CodeSketch: Drawing Diagrams in Source Code

Reza Adhitya Saputra
University of Waterloo
radhitya@uwaterloo.ca

Raminder Sodhi
University of Waterloo
rjsodhi@uwaterloo.ca

ABSTRACT

A picture is worth a thousand words. Similarly, software developers also sketch diagrams for various purposes. The diagrams they draw can be flowcharts, architecture diagrams, or algorithm descriptions. We develop a tool called CodeSketch based on several criteria of what make good diagramming tool in Open Source Software (OSS) development. Using CodeSketch, a developer can draw a diagram consists of rectangles, lines, arrows, and texts. The tool then converts the diagram to ASCII art which is directly embedded in source code. Compared to other diagram representation, diagrams created using CodeSketch can coexist with source code files, they do not require specific graphics viewer tool other than a text editor, and they are easy to be stored in a revision control system. Finally, we conduct a user study where participants can use our tool, then we collect their feedback.

Keywords

Source code comments, documentation, diagrams, ASCII art

1. INTRODUCTION

Diagrams are important visual representation which are widely used in design arts, engineering, and science. The use of diagrams and images is also pervasive in software development since diagrams are viewed as alternatives to verbal communication. Furthermore, diagrams enables developers to visualize concepts and communicate ideas. For example, a developer would sketch informal diagrams on scrap papers or whiteboards, then have a discussion with their coworkers. In more formal purposes, diagrams can be created using a specific designer tool. However, the designer tool restricts the developer to use a particular framework so the created diagrams cannot be exported easily to other platforms with different development tools.

In co-located software development, diagrams are used during discussions, meetings, or presentations. However, these diagrams have transitory nature [4]. One reason is that because they only serve a purpose to explain concepts to coworkers. After the coworkers understood, sketches on whiteboard will be erased and diagrams on scrap papers would be thrown into garbage bins. Although these diagrams can be converted digitally for long term purposes, such as documentation, redrawing these diagrams is tedious.

In Open Source Software (OSS) development, a practice of drawing diagrams using ASCII art has been reported. They avoid creating a lot of bitmap-based diagrams on documentation or wiki pages because they do not bother to keep these diagrams be updated when they changed their source code [11]. Nevertheless, if they must draw diagrams to communicate their ideas, they tend to use ASCII art. Although ASCII art is a simple form of diagrams, it has a practical benefit compared to other graphical representations. ASCII art enables developers not to depend on a specific designer tool. Indeed, in an open source project every developer may not use the same tools. Based on the observation, a tool-independent diagram representation is required. Therefore, our project focuses on OSS settings and the use of ASCII art.

In this project, we developed a diagramming tool to create ASCII art based on criteria we have collected during literature study. The created diagrams can be edited easily and viewed without a specific tools. We did a user study conducted on <X> participants. The user study consists of several tasks on which participant can try to use our tool, then followed with questionnaire. As a final step, we gather several interesting feedback.

2. RELATED WORK

Our proposed idea is related to why developers draw diagrams and creating diagrams as source code comments.

Why developers draw diagrams. In co-located teams, diagrams help developers to understand source code, starting discussions, improve documentations and they can be used as presentation aids during a meeting [4]. In OSS community, developers agree that diagramming is useful but they do not like to redraw diagrams after they updated their source code [11, 5]. If they must draw diagrams, they tend to use ASCII art since they have low cost in creation. Moreover, they prefer to put diagrams inside a revision control system, so everyone can keep track changes. In another study, the use of ASCII art in bug reports also has been investigated [10].

Creating diagrams as source code comments. GUIIO is a tool to create a GUI mockup embedded in source code [9]. Similar to our tool, they also use ASCII art. The

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
CS846 '15 Waterloo, Canada
Copyright 2015 ACM X-XXXXX-XX-X/XX/XX ...$15.00.

mockup is used as an explanation to source code snippets which actually implement the actual GUI. They argue that the approach can reduce window switching between source code and designer tool. However, GUIIO cannot be used to draw more general diagrams, e.g, flowcharts.

3. SYSTEM DESIGN

We summarize nine criteria of an ideal diagramming tool based on literature study of previous research on diagramming practice in OSS development [11, 5]. Subsequently, table 1 shows the supportability of CodeSketch to these criteria. Note that CodeSketch does not satisfies all criteria.

1. Diagrams coexist with existing communication tools.
2. The diagramming tool is widely available and free.
3. Diagrams can be published based on who the intended audience is.
4. The diagramming tool can be Integrated into development infrastructure.
5. Diagrams are easily to be stored in revision control system.
6. The "source code" of diagrams can be shared.
7. Diagrams are not tedious to create and edit.
8. Diagrams do not require specific tools to create and display.
9. Diagrams support diff tool.

