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
wahab.hamou-lhadj@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 automatically detect coding defects as programmers write code. The main novelty of CLEVER is that it relies on code matching techniques to detect coding mistakes based on a database of historical code defects found in multiple related projects. The tool also proposes fixes based on known patterns.

CCS CONCEPTS

• **Software and its engineering** → Software defect analysis

KEYWORDS

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

1 INTRODUCTION

Software maintenance tasks are known to be 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 continue its expansion with more and larger 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.

SER&IP'18, May 29, 2018, Gothenburg, Sweden

ACM ISBN 978-1-4503-5744-9/18/05…$15.00
https://doi.org/10.1145/3195546.3206423

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 require extensive settings and a high learning curve. The new tool should also be scalable to work with Ubisoft complex ecosystem, composed 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 one year. 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, which 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. For this phase, we adapted Commit-guru, a tool proposed by Rosen et al. [2, 3]. The next phase, which is the main novelty of CLEVER and the focus of this talk, relies on clone detection to compare code blocks extracted from suspicious commits, detected in the first phase, with millions of known fault-introducing commits that are saved in a database. This phase was based on earlier work of the research team, in which they developed techniques that use code matching to detect defects at commit-time with applications to open source systems.

In a nutshell, 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 risky. To compare the extracted blocks to the ones in the database, we use clone detection techniques, more specifically, text-based clone detection techniques. For this, we use a modified version of NICAD, a known clone detector [4]. NICAD is freely available and has shown to perform well. Another important feature of CLEVER is that it operates across multiple projects by comparing incoming commits to commits from other systems that share common dependencies. This is important because industrial systems, such as those at Ubisoft, 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, 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. An enhanced version of CLEVER is currently being deployed at Ubisoft and will be made available to thousands of developers across various divisions. A paper describing the technical aspects of CLEVER was accepted for publication at The Mining Software Repository Conference (MSR 2018).

3 LESSONS LEARNED

Many lessons learned were already reported in the MSR paper such as the need to understand the industrial context, leveraging an iterative and incremental process, communicating effectively, and estimating properly the effort and time it takes to develop production-level tools. In this talk, we will review these lessons and cover additional ones, mentioned below. We hope that this feedback would be useful to researchers who want to embark on a collaborative project with industry.

Understanding the benefits of the project to both parties: Understanding how the project benefits the company and the university helps both parties align their vision and work towards a common goal and set of objectives. From Ubisoft's perspective, the project provides sound mechanisms for building reliable systems. In addition, the time saved from detecting and fixing defects can be shifted to the development of new functionalities that add value to Ubisoft customers. For the university research team, the project provides an excellent opportunity for gaining a better understanding of the complexity of industrial systems and how research can provide effective and practical solutions. Also, working closely with software developers helps uncover the practical challenges they face within the company's context. Companies vary significantly in terms of culture, development processes, etc. Research effort should be directed to develop solutions that overcome these challenges, while taking into account the organizational context.

Focusing on low-hanging fruits in the beginning of the project: 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.

Building a strong technical team: Working on industrial projects requires all sort of technical skills including programming in various programming languages, use of tools, tool integration, etc. The strong technical skills of the lead student of this project were instrumental to 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.

Managing change: Any new initiate brings with it important changes to the way people work. Managing these changes from the beginning of the project increases the chances for tool adoption. To achieve this, we used a communication strategy that involved all the stakeholders including software developers and management to make sure that potential changes that CLEVER would bring are thoroughly and smoothly implemented, and that the benefits of change are long-lasting.

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. An enhanced version of CLEVER is being deployed at Ubisoft.

ACKNOWLEDGEMENTS

We thank the teams at Ubisoft for their participation in this project, and acknowledge the role of the Natural Science and Engineering Research Council of Canada (NSERC) for funding partly this project.

REFERENCES

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

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

[3] Y. Kamei, E. Shihab, B. Adams, A. Hassan, A. Mockus, A. Sinha, and N. Ubayashi. A Large-Scale Empirical Study of Just-In-Time Quality Assurance. In *IEEE Transactions on Software Engineering (TSE)*, 39(6), (2013), 757 - 773.

[4] J. R. Cordy and C. K. Roy. The NiCad Clone Detector. In *Proceedings of the International Conference on Program Comprehension*, (2011), 219–220.