Multi-touch Table User Interfaces for Co-located Collaborative Software Visualization

Craig Anslow
School of Engineering and Computer Science
Victoria University of Wellington, New Zealand
craig@ecs.vuw.ac.nz

ABSTRACT
Most software visualization systems and tools are designed from a single-user perspective and are bound to the desktop, IDEs, and the web. Few tools are designed with sufficient support for the social aspects of understanding software such as collaboration, communication, and awareness. This research aims at supporting co-located collaborative software analysis using software visualization techniques and multi-touch tables. The research will be conducted via qualitative user experiments which will inform the design of collaborative software visualization applications and further our understanding of how software developers work together with multi-touch user interfaces.

ACM Classification: H1.2 [User/Machine Systems]: Human Factors; H5.2 [Information interfaces and presentation]: User Interfaces. - Multi-touch user interfaces.

General terms: Experimentation, Human Factors.

Keywords: Multi-touch user interfaces, software visualization, user evaluation.

1. INTRODUCTION

Everything we do in society ranging from web browsing to purchasing items depends on reliable software. Maintaining existing large and complex systems requires understanding the underlying software. Understanding the complex structure and behaviour of large software systems is a hard task. Software visualization aims to help understanding with techniques to visualize the structure, behaviour, and evolution of software [4]. Understanding software is often a social activity and involves teams of software developers. The field of CSCW explores how users behave with digital tools during collaborative work. Few studies have explored how tools support collaborative software understanding [12] and collaborative software visualization [19].

Most software visualization systems and tools are designed from a single-user perspective and are bound to the desktop, Integrated Development Environments (IDE) like Eclipse, and the web. These design decisions do not allow users in a co-located environment (within the same room) to easily collaborate, communicate, or be aware of each others activities to analyse software [19]. Multi-touch user interfaces are an interactive technique that allows single or multiple users to collaboratively control graphical displays with more than one finger, mouse pointer, or tangible object on various kinds of surfaces and devices.

The goal of this research is to investigate whether multi-touch interaction techniques are more effective for co-located collaborative software visualization than existing single user desktop interaction techniques. The research will be conducted by user experiments to observe how users interact and collaborate with a multi-touch software visualization application. Users will complete a sequence of scenarios that involve software understanding tasks using the software visualization application and a low-cost multi-touch table.

2. METHODOLOGY

A qualitative research approach will be used for the methodology as it is well suited for giving an overview of a situation and to examine how and why certain users behave in certain environments [3]. The qualitative approach includes observational studies conducted as part of the design process and in situ interviews. These approaches have been successfully adopted within the CSCW community to understand the different behaviour of users with tools during collaborative work. These approaches have bee adopted to provide insight into the design of tabletop displays for information visualization [7], visual analytics [8], and collaborative design [17]. A similar approach will be adopted for this research.

A significant barrier to exploring multi-touch table applications is the cost of the necessary hardware and software. Commercial multi-touch tables such as Microsoft Surface, Mitsubishi DiamondTouch, or Smart Touch Tables cost many thousands of dollars and are well beyond the reach of consumers and most research labs. Many research groups are addressing the problem of the cost of commercial multi-touch tables by developing their own low-cost multi-touch tables [5, 6, 11]. For this research a rear diffused illumination multi-touch table will be built [16, 20].

Multi-touch programming toolkits will first be surveyed and then a software visualization application for the multi-touch table will be built using one of these toolkits. Successful software visualization techniques for understanding the structure of large software systems will be identified and then adopted and modified where necessary for the application. Where appropriate new multi-touch software visualization gestures will be designed.

Once a multi-touch software visualization application has been built user experiments will be conducted. The purpose of the user experiments is to investigate whether the design of the multi-touch interaction techniques are more effective for co-located collaborative software visualization than existing single user interaction techniques. The design of the multi-touch software visualization application and user experiments will follow an iterative cycle using a grounded evaluation process to validate the design decisions [9].

3. PRELIMINARY RESULTS

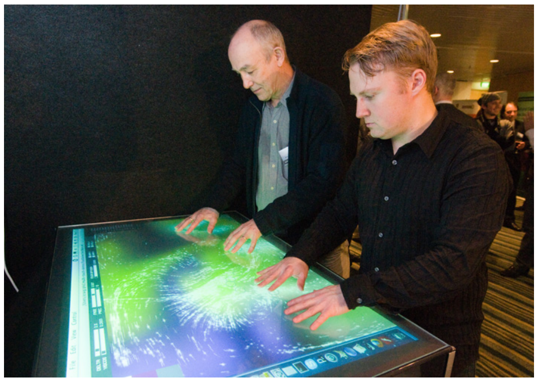
Different information visualization tools and toolkits have been explored to create visualizations of software metrics [1]. Polymetric Views [13] have been identified as a successful visualization technique for understanding the structure of large software systems and have been widely adopted by many tools including Code Crawler [14], CodeCity [22], and Lagrein [10].

