABSTRACT

In this talk, we describe a research collaboration project between Concordia University and Ubisoft. The project consists of investigating techniques for defect prevention at commit-time for increased software quality. The outcome of this project is a tool called CLEVER (Combining Levels of Bug Prevention and Resolution techniques) that uses machine learning to detect programmers' buggy code before reaching the central code repository. CLEVER is currently being deployed at Ubisoft, to be used by thousands of developers. In this talk, we discuss the research problem, the project organization, the design of CLEVER, and the lessons learned.

KEYWORDS

University-Industry Research Project, Bug Prevention at Commit-Time, Machine Learning, Software Maintenance and Evolution.

1 INTRODUCTION

Software maintenance tasks continue to be challenging and costly [1], which explains the growing interest in industry-scale techniques for the detection and prevention of software defects. This is valid for most software organizations including Ubisoft, one of the world's leading video game development companies. The company specializes in the design and implementation of high-budget video games such as Prince of Persia, Far Cry and Assassin's Creed, and is heavily invested in software development and maintenance tasks. To continu its expansion with more and bigger games, while preserving quality, Ubisoft embarked on a research project in collaboration with Concordia University to explore software quality control techniques that can detect or even prevent the insertion of bugs, preferably, before system modifications reach the central software repository, i.e., at commit-time.

The project should achieve a number of requirements. The techniques must operate at commit-time, embedded within Ubisoft's code versioning systems. This requirement ensures that the accompanying tool fits well with the developers' workflow, eliminating the need to download and install external tools, which typically requires extensive settings and a high learning curve. The new tool should also be scalable to work with Ubisoft complex ecosystem, constituted of software systems that are highly coupled containing millions of files and commits, developed and maintained by more than 8,000 developers scattered across 29 locations in six continents.

With these requirements in mind, the research team at Concordia started working with Ubisoft software developers to understand better the industrial context and the type of systems that are used. Together, we defined the scope of the project and formed a team that includes students and developers from Ubisoft. The lead student was assigned to work on Ubisoft premises on a full-time basis during the project. The project lasted approximately eight months. Regular meetings were held to follow progress and make the necessary adjustments.

2 THE CLEVER SOLUTION

After reviewing the literature and analyzing Ubisoft systems, we came up with a two-phase approach for detecting risky commits that we call CLEVER (Combining Levels of Bug Prevention and Resolution techniques). The first phase consists of building a metric-based model to assess the likelihood that an incoming commit is risky or not. This is similar to existing work (see [2]). In this phase, we used Commit-guru, a tool proposed by Rosen et al. [2]. The next phase, which is the main novelty of CLEVER, relies on clone detection to compare code blocks extracted from suspicious risky commits, detected in the first phase, with those of known historical fault-introducing commits.

The second phase works as follows: for each commits that is detected as suspicious using a metric-based model, we extract the corresponding code block and compare it to the code blocks of a database of historical defect-introducing commits that we built by mining commits of Ubisoft systems. If there is a match, then the new commit is flagged as suspicious. To compare the extracted blocks to the ones in the database, we resort to clone detection techniques, more specifically, text-based clone detection techniques. More precisely, we use NICAD clone detector because it is freely available and has shown to perform well [3].

The problem with the current implementation of NICAD is that it only considers complete Java, C#, and C files. We improved NICAD to process blocks that come from commit-diffs. This is because the current version of NICAD can only process syntactically correct code and commit-diffs are, by definition, snippets that represent modified regions of a given set of files. By reusing NICAD, CLEVER can detect Types 3 software clones [3]. Type 3 clones contain added or deleted code statements, which make them suitable for comparing commit code blocks. Once a commit is flagged as risky, CLEVER goes one step further to

retrieve the commits that were used to fix the matching commits (the ones in the database) and display them to developers. This feature was well received by Ubisoft developers because it allows them to explore existing solutions, developed by other teams.

In addition, unlike existing work in the literature, CLEVER detects risky commits across systems, meaning that it does not only compare incoming commits to the commits of the same system, but also to those belonging to other systems that share common dependencies. This is a powerful feature because it takes into accounts the context of Ubisoft where systems are highly coupled and tend to have many dependencies, making them vulnerable to the same faults. This feature also adds value to CLEVER because it provides developers with the opportunity to share code written for other systems by completely different teams. The long term objective is to have CLEVER as a tool where developers, across multiple teams, can share fixes and experiences.

