Empirical aspects of FLOSS ecosystems

November 16, 2017

Chapter 1

Types of data in ecosystems

Barbara

FLOSS ecosystems produce big data of different types coming from various sources such as development artefacts and operations logs. Such data can be used to describe properties of the ecosystem that produced it and the environment in which it runs by, for example, modelling the reliability of the ecosystem's components with defect data. Unfortunately, the information such data carries may often be incomplete and sometimes totally missing. The one major reason is that some types of data are actually unobservable or unaccessible as they are hard to collect or analyse (e.g., data from chats [Zou and Song, 2016]) or have specific access constraints (privacy or proprietary rights). On the other hand, partial data of a FLOSS ecosystem can be potentially augmented by observing a phenomenon and gather data across the whole ecosystem. For example, the activity of a developer on a single FLOSS project can be measured with more accuracy by observing the entire ecosystem the original project belongs to: developers might not have contributed to a project since they have worked in other projects of the same ecosystem. Thus, observing the entire ecosystem gives a better indication of if a person is still active, has moved from one project to another, or is inactive [**?**]. Thus, what are the major sources of data generated by today FLOSS ecosystems?

1.1 Sources of data

Data of FLOSS ecosystems are typically collected from and about the development process. Such data carries information of different qualities of the

software process and product. Being tidy connected with process and products, the types of relevant data and the techniques to obtain them evolve as new practices, technologies or development needs appear. For example, on-line controlled experiments via A/B testing have been recently integrated in the development for"continuous software delivery" - the ability to incorporate seamlessly changes into production - as decision making instrument for fast delivering of high quality products [Kevic et al., 2017]. On the other hand, process and product data can also help produce tools or support new methods of development. For example, data stored in code repositories, can be exploited to build recommender systems for developers (e.g., [**?**]). Usage and operations' data can produce added value in modern development processes as well. For example, in "DevOps" - a recently-born development paradigm that aims at fostering the integration between software development and operations - systems and use logs can be exploited to provide feedback for the development process (e.g., again as recommenders). Worth noticing that such development paradigm not only facilitates the communication between development and operations that traditionally are kept separated, but it also produces new data types by combining existing data (e.g., log events and source code data [**?**]). Operations data are anyway less accessible though as they concern the internal processes of organisations. Another promising sources of data can potentially come from new distributed contexts like the ones compliant with Industry 4.0 standards, where software / system data can be complemented with data coming from sensors and actuators in cloud or edge networks.

Finally, with all this abundance of data and data types, how far can we go? By now, two new questions become impelling: What recommendations can we give to ecosystems like GitHub to collect such new data? What is the role of qualitative data and how can we integrate it automatically?

Chapter 2

Data provenance

Sean P. Goggins, Ph.D

Challenges and Solutions for Data Provenance Management for Understanding Open Source Ecosystems

2.1 Introduction

Humans are not great at remembering and recording work they do every day. A machinist is unlikely to recall and recount each step involved in setting up a machine to build a new, custom part. Similarly, scientists who examine electronic trace data (logs from human computer interaction) do not always track specifically how they fill in missing data, standardize categories, clean, reshape and reduce teh data they use [**?**].

The Mining Software Respositories, open collaboration data exchange and GenBank communities provide examples open source software ecosystem researchers may be able to utilize as exemplars to emulate for tracking data provenance. Data provenance is a record of how data was collected, processed, altered and reshaped to provide the foundation for analysis and findings reported in academic papers or professional, industrial project dashboards. #TODO describe prior work looking at data provenance across fields

Reporting of the provenance of data used for research and practice examining open source software is inconsistent. ##TODO Describe examples and make the gap clear

In this chapter we enumerate the challenges, solutions and 2 exemplars for managing and documenting data provenance in open source software research.

2.2 State of the Art

2.3 Challenges in Research Practice

2.4 Methods for Addressing Challenges

2.5 The Open Collaboration Data Exchange Manifest Structure

2.5.1 The SPDX Example in the Linux Foundation

2.5.2 OCDX Current State

2.6 Data Provenance Technical Pipeline Examples From Social Computing Research

2.7 Data Provenance Technical Pipeline Examples From The Mining Software Repositories Community

2.8 Next Steps

2.9 Conclusion

Chapter 3

Data acquisition

Sean Goggins

3.1 Introduction

If you give a mouse a cookie, she is going to ask you for a glass of milk [Numeroff, 1985]. Similarly, if one provides an analyst, stakeholder or researcher with data representing one aspect, section or dimension of a phenomena, they are going to ask for a glass of milk. Or additional perspectives. This is the experience of having too much data, and possibly too much access to data.

Data acquisition in the land of plenty has two contrasting cases; or perhaps we say that data scarcity comes in two forms. Scarcity can include cases where there are not available repositories of research data. For example, while there is great interest in examining diversity and inclusion in open source software, our usual software repositories do not capture essential demographic information. Scarcity can also be related to access. Perhaps there is a privileged data repository that you would like access to, but are unable to obtain. In this case you are "locked out" [Goggins et al., 2013].

