SourceVis: A Tool For Multi-touch Software Visualization

Craig Anslow, Stuart Marshall, James Noble
School of Engineering and Computer Science
Victoria University of Wellington
Wellington, New Zealand
{craig,stuart,kjx}@ecs.vuw.ac.nz

Robert Biddle
School of Computer Science
Carleton University
Ottawa, Canada
robert_biddle@carleton.ca

ABSTRACT

Most software visualization systems and tools are designed from a single-user perspective and are bound to the desktop and Integrated Development Environments (IDEs). These design decisions do not allow users to easily navigate through software visualizations or to analyse software collaboratively. We have developed SourceVis, a collaborative multi-touch software visualization prototype for multi-touch tables. In this paper we describe the visualizations and interaction capabilities of our prototype.

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, software visualization, user study.

1. INTRODUCTION

Maintaining large software systems requires understanding the underlying systems. Understanding the complex structure and APIs of these systems is a hard task. Understanding software is often a social activity and involves teams of software developers. Software visualization aims to help with techniques to visualize the structure, behaviour, and evolution of software [2]. Visualization tools designed for a single user perspective make it hard for developers to analyse software when working together in a co-located environment (within the same room) using the same interface.

Multi-touch table user interfaces are an example of a co-located collaborative tool which could be used for software analysis. We are investigating whether multi-touch table interaction techniques are more effective for co-located collaborative software visualization than existing single user desktop interaction techniques. In this paper we describe our emerging multi-touch software visualization prototype.

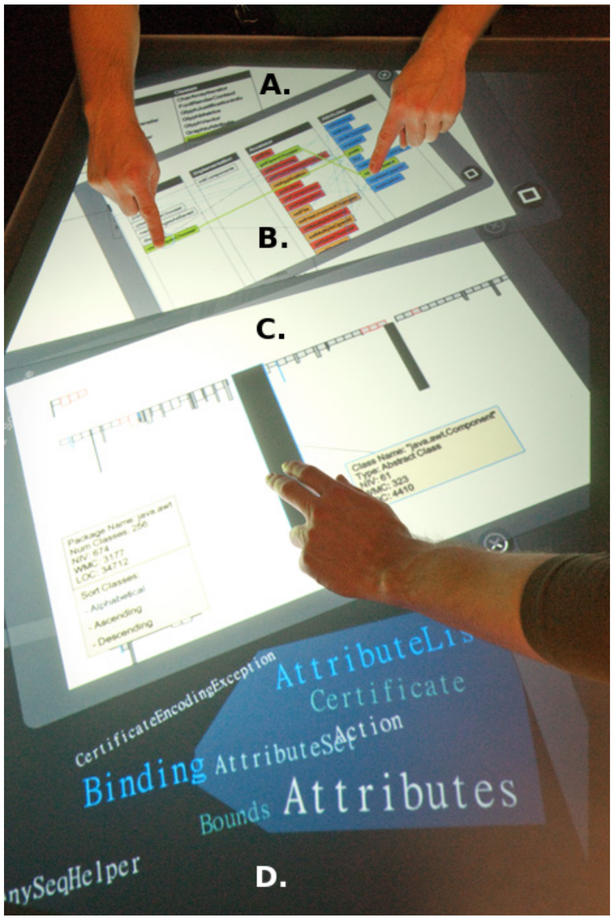

Figure 1: SourceVis. Users interacting with visualizations: A. Metrics Explorer, B. Class Blueprint, C. System Hotspots View, D. Wordle Vocabulary.

2. SOURCEVIS

SourceVis currently supports four visualization techniques for analysing the structure and vocabulary of software systems, see Figure 1. The visualizations adapt existing information and software visualization techniques and modify them to support multi-touch and multi-user interaction. SourceVis was developed using the MT4j toolkit [5].

Interaction. We envision developers using our prototype to explore how a software system has been structured by looking at visualizations that depict measurements about the underlying system and to look at what kind of vocabulary has been employed. This is a first maintenance step to analyse what parts of the system are large, problematic, and need to be refactored. Users first load a system before any visualizations can be displayed. An overview visualization can then be displayed such as the Metrics Explorer or System Hotspots View and then drill down into details to display one of the linked visualizations such as the Class Blueprint. Multiple visualizations can be displayed at once.

For a multi-user scenario one developer might be looking at the overview of a system while another is looking at the details of a class within the system. Each visualization is displayed in a rotatable and scalable window with options to display at full screen or close. This allows users to orient visualizations to where they are standing and change the size of each visualization.

Users can interact with individual elements in the visualizations. Tapping for select, dragging an element with one finger, and rotating and resizing elements with two fingers. Double tapping an element displays properties about that element. Multiple elements can be grouped by drawing a shape around them using a lasso gesture and then move them around the display. A tap and hold gesture displays a new visualization. Zooming uses a pinch gesture with one or two hands and navigation by scrolling with two fingers.

