Project specification for “Information visualization of software source code evolution in order to manage technical debt”

Student

Axel Ekwall, axelekw@kth.se

Client

Mentimeter AB, mentimeter.com

Mentors at the client

- Niklas Ingvar, niklas@mentimeter.com
- Christofer Olaison, christofer.olaison@mentimeter.com

Supervisor

Björn Thuresson, thure@csc.kth.se

Examiner

Tino Weinkauf, weinkauf@kth.se

Date

2020-01-27

Table of Contents

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.

A term commonly used to describe this type of issue which might arise in large projects is *technical debt,* describing the increasing const 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 suboptimal.

Today, it is common for software development teams to work according to an *agile methodology* 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 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 in order to evaluate the amount of technical debt.

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

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?*

Expected results

The result of the study should be an interactive prototype to showcase and evaluate a design concept created through a concept driven design research approach. The design concept will include information visualizations of the evolution of software source code and relevant information in order to understand and explore the extent end location of technical debt in the project.

Further, the study should result in a set of design guidelines and principles which can be used to continue development of the prototype or by other projects within the area of visualization and software source code evolution. These guidelines may include knowledge such as usability principles, relevant visual encodings for source code data and visualization techniques that suites the context.

These results are of interest to software development teams working in large, long term software development projects where managing technical debt over time is crucial to keep development velocity high and cost under control. The results will also be a contribution to the research into technical debt and software evolution by providing knowledge about how visualization can be used as a tool to communicate in this space.

Relations to research and development

Previous research has been published investigating software evolution through different visualization techniques, examples include code_swarm (M. Ogawa & Kwan-Liu Ma, 2009), CVSscan (Voinea et al., n.d.) and Software evolution storylines (Michael Ogawa & Ma, 2010). 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.

The client's interest

Mentimeter AB is a Swedish software company delivering an interactive presentation software as a service to companies and institutions around the world. The software service is developed in-house and spans several long-lived source-code repositories maintained by a large team of software developers. Over time, the complexity of the source code has increased, making it hard for developers and project managers to overview the development process and understand when and how the code evolves.

The prototype tool to visualize software evolution developed as a part of this thesis can be a possible solution to this problem and add value to Mentimeter AB development process. With the design guidelines produced as a result of the research in this study the prototype could be further developed into a fully functioning tool and integrated into the agile workflow.

Method

In order to investigate the research question, a methodology informed by Stoltermans *Concept-Driven Design Research* will be used. (Stolterman & Wiberg, 2010) Two design iterations will be conducted in 6 steps listed below over a period of 10 weeks.

First iteration:

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

Second iteration:

4. Develop design artefact, in this case an interactive prototype of the concept (3-4 weeks)
5. Second critique session, user study with prepared tasks and interview (1-2 week)
6. Concept contextualization and design refinement (2 weeks)

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.

Evaluation

The evaluation of the results will include a thematic analysis of the qualitative data, where the recorded answers will be analyzed, and theme based on the answers formed as a way of structuring the result. The quantitative data will be summarized and presented as a secondary measure of the tool.

Scope

The main objective of this study is to evaluate a design concept, not developing a fully functioning visualization tool. Only the features and interactions necessary to properly evaluate the concept and generate design guidelines will be implemented in the prototype. Further, the visualizations included in the design concept will be based on the data available from commits to a git repository with source code. In order to keep the scope focused, the source code itself will not be analyzed in this project, even though this would be an interesting topic for a complimentary study.

Literature

As a part of this project, a literature study will be conducted in order to establish a background for the work in this thesis and to build on the knowledge presented in earlier work within the research area. The Literature study will mainly focus on two areas; firstly, previous work studying source code visualizations, software evolution and technical debt, and secondly literature concerning methods of evaluating visualizations and conducting user studies within the visualization research context.

Examples of work within source code visualization includes *CVSscan* (Voinea et al., n.d.), *code_swarm* (M. Ogawa & Kwan-Liu Ma, 2009) and *Supporting Software Decision Meetings: Heatmaps for Visualising Test and Code Measurements* (Feldt et al., 2013). Regarding methods and user studies, this literature study will include *Top scientific visualization research problems* (Johnson, 2004), *Software Design Patterns for Information Visualization* (Heer & Agrawala, 2006) and *Handbook of human centric visualization* (Huang & Eades, 2014).

Time plan

The project will be conducted over 22 weeks between week 3 and week 24. The following table describes the planned activities during the project. Both the length and planning of the project might be updated at a later date.

Time Plan	Planning			Pre-study				Development and testing								Writing report					Examination	
	3	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**
Away				Half week						Half week												
Specification	🟨	🟨	🟨																			
Litterature study			🟩	🟩	🟩	🟩	🟩															
Write introduction and background					🟨	🟨	🟨															
Concept generation							🟦	🟦														
Explore design space (sketching)									🟦	🟦												
Develop design artifact (prototype)											🟦	🟦	🟦	🟦								
Design contextualisation & refinement															🟦	🟦						
Critique sessions (user testing)											🟪			🟪	🟪							
Write method											🟨	🟨	🟨									
Write result														🟨	🟨	🟨						
Write discussion																🟨	🟨	🟨				
Write conclusion																		🟨	🟨	🟨		
Presentation and feedback																					🟦	🟦

References

Feldt, R., Staron, M., Hult, E., & Liljegren, T. (2013). Supporting Software Decision Meetings: Heatmaps for Visualising Test and Code Measurements. *2013 39th Euromicro Conference on Software Engineering and Advanced Applications*, 62–69. https://doi.org/10.1109/SEAA.2013.61

Heer, J., & Agrawala, M. (2006). Software Design Patterns for Information Visualization. *IEEE Transactions on Visualization and Computer Graphics*, *12*(5), 853–860. https://doi.org/10.1109/TVCG.2006.178

Huang, W., & Eades, P. (Eds.). (2014). *Handbook of human centric visualization*. Springer.

Johnson, C. (2004). Top scientific visualization research problems. *IEEE Computer Graphics and Applications*, *24*(4), 13–17. https://doi.org/10.1109/MCG.2004.20

Ogawa, M., & Kwan-Liu Ma. (2009). code_swarm: A Design Study in Organic Software Visualization. *IEEE Transactions on Visualization and Computer Graphics*, *15*(6), 1097–1104. https://doi.org/10.1109/TVCG.2009.123

Ogawa, Michael, & Ma, K.-L. (2010). Software evolution storylines. *Proceedings of the 5th International Symposium on Software Visualization - SOFTVIS '10*, 35. https://doi.org/10.1145/1879211.1879219

Stolterman, E., & Wiberg, M. (2010). Concept-Driven Interaction Design Research. *Human–Computer Interaction*, *25*(2), 95–118. https://doi.org/10.1080/07370020903586696

Voinea, L., Telea, A., & van Wijk, J. J. (n.d.). *CVSscan: Visualization of Code Evolution*. 12.