Introduction to the Special Issue on Software Maintenance and Evolution Research

Keywords: Experiments and reproducibility; Bug root cause; Ecosystem dependency upgrades; Socialization of committers; Priority level of bug report; Rapid release and testing.

Introduction

The past decade has seen tremendous changes in the landscape of software engineering activities in industry and of software engineering research topics in academia. Software engineering activities used to be mostly "hidden" behind the closed doors of software industries. In the 1990's, the emergence of the open-source development model and movement revolutionized software development but also software engineering research. Nowadays, open-source software (OSS) development has become a major source of new libraries and programs, and more and more industries either use OSS or even release OSS themselves. Even large corporations, in which trade secrets were the norm, now consider releasing their source code as open-source, for example Microsoft with its Windows operating system [1].

The open-source revolution gave rise to the development of many collaborative development tools, including distributed version control systems (such as Git and its popular Web platforms GitHub and BitBucket), issue trackers (such as Bugzilla), and continuous integration tools (such as Travis). These tools now depend on one another, leading to the creation of so-called software ecosystems, sets of programs and libraries relying on one another. These ecosystems further feed the dependence on, and development of, OSS.

This *technical* revolution is accompanied by a *social* revolution, where software systems are no longer produced by single "craftsmen", like early games (even such famous games as Prince of Persia [2]) but involve large networks of collaborating developers, which are as much about social interactions as about technical development. Despite some outliers [3], OSS development is not reserved to "lonely geeks" anymore but, indeed, requires considerable social skills and interactions.

Yet, the creation of socio-technical software ecosystems is not without problems. Mostly, the numerous interdependencies among programs and libraries make it more difficult for developers to change their software by fear of hidden negative consequences on other programs and libraries. Conversely, dependent software programs may prefer to stay

dependent on older libraries because of the costs of upgrades and fear of unknown, hidden impacts.

In our fast-paced, technological society [4], lots of our daily activities depend upon technology, be it for work, education, or recreation. The open-source movement was seen as an opportunity to create better software but recent problems show that OSS also has its shortcomings [5] despite existing tools to track feature requests and bugs and prioritise work. More formal, agile processes are an answer to ease developers' work and help them release more rapidly new versions of their systems with corrected bugs and new features.

All these changes fostered the apparition of new research topics in academia and, more importantly, highlighted the need for sound, grounded, repeatable empirical studies. Researchers, observing the open-source revolution, want to study the impact of these changes and are in the enviable position to have easily accessible historical data of a "before" and an "after". Consequently, researchers can now perform interesting and relevant case studies, industrial comparisons, and (quasi-)controlled experiments and surveys. Until about a decade ago, it was possible to publish papers in software engineering presenting novel ideas with little or no evaluation. Nowadays, it has become very difficult (and undesirable) to publish papers without some well-grounded case studies and many papers even include full-fledged empirical studies. Replications are also finally becoming to be accepted as necessary to the advance of software engineering, as in other sciences [6], although there are still some misunderstandings.

The Papers in This Issue

The papers in this special issue of Springer's journal of Empirical Software Engineering, which are significant extensions of conference papers published in the proceedings of the 29th IEEE International Conference on Software Maintenance (ICSM'13), reflect the open-source revolution and its impacts on both industry and academia. They propose empirical studies in timely topics directly related to the open-source movement: Experiments and reproducibility; Bug root cause; Ecosystem dependency upgrades; Socialization of committers; Priority levels of bug reports; Rapid release and testing.

Bogdan Dit, Evan Moritz, Mario Linares-Vásquez, and Denys Poshyvanyk, in "Supporting and Accelerating Reproducible Empirical Research in Software Evolution and Maintenance using TraceLab Component Library", address the problem of experiment reproducibility in software maintenance. They describe a possible long-term solution for ensuring that future experiments will be reproducible and extensible. They introduce components dedicated to software maintenance approaches and integrate them in TraceLab. They also present experiments realised using these components. The goal of experiments and components is to create a body of actionable knowledge that would (1) facilitate future research and (2) allow the research community to contribute to it as well.

Tien-Duy B. Le and David Lo, in “Should I Follow This Fault Localization Tool's Output? Automated Prediction of Fault Localization Effectiveness”, highlight that the root causes of many bugs are often low in the ordered list of entities provided by fault localisation tools. This low position causes developers to distrust fault localization tools. Consequently, they build an oracle to predict whether the output of a fault localization tool can be trusted or not. Their experiments demonstrate that they can predict the effectiveness of nine fault localization tools with a high precision, recall, and F-measure. The obtained results indicate that many ineffective fault localization instances are identified correctly, while only few effective ones are identified wrongly.

Gabriele Bavota, Gerardo Canfora, Massimiliano Di Penta, Rocco Oliveto and Sebastiano Panichella, in “How the Apache Community Upgrades Dependencies: An Evolutionary Study” study the evolution of dependencies between projects in the Java subset of the Apache ecosystem, consisting of 147 projects, for a period of 14 years, resulting in 1,964 releases. The study results, which have been qualitatively confirmed by observations made by analyzing the developers' discussions, indicate that new releases of projects trigger upgrades when the new releases include major changes (e.g., new features/services) as well as large amount of bug fixes but not when the changes are considered “minor” by the dependents.

