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 [4] 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. [6] This practice has been conducted since long before modern technology was available to help and it is thus not nessecarily required to use technology to aid in this process. [3] However, the trend in recent years of collecting increasingly larger amounts of data //SOURCE creates new challenges in visualization and manual processing of data might not be an option. In this scenario, the capability of modern technology is a useful tool to help process, filter and map large data sets to visual representations in order to help the user to understand the data and form a mental model of the information. //SOURCE Digital visualization tools can also enable rich interaction and allow a user to explore the data step by step rather than getting overwhelmed by too much information at once. //SOURCE

Technical debt

When exploring source code evolution, and problems that commonly occur in long term software projects, a useful concept to utilize is *technical debt*. First described by Cunningham in 1992 [2], technical debt (TB) is a metaphor to financial debt and describes the common situation with increased development costs over time in software projects caused by poor software engineering practices. [8] In some cases, taking on TB can be a strategic short term desicion in order to reach a critical deadline in time. However, if not properly managed the debt can be costly by decreasing developemnt velocity and increasing maintenance costs. [5]

Research question

The goal of this paper is to design a visualization tool to explore the source code evolution in a software project in order to provide an overview and understanding of technical debt and how it is changing over time 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?

2 RELATED RESEARCH

Previous research has been published investigating software evolution through different visualization techniques. Bayer and Hassan developed "Evolution Storyboards" , 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.

3 METHOD

In order to investigate the research question, a methodology informed by Stoltermans Concept-Driven Design Research will be used. [7] Two design iterations will be conducted in 6 steps listed below over a period of 10 weeks.

First iteration:

(1) Generate concept.
(2) Explore design space and generate low-fi sketches.
(3) First critique session, focus group.

Second iteration:

(1) Develop design artefact, in this case an interactive prototype of the concept.
(2) Second critique session, user study with prepared tasks and interview.
(3) Concept contextualization and design refinement.

The critique sessions will be conducted with participants form the target group; developers, designers and project managers at Mentimeter AB. The critique session during the first iteration will be a focus group workshop where the participants will be presented with the concept idea and low-fi prototypes and sketches created during the first two steps. This session will be structured like group interview and workshop where the goal is to get early feedback on the overall concept design and help in choosing a direction for the second integration.

Based on the gathered feedback in the first critique session, an interactive high-fi prototype will be developed as a second iteration on the concept idea and design. When the defined concept and features are implemented in the prototype, a second critique session will be conducted. This session aims to gather more specific critique and will be structured as task- based user tests in a one-on-one setting where participants are asked to complete a set of tasks in the interactive prototype, followed by a semi-structured interview.

The data collected during the user tests will include both qualitative data gathered during the interviews and quantitative data from screen recordings in the form of time measurements from tasks and possibly quantitative results from structured parts of the interview.

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] Michael Friendly. 2008. A Brief History of Data Visualization. In *Handbook of Data Visualization*, Chun-houh Chen, Wolfgang Härdle, and Antony Unwin (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 15–56. https://doi.org/10.1007/978-3-540-33037-0_2

[4] 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

[5] Carolyn Seaman, Yuepu Guo, Nico Zazworka, Forrest Shull, Clemente Izurieta, Yuanfang Cai, and Antonio Vetro. 2012. Using technical debt data in decision making: Potential decision approaches. In *2012 Third International Workshop on Managing Technical Debt (MTD)*. IEEE, Zurich, Switzerland, 45–48. https://doi.org/10.1109/MTD.2012.6225999

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

[7] Erik Stolterman and Mikael Wiberg. 2010. Concept-Driven Interaction Design Research. *Human–Computer Interaction* 25, 2 (May 2010), 95–118. https://doi.org/10.1080/07370020903586696

[8] Edith Tom, Aybüke Aurum, and Richard Vidgen. 2013. An exploration of technical debt. *Journal of Systems and Software* 86, 6 (June 2013), 1498–1516. https://doi.org/10.1016/j.jss.2012.12.052