A Literature Survey of Software Analytics

Moritz Beller, IN4334 2018 TU Delft

2018-09-20

Contents

Chapter 1

Preamble

The book you see in front of you is the outcome of an eight week seminar run by the Software Engineering Research Group (SERG) at TU Delft. We have split up the novel area of Software Analytics into several sub topics. Every chapter addresses one such sub-topic of Software Analytics and is the outcome of a systematic literature review a laborious team of 3-4 students performed.

With this book, we hope to structure the new field of Software Analytics and show how it is related to many long existing research fields.

Moritz Beller

1.1 License

Chapter 2

A contemporary view on Software Analytics

2.1 What is Software Analytics?

2.2 A list of Software Analytics Sub-Topics

Chapter 3

Build analytics

3.1 Background

When building a project from source code to executables everything should go smoothly. This is not always the case, a build can break for several reasons. This chapter will give an overview of research done on build configurations and continuous integration.

3.2 Research Questions

RQ1 What is the current state of the art in the field of build analytics? **RQ2** What is the current state of practice in the field of build analytics? **RQ3** What future research can we expect in the field of build analytics?

3.3 Search Strategy

Using the initial seed consisting of Bird and Zimmermann (2017), Beller et al. (2017a), Rausch et al. (2017), Beller et al. (2017b), Pinto and Rebouças (2018), Zhao et al. (2017), Widder et al. (2018) and Hilton et al. (2016) we used references to find new papers to analyze.

3.4 Study Selection

Through this we found the following papers

3.5 Summary of papers

3.5.1 Bird and Zimmermann (2017)

Initial Seed

This is a US patent grant for a method of predicting software build errors. This patent is owned by Microsoft. Using logistic regression a prediction can be made on the probability of a build failing. Using this method build errors can be better anticipated, which decreases the time until the build works again.

3.5.2 Beller et al. (2017a)

Initial Seed

This paper explores data from Travis CI[1] on a large scale by analyzing 2,640,825 build logs of Java and Ruby builds. It uses TravisTorrent as a data source. It is found that the number one reason for failing builds it test failure. It also explores differences in testing between Java and Ruby.

3.5.3 Rausch et al. (2017)

Initial Seed

A stuy on the build results of 14 open source software Java projects. It is similar to Beller et al. (2017a), albeit on a smaller scale. It does go more in depth on the result and changes over time.

3.5.4 Beller et al. (2017b)

Initial Seed

This paper introduces TravisTorrent, a dataset containing analyzed builds from more than 1,000 projects. This data is freely downloadable from the internet. It uses GHTorrent to link the information from travis to commits on GitHub.

3.5.5 Pinto and Rebouças (2018)

Initial Seed

This paper is a survey amongst Travis CI users. It found that users are not sure whether a job failure represents a failure or not, that inadequate testing is the most common (technical) reason for build breakage and that people feel that there is a false sense of confidence when blindly trusing tests.

3.5.6 Zhao et al. (2017)

Initial Seed

This paper analyzed approximately 160,000 projects written in seven different programming languages. It notes that adoption of CI is often part of a reorganization. It collected information on the differences before and after adoption of CI. There is also a survey amongst developers to learn about their experiences in adopting Travis CI.

3.5.7 Widder et al. (2018)

Initial Seed

This paper analyzes what factors have impact on abandonment of Travis. They find that increased build complexity reduces the chance of abandonment, but larger projects abandon at a higher rate and that a project's language has significant but varying effect. A surprising result is that metrics of configuration attempts and knowledge dispersion in the project don't affect the rate of abandonment.

[1]See https://travis-ci.org

3.5.8 Hilton et al. (2016)

Initial Seed

This paper explores which CI system developers use, how developers use CI and why developers use CI. For this it analyzes data from Github, Travis CI and it conducts a developer survey. It finds that projects using CI release twice as often, accept pull requests faster and have developers who are less worried about breaking the build.

3.5.9 Vassallo et al. (2017)

References Beller et al. (2017a)

This paper discusses the difference in failures on continuous integration between open source software (OSS) and industrial software projects. For this 349 Java OSS projects and 418 project from ING Nederland, a financial organization.

Using cluser analysis it was observed that both kinds of projects share similar build failures, but in other cases very different patterns emerge.

3.5.10 Hassan and Wang (2018)

References Beller et al. (2017b)

This paper uses TravisTorrent (Beller et al. (2017b)) to show that 22% of code commits include changes in build script files to keep the build working or to fix the build.

In the paper a tool is proposed to automatically fix build failures based on previous changes.