Mohammad Gharehyazie, Bogdan Vasilescu, Daryl Posnett and Vladimir Filkov, in “Predicting OSS Developer Initiation from Their Social Activities”, study social and technical interactions of software developers with their community and observe that social network metrics, in particular the amount of two-way communication in which a developer participates, are more significant predictors of one's likelihood to becoming a committer than her technical contributions. They also observe that one's level of socialization ramps up before the time of becoming a committer. After obtaining committer status, social behaviour tends to be more individualized: immediately after the initiation there is a notable social cooling-off period.

Yuan Tian, David Lo and Chengnian Sun, in “Automated Prediction of Bug Report Priority Using Multi-Factor Analysis”, propose an automated approach based on machine learning to recommend a priority level based on information available in bug reports. Their approach considers multiple factors, temporal, textual, author, related-report, severity, and product, which potentially affect the priority level of a bug report. They show that their approach outperforms baseline approaches in terms of average F-measure by a relative improvement of up to 209%.

Finally, Mika Mäntylä, Foutse Khomh, Bram Adams, Emelie Engstrom and Kai Petersen, in “On Rapid Releases and Software Testing”, start by providing a semi-systematic literature review, which shows that rapid releases are a prevalent industrial practice that are utilized even in some highly critical domains of software engineering. Consequently, they study the shift to rapid releases that occured in Firefox recently and find that rapid releases have a narrower test scope that enables a deeper investigation of the features and regressions with the highest risk. However, rapid releases make it more difficult to build

a large testing community, and they decrease test suite diversity and make testing more deadline-oriented. They triangulate their observation with a Mozilla QA engineer.

Where Will the Future Take Us?

The papers in this special issue address relevant research topics and provide timely, constructive results. However, as any good paper should, they also give rise to new questions. The open-source revolution was lead by independent developers contributing their own time and effort for the greater good. Yet, the quality of the resulting programs and libraries led industry to take a keen interest in OSS development. Nowadays, many high-profile open-source projects are backed by industries. The impact of industrial support is not yet well-understood and raises technical, social and ethical concerns [7, 8].

Programs and libraries in software ecosystems depend on one-another and their dependence is generally considered as mutually beneficial. Yet, these dependencies create a complex web of relations, whose threads may propagate changes in unforeseen ways and with dramatic consequences. Consequently, developers may be reluctant to change their code, when fixing major bugs may require major design and–or architectural changes, by fear of the consequences of changes [9]. Researchers must study these changes and their consequences and help developers with tools and techniques to predict the consequences of changes.

Fear of the consequences of changes is not the only reason why developers may be unwilling to evolve their software. Rapid releasing gives less time to developers to study the real root cause of bugs. Higher priority bugs may be left aside because too complicated under time pressure.

We hope that these discussions and the many others in the papers in this special issue will give readers food for thought. They certainly advance the state-of-the-art in software engineering by providing thorough results on problems related to the social and technical revolutions brought by the emergence of software ecosystems: experiments and reproducibility; bug root cause; dependencies upgrades; socialization of committers; priority level of bug report; and, rapid release and testing.

We wish you a happy reading!
Yann-Gaël Guéhéneuc and Tom Mens
ICSM'13 Program Co-chairs

References

[1] http://arstechnica.com/information-technology/2015/04/open-source-windows-is-definitely-possible-but-dont-hold-your-breath/, last access on 2015/04/22.

[2] http://en.wikipedia.org/wiki/Jordan_Mechner, last access on 2015/04/22.

[3] http://arstechnica.com/business/2015/01/linus-torvalds-on-why-he-isnt-nice-i-dont-care-about-you/, last access on 2015/04/22.

[4] Hartmut Rosa and William E. Scheuerman (Eds.) ; High-speed Society: Social Acceleration, Power, and Modernity ; Penn State University Press, December 2006.

[5] http://heartbleed.com/, last access on 2015/04/22.

[6] Jane Cleland-Huang ; On Whose Shoulders? ; Keynote at the 22nd IEEE International Conference on Software Analysis, Evolution, and Reengineering, IEEE Computer Society Press, March 2015. Available at: http://saner.soccerlab.polymtl.ca/lib/exe/fetch.php?media=jane_cleland-huang_-_on_whose_shoulder.pdf, last access on 2015/04/22.

[7] Bertrand Meyer ; The Ethics of Free Software ; Dr. Dobb's Journal, UBM Tech, March 2000. Available at: http://www.drdobbs.com/the-ethics-of-free-software/, last access on 2015/04/22.

[8] Monty Montgomery ; A rebuttal to Meyer's "The Ethics of Free Software" ; Advogato, May 2000. Available at: http://www.advogato.org/article/94.html, last access on 2015/04/22.

[9] http://agileotter.blogspot.ca/2009/01/fear-of-changing-code.html, last access on 2015/04/22.