Information visualization of software source code evolution to manage technical debt

Axel Ekwall
axelekw@kth.se
KTH Royal Institute of Technology
Stockholm, Sweden

ABSTRACT

A clear and well-documented LaTeX document is presented as an article formatted for publication by ACM in a conference proceedings or journal publication. Based on the "acmart" document class, this article presents and explains many of the common variations, as well as many of the formatting elements an author may use in the preparation of the documentation of their work.

KEYWORDS

information visualization, technical debt, software evolution

1 INTRODUCTION

In recent years society has undergone a digital transformation and our everyday life depends on digital technology more than ever before. This development relies on an ever-growing collection of software systems and services that span our society and connect our lives. With the increasing size and scope of these software projects, and the growing numbers of software developers working in the same project simultaneously, the complexity of the source code can become a large problem. It is hard for stakeholders to get an overview of how the project in general, and more specifically the source code, evolves over time.

Today, it is common for software development teams to work according to an agile methodology [2] where requirements and solutions in a project are evolving over time in collaboration between developers, project managers and users. This process is conducted in an iterative cycle with continuous reflections and improvements on previous work.

In order to handle these iterative and fast paced change to the source code in a project, development teams are using a version control system to keep track of changes. Based on search trends on *Google*[1] displayed in figure 1, the most used tool for software version control is *Git*[2], a system where developers can commit incremental changes to the code and collaborate on different versions of the source code in parallel. All these small incremental changes are recorded and make up a rich source of information into the evolution of the code. However, this information is cumbersome to grasp and oversee and therefore often not used in an effective way.

In this context, where rapid continuous decision making and reflection is important, insights into the evolution of the source code and a shared understanding of the current state of the code can be valuable. [1]

Technical debt

A term commonly used to describe this type of issue which might arise in large projects is technical debt, describing the increasing cost of development over time in a poorly maintained software project. The term is a metaphor to financial debt in the sense that a development team might increase their technical debt by taking shortcuts in the development in order to save time in the short term, but this debt can be costly of not payed back in time by going back and refactoring parts of the code that might be sub-optimal.

Information visualization

In order to handle these iterative and fast paced change to the source code in a project, development teams are using a version control system to keep track of changes. One of the most used tools for software version control is git, a system where developers can commit incremental changes to the code and collaborate on different versions of the source code in parallel. All these small incremental changes are recorded and make up a rich source of information into the evolution

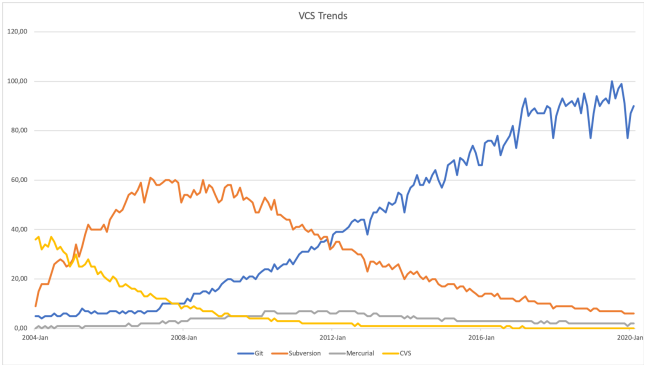

Figure 1: Popularity of version control systems 2004-2020

[1] https://trends.google.com
[2] https://git-scm.com/

of the code. However, this information is cumbersome to grasp and oversee and therefore often not used in an effective way.

2 THEORY AND RELATED RESEARCH

Previous research has been published investigating software evolution through different visualization techniques, examples include code-swarm, CVSscan and Software evolution storylines. However, during the last decade tools and workflows used by software development teams have evolved and more information is recorded with every change, allowing for rich analyses of the history and evolution of software source code.

The previous work mentioned contributed with valuable knowledge and in doing that also created new questions to be answered. In comparison to CVSscan which focuses on single files, this thesis will investigate a whole repository of source code to give a better overview and richer understanding about how the whole projects evolves. code-swarm and Software evolution storylines presented intuitive visual mappings for data points but targeted casual users and did not allow for interaction and deeper exploration of the data. In contrast, this study targets advanced users with domain knowledge and information about the context of the software project, allowing for a more advanced interactive visualization tool that presents complex information.

Research question

The goal of this study is to analyze and visualize this information in order to provide an overview and understanding of technical debt in a project and how it is changing with the evolution of the software and source code.

In doing that, his thesis aims to investigate whether information visualization can be used to empower software development teams to make better informed decisions about software architecture and source code maintenance on order to manage technical debt. Based on this goal, a prototype of such an information visualization tool will be created to answer the following research question.

Does a visualization of the evolution of software source code help software development teams manage technical debt by informing decision regarding maintenance and refactoring?

In order to answer the research question, the prototype will be evaluated with the following sub-questions:

- Does the prototype present enough information for developers to make decisions about whether to refactor and/or break up specific parts of the source code in order to pay of technical debt?
- Does the prototype present the information required in order to identify technical debt by highlighting "hotspots" with high rates of change and errors in the source code?

3 METHOD

4 RESULTS

5 DISCUSSION

6 CONCLUSION

ACKNOWLEDGMENTS

I want to thank Björn and Tino...

REFERENCES

[1] Thomas Ball, Jung-Min Kim, Adam A Porter, and Harvey P Siy. 1997. If your version control system could talk, Vol. 11.

[2] Orit Hazzan and Yael Dubinsky. 2014. The Agile Manifesto. In *Agile Anywhere*. Springer International Publishing, 9–14. https://doi.org/10.1007/978-3-319-10157-6_3