Metrics Explorer (Annotated A. in Figure 1). The Metrics Explorer shows metrics about the different modules in a system such as packages and classes. The metrics for a package are the total number of classes, number of variables (NIV), number of methods (WMC), and number of lines of code (LOC); and NIV, WMC, LOC for a class [3]. All the packages in a system are first displayed alphabetically. Tapping a package displays the metrics about the package in the metrics pane and the name of the package is highlighted cyan in both panes. Subsequently a list of classes from a package is also displayed in the classes pane. Tapping a class displays the metrics about a class in the metrics pane and highlighted green in both panes. In the figure a user has done a tap and hold gesture on one of the classes which has displayed the associated Class Blueprint and partially obscured the Metrics Explorer, but both can be repositioned.

Class Blueprint (B. in Figure 1). The Class Blueprint, adapted from Lanza et al., shows the dependencies and references between methods and attributes within a class [4]. The visualization is broken into five layers from left to right. The first four layers relate to the methods and the final layer to the attributes. The initialization layer displays initialization methods (e.g. constructors), interface layer public methods, implementation layer private methods, and accessor layer accessor and mutator methods (e.g. get()). The attribute layer displays all attributes of a class. The methods and attributes have a different fill colour depending on which layer they belong to. Likewise the edges for the dependencies and references. The weight of an edge can be adjusted by moving a slider up (for thickest) or down (for invisible). In the figure a user has selected one of the interface methods by tapping and holding which has highlighted in green one accessor method that it calls and one attribute it accesses. With their other hand they have selected one of the attributes which has highlighted a method that references that attribute which is the same accessor method the interface method calls.

System Hotspots View (C. in Figure 1). The System Hotspots View, adapted from Lanza et al., aims to highlight large packages and classes in a system [4]. Packages are displayed down the Y axis and classes from each package along the X axis. In the figure a user has double tapped on the package label which has displayed properties about the package in a linked yellow box. The properties include the total metrics (e.g. classes, NIV, WMC, LOC), and options for visually sorting the classes in the package alphabetically, ascending, or descending by individual metrics or groups of the metrics. Each class is represented as a rectangle where the width indicates the NIV and height WMC. The colour of a class is represented as the number of LOC. The darker the rectangle the more lines of code the class contains. Different border colours represent the type of class (e.g. red is interface, blue abstract class, no border a concrete class). A double tap on a class displays the properties including metrics. Classes can be moved around the visualization to be compared with other classes and can be grouped together. A tap and hold gesture on a class displays the associated Class Blueprint.

Vocabulary (D. in Figure 1). The modified Word Cloud and Wordle visualizations provide a quick overview of the vocabulary used in the entities (e.g. packages, classes, methods) of a software system to understand the coding standards employed. In the figure a user has grouped some of the larger words together and filtered out some of the smaller words. We intend to be able to link this visualization with others so that selecting or grouping words highlights entities in other displayed visualizations that use these words.

A user study of SourceVis is described elsewhere [1]. The results showed that participants were able to successfully answer all the questions in the user tasks and preferred to work collaboratively. We plan to extend SourceVis to support evolution visualizations of software. Once SourceVis is mature we will conduct a collaborative quantitative user experiment similar to Wettel et. al. involving industry professionals [6].

Acknowledgements. This work is supported by the New Zealand Ministry of Science and Innovation for the Software Process and Product Improvement project, TelstraClear, and the Canadian NSERC SurfNet Network.

REFERENCES

1. Craig Anslow, Stuart Marshall, James Noble, and Robert Biddle. A qualitative user study of SourceVis. Technical Report 11-09, Victoria University of Wellington, 2011.
2. Stephan Diehl. *Software Visualization: Visualizing the Structure, Behaviour, and Evolution of Software*. Springer Verlag, 2007.
3. Norman E. Fenton and Shari Lawrence Pfleeger. *Software Metrics: A Rigorous and Practical Approach*. PWS Publishing, 1998.
4. Michele Lanza and Stéphane Ducasse. Polymetric views-a lightweight visual approach to reverse engineering. *IEEE Trans. on Soft. Eng.*, 29(9):782–795, 2003.
5. Uwe Laufs, Christopher Ruff, and Jan Zibuschka. MT4j a cross-platform multi-touch development framework. In *Proc. of the Workshop on Engineering Patterns for Multi-Touch Interfaces at Symposium Engineering Interactive Computing Systems (EICS)*. ACM, 2010.
6. Richard Wettel, Michele Lanza, and Romain Robbes. Software systems as cities: A controlled experiment. In *Proc. of ICSE*. ACM, 2011.