Information visualization in software testing and maintenance
A Literature Survey

Marjane Namavar

marjane@ece.ubc.ca

Introduction

Software developers continuously apply changes to add new features or improve code quality. Such changes introduce bugs, which are estimated to cost the global economy $312 billion per year and which software developers are thought to spend at least 50% of their programming time finding and fixing[1]. Test techniques include the process of executing a program or application with the intent of finding software bugs (errors or other defects), and verifying that the software product is fit for use. Software maintenance address bug fixes and minor enhancements. Visualization of the faults and enhancements can improve testing and maintenance tasks.

Goals and Objectives:

(1) Survey the existing literature focusing on the use of visualization for software testing and maintenance. (2) Analyze the data from empirical experiments under what/why/how framework. (3) Abstract gathered information to categorize existing approaches.

Background:

Extending definition of program visualization [2] to other software testing and maintenance artifacts; software visualization can be defined as the mapping from software artifacts—including programs—to graphical

representations. Software testing and maintenance visualization is needed because software itself, software bugs and fixed are invisible[3]. In the simplest case, we may visualize artifacts textually, which is considered the most primitive kind of visualization (roughly speaking). There are empirical studies that show evidence that specific ways of graphical visualization work better than textual visualization for certain tasks [4]. In other cases, a textual presentation is likely to be the most appropriate [5,6]. We know from empirical studies that maintenance programmers spend 50% of their time simply trying to understand the software to be changed [7] and it is plausible that the method of visualization has a substantial effect on the time needed to comprehend large programs—be it positive or negative.

Visualization in general is created to augment human capabilities in performing a task [11]. Visualizations in software testing and maintenance have been proposed in prior research. For example one of the tasks required to reduce the number of delivered faults, is debugging which is one of the most time-consuming [8, 9], and locating the errors is the most difficult component of this debugging task [10]. Clearly, techniques that can reduce the time required to locate faults can have a significant impact on the cost and quality of software development and maintenance. Another work [7] presented a visualization technique that provides a global view of the results of executing a program with an entire test suite.

Proposed Plan:

I plan to conduct this study in two major phases. For the first phase – to be completed by November 12th – I will gather the relevant methodology papers that tried to apply visualization for software testing and maintenance. The second phase will involve performing the review of the papers found in the first phase. In my initial search, I found some papers that had done an

empirical study comparing different visualization methods in software maintenance [3]. The contribution of this work would be analysis and synthesis of the findings of past researchers, and broadening the base of their studies. The second phase will be completed by December 10th.

Milestone	deadline	Hours
- Gather (23-25) relevant papers	12th Nov.	5
- Review all papers one time to achieve a big picture - Review some relevant survey papers to gain an idea about doing survey project in this area - Prepare slides and describe the big picture and findings so far	19th Nov.	10
- Select papers and the extent to which they're going to be analyzed - Analysis of all selected papers under what/why/how framework - Prepare slides and do a high-level presentation of the analysis work	4th Dec.	30
- Synthesize information and categorize approaches in selected papers - Prepare final presentation - Start Writing final paper	10th Dec.	15
- Complete and edit final paper	13th Dec.	10

Personal Background:

I started my M.A.Sc. program at the Electrical and Computer Engineering Department, UBC in September 2019, under the supervision of Professor Ali

Mesbah who has published high impact research in areas of software testing and maintenance which will be my area of research. The proposed project for this course provides high synergy and complementarity value with my future research.

References:

[1] S. Collofello and S. N. Woodfield. Evaluating the effectiveness of reliability-assurance techniques. Journal of Systems and Software, 9(3):191-195, 1989.

[2] Roman G-C, Cox KC. Program visualization: The art of mapping programs to pictures. Proceedings of the International Conference on Software Engineering. ACM Press: New York, 1992; 412–420.

[3] Koschke, Rainer. "Software visualization in software maintenance, reverse engineering, and re-engineering: a research survey." Journal of Software Maintenance 15 (2003): 87-109.

[4] Hendrix TD, Cross JH,Maghsoodloo S, McKinney ML. Do visualizations improve program comprehensibility experiments with control structure diagrams for Java? Proceedings of the 31st SIGCSE Technical Symposium on Computer Science Education. ACM Press: New York, 2000; 382–386.

[5] Curtis B, Sheppard SB, Kruesi-Bailey E, Bailey J, Boehm-Davis DA. Experimental evaluation of software documentation formats. Journal of Systems and Software 1989; 9(2):167–207.

[6] Green TRG, Petre M. When visual programs are harder to read than textual programs. Proceedings of the 6th European Conference on Cognitive Ergonomics. Springer: Berlin, 1992; 167–180.

[7] Eagan, M. J. Harrold, J. Jones, and J. Stasko. Technical note: Visually encoding program test information to find faults in software. In Proceedings of IEEE Information Visualization, pages 33-36, October 2001.

[8] Ball and S. G. Eick. Software visualization in the large. Computer, 29(4):33-43, Apr. 1996.

[9] Telcordia Technologies, Inc. xA TAC: A tool for improving testing effectiveness, http://xsuds.argreenhouse.com/htmlman/coverpage.html.

[10] I. Vessey. Expertise in debugging computer programs. International Journal of Man-Machine Studies: A process analysis, 23(5):459-494, 1985.

[11] Munzner, T., Visualization Analysis and Design, A K Peters Visualization Series, CRC Press, 2014.