The Mining Software Respositories, open collaboration data exchange and GenBank communities provide examples open source software ecosystem researchers may be able to utilize as exemplars to emulate for tracking data provenance. Data provenance is

a record of how data was collected, processed, altered and reshaped to provide the foundation for analysis and findings reported in academic papers or professional, industrial project dashboards. #TODO describe prior work looking at data provenance across fields

Reporting of the provenance of data used for research and practice examining open source software is inconsistent. ##TODO Describe examples and make the gap clear

In this chapter we enumerate the challenges, solutions and 2 exemplars for managing and documenting data provenance in open source software research.

3.2 State of the Art

3.3 Challenges in Research Practice

3.4 Methods for Addressing Challenges

3.5 The Open Collaboration Data Exchange Manifest Structure

3.5.1 The SPDX Example in the Linux Foundation

3.5.2 OCDX Current State

3.6 Data Provenance Technical Pipeline Examples From Social Computing Research

3.7 Data Provenance Technical Pipeline Examples From The Mining Software Repositories Community

3.8 Next Steps

3.9 Conclusion

Chapter 4

Experimenting With Data: Quasi-Experimental Methods in Online Environments

Bogdan Vasilescu

Chapter 5

Data Processing

Georgios Gousios

Software engineering is an exceedingly data abundant activity. Nearly all artifacts of a software development project, both static, such as source code, software repositories and issues and dynamic, such as run-time logs and user activity streams, contain information valuable for understanding and optimising both the software products and the processes that created them. Unfortunately, modern organisations often do not utilise this wealth of information as a feedback and decision support instrument. Rather, teams adopt new software development practices and employ the latest and greatest technology, without questioning the results of their decisions. Consequently, this leads to decisions that are often unnecessarily suboptimal or downright wrong [Strigini, 1996].

5.1 Streaming Analytics

Despite the existence of tools and methods for extracting data from software development processes, products and ecosystems, one key aspect is missing: real-time operation. To compensate for this deficiency, we need to come up with tools and methods to collect, query, aggregate, and summarise ecosystem data as streams, enabling software practitioners to use analytics as a core feedback loop. In the context of software engineering, we expect that streaming software analytics will enable software practitioners to move beyond information toward actionable insight, hence allowing them for an increase of software process and system technical quality.

5.2 A research agenda

With the proposed research, we aim at utilising the wealth of information produced by distinct software development artifacts in order to enable software practitioners to use software analytics as a feedback and decision support instrument. Realising streaming software analytics, therefore requires an understanding of distinct stakeholder information needs, and decisions they influence. But more importantly, we strive to understand how those needs map to software analyses. We believe that community co-ordinated efforts can help realizing the streaming software analytics vision in at least the following ways:

Requirements for analytics Researchers need to identify the stackeholders' information needs by means of qualitative research. Early works by Buse and Zimmerman [Buse and Zimmermann, 2012] and Begel and Zimmerman [Begel and Zimmermann, 2013] are on this direction, but need to be revisited in the light of real-time analytics and instant feedback.

Infrastructure work Developing analytics pipelines is a complex and error prone task. It is however an area of intense competition (both academic and industrial) and there is ample room for improvement. Software engineering researchers should work together with researchers in other fields (e.g. databases, programming languages) to tailor existing analytics systems to software engineering requirements.

Feedback-driven ecosystem evolution Software engineering development methods (e.g. Scrum) are currently based on much folklore and little evidence. As researchers, we should aim to enable feedback loops within the software engineering practice. Developers should be able to easily execute experiments (e.g. A/B tests), collect and correlate the results of applying specific design and development decisions and their outcomes. This way, teams and organizations can organically evolve their tooling and optimize their processes based on data-driven decisions rather than black-box methodologies.

Chapter 6

Data quality

Mei

Chapter 7

Privacy and Ethics

Georgios Gousios

Bibliography

[Begel and Zimmermann, 2013] Begel, A. and Zimmermann, T. (2013). Analyze this! 145 questions for data scientists in software engineering. Technical Report MSR-TR-2013-111.

[Buse and Zimmermann, 2012] Buse, R. P. L. and Zimmermann, T. (2012). Information needs for software development analytics. In *Proceedings of the 34th International Conference on Software Engineering*, ICSE '12, pages 987–996, Piscataway, NJ, USA. IEEE Press.

[Goggins et al., 2013] Goggins, S. P., Mascaro, C., and Valetto, G. (2013). Group informatics: A methodological approach and ontology for sociotechnical group research. *Journal of the American Society for Information Science and Technology*, 64(3):516–539.

[Kevic et al., 2017] Kevic, K., Murphy, B., Williams, L., and Beckmann, J. (2017). Characterizing experimentation in continuous deployment: A case study on bing. In *Proceedings of the 39th International Conference on Software Engineering: Software Engineering in Practice Track*, ICSE-SEIP '17, pages 123–132, Piscataway, NJ, USA. IEEE Press.

[Numeroff, 1985] Numeroff, L. J. (1985). *If you Give a Mouse a Cookie*. Harper Collins.

[Strigini, 1996] Strigini, L. (1996). Limiting the dangers of intuitive decision making. *IEEE Softw.*, 13(1):101–103.

[Zou and Song, 2016] Zou, L. and Song, W. W. (2016). Lda-tm: A two-step approach to twitter topic data clustering. In *2016 IEEE International Conference on Cloud Computing and Big Data Analysis (ICCCBDA)*, pages 342–347.