As part of identifying Polymetric Views a user study was conducted to evaluate the effectiveness of one of the techniques, a modified version of the System Hotspots View, using a large visualization wall [2]. Figure 1(a) illustrates the large visualization wall which has 12 screens for a total display of 10240 x 4800 pixels. The study asked 14 participants to identify key measurements and comparisons of the package and classes from the Java Standard API version 1.6 using the System Hotspots View displayed on the visualization wall. The results indicated that users were able to effectively use the modified System Hotspots View to explore the example domain. However, there were issues around interacting with the visualization wall as the images were static and hard for the user to manipulate. The results of this study led to exploring the more interactive multi-touch interface paradigm.

A prototype rear DI multi-touch table was built by adapting an existing trolley table to create a 28 inch multi-touch surface. The tabletop used 5mm of clear acrylic glass (top, detection layer) and 3mm of Plexiglas RP 99561 (bottom, projection layer). The projector was a Sony VPS 1300 XGA at 1024x768 (4:3) resolution. The image was bounced off a mirror onto the bottom of the projection layer. A Sony PS3 Eye camera was modified to see only infrared. Two infrared LED security spotlights (850 nanometers) were purchased from a local electronics store. Community Core Vision (CCV)[1] 1.2 was used for blob detection. A MacBook Pro operating MacOSX 10.5.8 with 4GB RAM and 2.6GHz Intel Core 2 Duo was used for hosting the detection software and the client applications.

A number of issues were discovered from building the prototype table. The size of the touch surface was too small for two or more users, and the projection layer caused IR hotspots, created a poor viewing angle for users, and could

(a) Polymetric Views on large visualization wall.


(b) Rear DI multi-touch table.

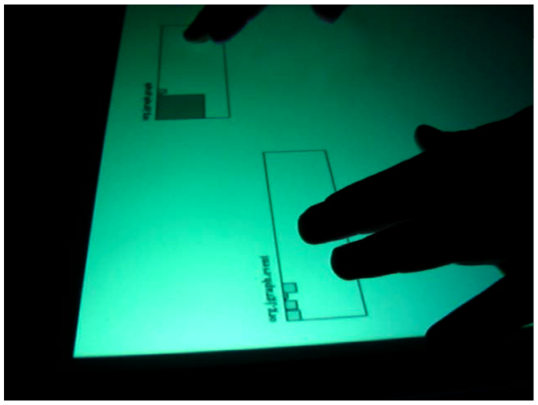

(c) Multi-touch Polymetric Views.

Figure 1: Preliminary Results.

only operate in dark lighting conditions. The mirror caused ghosting effects with the displayed image, the frame rate of the camera was too slow for detecting fast movement, and the table was not very portable.

Figure 1(b) illustrates the second and improved larger rear DI multi-touch table with an example from the MT4J toolkit[2]. The table is made out of a steel frame, with wooden slidable panels, and sturdy wheels for portability. The touch surface of the table is 1080mm width, 680mm depth, 1280mm di-

[1]http://ccv.nuigroup.com

[2]http://www.mt4j.org

agonal which is approximately 50 inches. The height of the table is 930mm. The tabletop uses 3mm of clear acrylic glass (top, detection layer) and 5mm of Plexiglas RP 7D006 (bottom, projection layer) which removes the IR hotspot issue, allows a greater viewing angle, and can be used in natural light. A Sanyo PLC WXL46 WXGA at 1280x768 (16:10) resolution short throw projector is used, which removes the mirror ghosting issue. The projector is mounted to a draw which helps with portability as the draw slides out when in operation and closed when transporting. The same Sony PS3 camera is used but with a wide angle lens mounted to cover the larger touch surface. Eight infrared LED bars from Environmental Lights and four LED security spotlights are used. CCV 1.3 is used which allows higher frame rates and a Dell Optiplex 780 operating Windows Vista with 4GB RAM and 2.7GHz Intel Core 2 Duo for both client and server.

Figure 1(c) shows a multi-touch System Hotspots View for the JGraph application (Java graph application) that was implemented using the TUIOZones library[3], a Processing library[4]. The figure shows just two packages from JGraph with about 10 classes in each package. Users can drag and resize packages and select classes to see their inner details.