We tested CLEVER on 12 major Ubisoft systems. The results show that CLEVER can detect risky commits with 79% precision and 65% recall. In addition, a user study, conducted with Ubisoft developers, showed that 66.7% of CLEVER-proposed fixes were accepted by Ubisoft software developers, making CLEVER a simple and yet powerful approach for the detection and resolution of risky commits.

CLEVER is currently being deployed at Ubisoft, to be made available to thousands of developers across various divisions. The research team will provide training to developers on how to use the new tool. In addition, Ubisoft developed an instructional video to support the launch of CLEVER and raise awareness about the tool. The research team continues to work with Ubisoft to monitor the use of CLEVER at Ubisoft.

3 LESSONS LEARNED

The lessons learned through this industrial collaboration are numerous. In this talk, we discuss the most important ones that are summarized in what follows:

Deep understanding of the project requirements: Throughout the design of CLEVER, it was important to have a close collaboration between the research team and the Ubisoft developers. This allowed the research team to understand well the requirements of the project. Through this collaboration, both the research and development teams quickly realized that existing work in the literature was not sufficient to address the project's requirements. In addition, existing studies were mainly tested using open source systems, which may be quite different in structure and size from large industrial systems. In our case, we found that a deep understanding of Ubisoft ecosystem was an important enabler for many decisions we made in this project including the fact that CLEVER operates on multiple systems and that it uses a two-phase mechanism. It was also important to come up with a solution that integrates well with the workflow of Ubisoft developers. This required the development of CLEVER in a way it integrates well with the entire suite of Ubisoft's version control systems. The key lesson here is to understand well the requirements of a project and its complexity.

Focusing in the Beginning on Low-Hanging Fruits: Low-hanging fruits are quick fixes and solutions. We found that it is a good idea to showcase some quick wins early in the project to show the potential of the proposed solutions. In the beginning of the project, we applied the two-phase process of CLEVER to some small systems with a reasonable number of commits. We showed that the approach improved over the use of metrics alone. We also showed that CLEVER was able to make suggestions on how to fix the detected risky commits. This encouraged us to continue on this path and explore additional features. We continued to follow an iterative and incremental process throughout the project where knowledge transition between the University and Ubisoft teams is done on a regular basis.

Building a Strong Technical Team: Working on industrial projects requires all sort of technical skills including programming in various programming languages, the use of tools, tool integration, etc. The strong technical skills of the lead student of this project were instrumental in the success of this project. It should be noted that Ubisoft systems are programmed using different languages, which complicated the code matching phase of CLEVER. In addition, Ubisoft uses multiple bug management and version control systems. Downloading, processing, and manipulating commits from various environment requires excellent technical abilities.

Communicating effectively: During the development of CLEVER, we needed to constantly communicate the steps of our research to developers and project owners. It was important to adopt a communication strategy suitable to each stakeholder. For example, in our meetings with management, we focused more on the ability of CLEVER to improve code quality and reduce maintenance costs instead of the technical details of the proposed approach. Developers, on the other hand, were interested in the potential of CLEVER and its integration with their work environment.

4 CONCLUSION

In this talk, we share our experience conducting a research project at Ubisoft. The project consists of developing techniques and a tool for detecting defects before they reach the code repository. Our approach, called CLEVER, achieves this in two phases using a combination of metric-based machine learning models and clone detection. CLEVER is being deployed at Ubisoft.

ACKNOWLEDGEMENTS

We would like to thank the teams at Ubisoft that participated in this project. We also acknowledge the role of NSERC for supporting partly this project.

REFERENCES

[1] Newman, M. Software errors cost us economy $59.5 billion annually. NIST Assesses Technical Needs of Industry to Improve Software-Testing, 2002.

[2] Rosen, C. Graw , B. Shihab, E.. Commit Guru: Analytics and Risk Prediction of Software Commits. In *Proceedings of the Joint Meeting on Foundations of Software Engineering* , (2015), 966–969.

[3] Cordy, J. R. and Roy, C. K. The NiCad Clone Detector. In *Proceedings of the international conference on program comprehension*, (2011), 219–220.