Criteria - Supportability	**Explanation**
1 - Yes	Diagrams created using CodeSketch can be easily embedded into various communication tools such as mailing list, IRC, bug report tools, code review tools, etc.
2 - Yes	CodeSketch is a free web-based app.
3 - No	CodeSketch is represented using unicode characters. It can be used for explaining source code to coworkers, but unsuitable for a presentation in a formal meeting, e.g. with stakeholders or end users.
4 - Yes	It is a web based tool and will be embedded as source code comment, so is platform independent.
5 - Yes	Unlike binary images, our diagram representation is text based so it can be stored in any revision control system
6 - Yes	The diagrams themselves are also the source code. However, other diagram representations do have "source code" which can be shared, for example XML-like language for GUI.
7 - No	An ambitious goal is to automate the creation of diagrams.However, our tool does not make drawing diagrams easier.
8 - Partially	It requires a specific tool to create proper ASCII arts, since drawing an ASCII art manually would be difficult. Still, displaying an ASCII art does not require any specific tools other than a text editor.
9 - Partially	Diff tool can be used to compare two ASCII diagrams. Since diff tool is line based, any editing on a particular line will highing the entire line

Table 1: CodeSketch's criteria supportability

4. IMPLEMENTATION

■<Github, js, chrome extension, html5 canvas■> ■< Features: draw rectangle, draw lines, draw arrow lines, text, delete ■>

■<add a screenshot of the tool, the canvas should visible and the converted diagram should also be visible■> ■<a paragraph of explanation of the screenshot■>

5. USER STUDY

■<How do we conduct the user study■> ■<Evaluation■>

6. RESULTS

■<How do we conduct the user study■> ■<Evaluation■>

7. CONCLUSIONS

We have several future work in mind. First, we would like to know the effectiveness of using diagrams in code reviews or bug reports. In code reviews, developers have a problem in understanding source code they do not own so they cannot effectively find defects on the source code [1]. A similar problem is also found in bug reports where reporters do not give much explanation. Therefore, an important question is embedding diagrams would improve code understanding in code review? and would embedding diagrams in bug reports would make the bug description is more informative? Second, we feel it is necessary to develop a way for easier diagramming. Ideally, editing and updating ASCII diagrams should not be tedious, or even fully automatic. Third, we would like to know whether diagramming would be useful in collaborative settings, for example, micro-outsourcing [7]. Lastly, developing a customized diff tool to visualize how a diagram evolves over time would be an interesting research direction, similar to Nonlinear Revision Control for Images [3].

8. REFERENCES

[1] A. Bacchelli and C. Bird. Expectations, outcomes, and challenges of modern code review. In *Proceedings of the 2013 International Conference on Software Engineering*, ICSE '13, pages 712–721, Piscataway, NJ, USA, 2013. IEEE Press.

[2] A. Bragdon, R. Zeleznik, S. P. Reiss, S. Karumuri, W. Cheung, J. Kaplan, C. Coleman, F. Adeputra, and J. J. LaViola, Jr. Code bubbles: A working set-based interface for code understanding and maintenance. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '10, pages 2503–2512, New York, NY, USA, 2010. ACM.

[3] H.-T. Chen, L.-Y. Wei, and C.-F. Chang. Nonlinear revision control for images. In *ACM SIGGRAPH 2011 Papers*, SIGGRAPH '11, pages 105:1–105:10, New York, NY, USA, 2011. ACM.

[4] M. Cherubini, G. Venolia, R. DeLine, and A. J. Ko. Let's go to the whiteboard: How and why software developers use drawings. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '07, pages 557–566, New York, NY, USA, 2007. ACM.

[5] E. Chung, C. Jensen, K. Yatani, V. Kuechler, and K. Truong. Sketching and drawing in the design of open source software. In *Visual Languages and Human-Centric Computing (VL/HCC), 2010 IEEE Symposium on*, Sept 2010.

[6] Github. Github, 2015.

[7] M. Goldman, G. Little, and R. C. Miller. Real-time collaborative coding in a web ide. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology*, UIST '11, pages 155–164, New York, NY, USA, 2011. ACM.

[8] D. E. Knuth. Literate programming. *The Computer Journal*, 27:97–111, 1984.

[9] J. Simpson and M. Terry. Embedding interface sketches in code. In *Proceedings of the 24th Annual ACM Symposium Adjunct on User Interface Software and Technology*, UIST '11 Adjunct, New York, NY, USA, 2011. ACM.

[10] M. Twidale and D. Nichols. Exploring usability discussions in open source development. In *System Sciences, 2005. HICSS '05. Proceedings of the 38th Annual Hawaii International Conference on*, pages 198c–198c, Jan 2005.

[11] K. Yatani, E. Chung, C. Jensen, and K. N. Truong. Understanding how and why open source contributors use diagrams in the development of ubuntu. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '09, New York, NY, USA, 2009. ACM.