Empirical aspects of FLOSS ecosystems

September 25, 2017

Chapter 1

Types of data in ecosystems

Barbara

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

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 [**?**].

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 [**?**] and Begel and Zimmerman [**?**] 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