3.5.11 Vassallo et al. (2018)

References Beller et al. (2017a), Rausch et al. (2017)

This paper proposes a tool called BART to help developers fix build errors. This tool eliminates the need to browse error logs which can be very long by generating a summary of the failure with useful information.

3.5.12 Zampetti et al. (2017)

Referenced by Vassallo et al. (2018)

This paper studies the usage of static analysis tools in 20 Java open source software projects hosted on GitHub and using Travic CI as continuous integration infrastructure. There is investigated which tools are being used, what types of issues make the build fail or raise warnings and how is responded to broken builds.

3.6 What is the current state of practice in the field of build analytics?

Chapter 4

Sample Sub-Topic

This is an example for the deliverable every group works on. Every group works on one independent chapter (starting as one Rmd file).

4.1 Motivation

A short introduction about why the topic you are working on is interesting.

The RQs that everyone should be aiming at are:

- **RQ1** Current state of the art in software analytics for *your topic* :
  - Topics that are being explored
  - Research methods, tools and datasets being used
  - Main research findings, aggregated
- **RQ2** Current state of practice in software analytics for *your topic* :
  - Tools and companies creating / employing them
  - Case studies and their findings
- **RQ3** Open challenges and future research required

4.2 Research protocol

Here, you describe the details of applying Kitchenham's survey method for your topic, including search queries, fact extraction, coding process and an intial groupping of the papers that you will be analyzing.

4.3 Answers

Aggregated answers to the RQs, per RQ. You need:

- For **RQ1**
  - Topics that are being explored
  - Research methods, tools and datasets being used
  - Main research findings, aggregated
- For **RQ2** :
  - Tools and companies creating / employing them
  - Case studies and their findings

- For **RQ3**:
  - List of challenges
  - An aggregated set of open research items, as described in the papers
  - Research questions that emerge from the synthesis of the presented works

Chapter 5

Final Words

We have finished a nice book on Software Analytics.

Bibliography

Beller, M., Gousios, G., and Zaidman, A. (2017a). Oops, my tests broke the build: An explorative analysis of travis ci with github. In *Mining Software Repositories (MSR), 2017 IEEE/ACM 14th International Conference on*, pages 356–367. IEEE.

Beller, M., Gousios, G., and Zaidman, A. (2017b). Travistorrent: Synthesizing travis ci and github for full-stack research on continuous integration. In *Proceedings of the 14th International Conference on Mining Software Repositories*, pages 447–450. IEEE press.

Bird, C. and Zimmermann, T. (2017). Predicting software build errors. US Patent 9,542,176.

Hassan, F. and Wang, X. (2018). Hirebuild: an automatic approach to history-driven repair of build scripts. In *Proceedings of the 40th International Conference on Software Engineering*, pages 1078–1089. ACM.

Hilton, M., Tunnell, T., Huang, K., Marinov, D., and Dig, D. (2016). Usage, costs, and benefits of continuous integration in open-source projects. In *Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering*, pages 426–437. ACM.

Pinto, G. and Rebouças, F. C. R. B. M. (2018). Work practices and challenges in continuous integration: A survey with travis ci users.

Rausch, T., Hummer, W., Leitner, P., and Schulte, S. (2017). An empirical analysis of build failures in the continuous integration workflows of java-based open-source software. In *Proceedings of the 14th International Conference on Mining Software Repositories*, pages 345–355. IEEE Press.

Vassallo, C., Proksch, S., Zemp, T., and Gall, H. C. (2018). Un-break my build: Assisting developers with build repair hints.

Vassallo, C., Schermann, G., Zampetti, F., Romano, D., Leitner, P., Zaidman, A., Di Penta, M., and Panichella, S. (2017). A tale of ci build failures: An open source and a financial organization perspective. In *Software Maintenance and Evolution (ICSME), 2017 IEEE International Conference on*, pages 183–193. IEEE.

Widder, D. G., Hilton, M., Kästner, C., and Vasilescu, B. (2018). I'm leaving you, travis: A continuous integration breakup story.

Zampetti, F., Scalabrino, S., Oliveto, R., Canfora, G., and Di Penta, M. (2017). How open source projects use static code analysis tools in continuous integration pipelines. In *Mining Software Repositories (MSR), 2017 IEEE/ACM 14th International Conference on*, pages 334–344. IEEE.

Zhao, Y., Serebrenik, A., Zhou, Y., Filkov, V., and Vasilescu, B. (2017). The impact of continuous integration on other software development practices: a large-scale empirical study. In *Proceedings of the 32nd IEEE/ACM International Conference on Automated Software Engineering*, pages 60–71. IEEE Press.