Information visualization of software source code evolution to manage technical debt

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

ABSTRACT

faucibus scelerisque eleifend donec pretium vulputate sapien nec sagittis aliquam malesuada bibendum arcu vitae elementum curabitur vitae nunc sed velit dignissim sodales ut eu sem integer vitae justo eget magna fermentum iaculis eu non diam phasellus vestibulum lorem sed risus ultricies tristique nulla aliquet enim tortor at auctor urna nunc id cursus metus aliquam eleifend mi in nulla posuere sollicitudin aliquam ultrices sagittis orci a scelerisque purus semper eget duis at tellus at urna condimentum mattis pellentesque id nibh tortor id aliquet lectus proin nibh nisl condimentum id venenatis a condimentum vitae sapien pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas sed tempus urna et pharetra pharetra massa massa ultricies mi quis hendrerit dolor magna eget est lorem ipsum dolor sit amet consectetur adipiscing elit pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas integer eget aliquet nibh praesent tristique magna sit amet purus gravida quis blandit turpis cursus in hac habitasse platea dictumst quisque sagittis purus sit amet volutpat consequat mauris nunc congue nisi vitae suscipit tellus mauris a diam maecenas sed enim ut sem viverra aliquet eget sit amet tellus cras adipiscing enim eu turpis egestas pretium aenean pharetra.

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, development planning and collboration becomes harder. //SOURCE A common method to handle these challanges is for software development teams to work according to an agile methodology [3] 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 version control systems to keep track of changes. This allowes developers to work on different versions of the software in parallel and *commit* their changes to the project in batches when new features are completed or bugs are fixed. 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, partly because it is not aggregated and presented as relevant metrics, but also because it is hidden from non-technical stakeholders who might not be able to retrieve the information from the VCS system. //SOURCE This makes it hard for stakeholders to get an overview of how the project in general, and more specifically the source code, evolves over time. 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]

Information visualization

One method to gain knowledge from a large set of complex information is to visualize it, or in other words, form a mental model of the information in order to understand it better. Information visualization tools can help this process by providing visual representations of the data. [4]

Technical debt

// Some info about the problem with 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, first described by Cunningham in 1992 [2], 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. However, 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.

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. codeswarm 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] Ward Cunningham. 1992. The WyCash portfolio management system. (1992), 2. https://doi.org/10.1145/157709.157715

[3] 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

[4] Robert Spence. 2014. *Information visualization.* Springer, New York.