4. CONTRIBUTIONS

The first contribution will be a multi-touch software visualization tool called SourceVis that supports collaborative software visualization on the multi-touch table. SourceVis will display metrics-based visualizations of the Java software from the Qualitas Corpus [21] which is a collection of Java programs to be used for empirical software engineering. The initial visualization technique for SourceVis will use modified versions of all the different Polymetric Views [13].

Each of the different Polymetric Views will be implemented following an iterative cycle beginning with the view that gives an overview of a system and progressing to the views that give more finer details about a system. The first view will be the System Hotspots View as that focuses on packages. The System Complexity view will come next as that shows the different relationships between packages and classes. The Class Blueprint will then follow as that is concerned with looking at the inner details of classes. Finally, views concerned with inheritance, methods, and attributes will be implemented.

SourceVis will support many of the user defined set of gestures [23]. Some of these gestures include tap for select, dragging an object with one finger, rotating and resizing an object with two fingers, manipulating the background by zooming in and out and scrolling up and down, and grouping objects by drawing a shape around them with one finger. During the iterative development cycle domain specific gestures for software visualization will become apparent and where necessary these gestures will be implemented within the visualizations. Gestures will be required to switch between different visualizations and allow users to have independent views of the software being visualized.

In addition to the Polymetric Views the visualizations will be augmented with the source code from the applications and associated Javadoc in different views to provide comprehensive documentation. For example a user will be able to select a package or class in the visualization and the source code or associated Javadoc page will be displayed in another view on the multi-touch user interface.

The main (second) contribution of the research will be user experiments conducted with SourceVis to collect qualitative data. The user experiment design will involve within-subjects testing [15]. Each experiment will have up to three users working as a group and they will get to use all the tools that are being tested. The tools include SourceVis and some of the single user tools that implement the Polymetric Views as cited earlier. The other software visualization tools will be setups on a desktop computer. Each group will start with either SourceVis or one of the tools that implement Polymetric Views and then switch around to use the other tools. The purpose for doing a within-subjects test is to see how the different technology influences the groups behaviour, whether their behaviour is different, and how the technology leads to different kinds of discoveries they can make.

A representative sample of applications from the Qualitas Corpus will be selected. Different Java applications will be used with each different tool. This is to avoid the bias of users becoming experts from learning the structure of one Java application with a tool and then using the knowledge gained about that application with the next tool.

The tasks for each experiment will involve software maintenance scenarios where participants will answer questions about the structure of the Java applications [12, 18]. Some example questions could include: “what are the largest packages and classes in the application?”, “what class has grown the most between the different versions of the application?”, and “what classes are coupled the most with other classes?”

Participants in the experiments will be professional and student software developers who will be recruited from local mailing lists and from within the department. With each participant’s consent their actions will be recorded with screen capture software and video recording. Participants will be asked to use the think aloud protocol which focuses on users talking about their actions, perceptions, and expectations regarding the interface and functionality [15]. Getting the users to talk about their actions and thoughts will help to gain insight into how each user views the computer system, identification of their misconceptions, and what parts of the interface cause the most problems.

5. CONCLUSIONS

The goal of this research is to investigate whether multi-touch interaction techniques are more effective for co-located collaborative software visualization than existing single user interaction techniques to understand the complex structure of large software systems. The preliminary results of the research have identified software visualization techniques for understanding the structure of large systems and conducted a user study of a Polymetric View. Two multi-touch tables have been built, multi-touch toolkits have been explored, and some multi-touch software visualization prototypes have

[3]http://jlyst.com/tz/

[4]http://processing.org

been implemented. The contributions of this research includes implementing a multi-touch software visualization application and conducting user experiments to observe the interaction behaviour of software developers with the multi-touch table and application. The results will inform the design of collaborative software visualization applications and further our understanding of how software developers work together with multi-touch user interfaces.

ACKNOWLEDGMENTS

This work is supported by the New Zealand Foundation for Research Science and Technology for the Software Process and Product Improvement project, and a Telstra Clear scholarship. Thanks to Stuart Marshall, James Noble, Robert Biddle, and Ewan Tempero for feedback. Thanks to Roger Cliffe, Jordan Hochenbaum, and Owen Vallis for assistance.

REFERENCES

1. Craig Anslow, James Noble, Stuart Marshall, and Ewan Tempero. Towards visual software analytics. In *Proceedings of the Australasian Computing Doctoral Consortium (ACDC)*, 2009.

2. Craig Anslow, James Noble, Stuart Marshall, Ewan Tempero, and Robert Biddle. User evaluation of polymetric views using a large visualization wall. In *Proc. of Symposium on Software Visualization (SoftVis)*. ACM, 2010.

3. John W. Creswell. *Qualitative Inquiry and Research Design Choosing Among Five Traditions*. Sage Publications, 1998.

4. Stephan Diehl. *Software Visualization: Visualizing the Structure, Behaviour, and Evolution of Software*. Springer Verlag, 2007.

5. Jefferson Han. Low-cost multi-touch sensing through frustrated total internal reflection. In *Proc. of UIST*. ACM, 2005.

6. Jordan Hochenbaum and Owen Vallis. Bricktable: A musical tangible multi-touch interface. In *Proc. of Berlin Open Conference*, 2009.

7. Petra Isenberg. *Collaborative Information Visualization in Co-located Environments*. PhD thesis, University of Calgary, 2009.

8. Petra Isenberg, Danyel Fisher, Meredith Ringel Morris, Kori Inkpen, and Mary Czerwinski. An exploratory study of co-located collaborative visual analytics around a tabletop display. In *Proc. of the Symposium on Visual Analytics Science and Technology (VAST)*. IEEE, 2010.

9. Petra Isenberg, Torre Zuk, Christopher Collins, and Sheelagh Carpendale. Grounded evaluation of information visualizations. In *Proc. of the Workshop on BEyond time and errors: novel evaLuation methods for Information Visualization (BELIV)*. ACM, 2008.

10. Andrejs Jermakovics, Raimund Moser, Alberto Sillitti, and Giancarlo Succi. Visualizing software evolution with lagrein. In *OOPSLA Companion*. ACM, 2008.

11. Rilla Khaled, Pippin Barr, Hannah Johnston, and Robert Biddle. Let's clean up this mess: exploring multi-touch collaborative play. In *CHI Extended Abstracts*, 2009.

12. Andrew J. Ko, Robert DeLine, and Gina Venolia. Information needs in collocated software development teams. In *Proc. of ICSE*. IEEE, 2007.

13. Michele Lanza and Stéphane Ducasse. Polymetric views-a lightweight visual approach to reverse engineering. *IEEE Transactions on Software Engineering*, 29(9):782–795, 2003.

14. Michele Lanza and Radu Marinescu. *Object-Oriented Metrics in Practice*. Springer Verlag, 2006.

15. Jakon Nielsen. *Usability Engineering*. Morgan Kaufmann, 1994.

16. Johannes Schning, Peter Brandl, Florian Daiber, Florian Echtler, Otmar Hilliges, Jonathan Hook, Marku Lchtefeld, Nima Motamedi, Laurence Muller, Patrick Olivier, Tim Roth, and Ulrich von Zadow. Multi-touch surfaces: A technical guide. Technical Report TUM-I0833, University of Munster, 2008.

17. Stacey D. Scott, M. Sheelagh T., Carpendale, and Kori M. Inkpen. Territoriality in collaborative tabletop workspaces. In *Proc. of CSCW*. ACM, 2004.

18. Jonathan Sillito, Gail C. Murphy, and Kris De Volder. Questions programmers ask during software evolution tasks. In *Proc. International Symposium on Foundations of Software Engineering (FSE)*. ACM, 2006.

19. Margaret-Anne D. Storey, Chris Bennett, R. Ian Bull, and Daniel M. German. Remixing visualization to support collaboration in software maintenance. In *Proc. of International Conference on Software Maintenance (ICSM)*. IEEE, 2008.

20. Alex Teiche, Ashish Kumar Rai, Chris Yanc, Christian Moore, Donovan Solms, Grkem Cetin, Justin Riggio, Nolan Ramseyer, Paul D'Intino, Laurence Muller, Ramsin Khoshabeh, Rishi Bedi, Mohammad Taha Bintahir, Thomas Hansen, Tim Roth, and Seth Sandler. Multi-touch technologies. http://nuigroup.com/, 2009.

21. Ewan Tempero, Craig Anslow, Jens Dietrich, Ted Han, Jing Li, Markus Lumpe, Hayden Melton, and James Noble. Qualitas corpus: A curated collection of Java code for empirical studies. In *Proc. of Asia Pacific Software Engineering Conference (APSEC)*, 2010.

22. Richard Wettel and Michele Lanza. Visualizing software systems as cities. In *Proc. of the Workshop on Visualizing Software for Understanding and Analysis (VISSOFT)*. IEEE, 2007.

23. Jacob O. Wobbrock, Meredith Ringel Morris, and Andrew D. Wilson. User-defined gestures for surface computing. In *Proc. of CHI*. ACM, 2009.