Collaborative Software Visualization in Co-located Environments

by

Craig Anslow

A thesis
submitted to the Victoria University of Wellington
in fulfilment of the requirements for the degree of
Doctor of Philosophy
in Software Engineering.

Victoria University of Wellington
2013

Abstract

Most software visualization systems and tools are designed from a single-user perspective and are bound to the desktop and IDEs. These design decisions do not allow users to analyse software collaboratively or to easily interact and navigate visualizations within a co-located environment at the same time. This thesis presents an exploratory study of collaborative software visualization using multi-touch tables in a co-located environment. The thesis contributes a richer understanding of how pairs of developers make use of shared visualizations on large multi-touch tables to gain insight into the design of software systems.

We designed a collaborative software visualization application, called SourceVis, that contained a suite of 13 visualization techniques adapted for multi-touch interaction. We built two large multi-touch tables (28 and 48 inches) following existing hardware designs, to explore and evaluate SourceVis. We then conducted both qualitative and quantitative user studies, culminating in a study of 44 professional software developers working in pairs.

We found that pairs preferred joint group work, used a variety of coupling styles, and made many transitions between coupling and arrangement styles. For collaborative group work we recommend designing for joint group work over parallel individual work, supporting a flexible variety of coupling styles, and supporting fluid transitions between coupling and arrangement styles.

We found that the preferred style for joint group work was closely coupled and arranged side by side. We found some global functionally was not easily accessible. We found some of the user interactions and visual interface elements were not designed consistently. For the design of collaborative software visualizations we recommend designing visualizations for closely coupled arrangements with rotation features, providing functionality in the appropriate locality, and providing consistent user interactions and visual interface design.

We found sometimes visualization windows overlapped each other and text was hard to read in windows. We found when pairs were performing joint group work the size of the table was appropriate but not for parallel individual. We found that because the table could not differentiate between different simultaneous users that some pair interactions were limited. For the design of multi-touch tables we recommend providing a high resolution workspace, providing appropriate table space, and differentiating between simultaneous user interactions.

Acknowledgments

Thanks to my supervisors Professor James Noble and Dr. Stuart Marshall, without either of them I could not see how I could have finished this mammoth task. Especially when times were very tough. Your persistence and encouragement were immensely appreciated. Thanks to Professor Robert Biddle for providing sound advice from further afield, it was much appreciated. Thanks to Associate Professor Ewan Tempero for providing advice throughout the thesis. Special thanks to Roman Klapaukh for proof reading the entire thesis and offering useful technical suggestions.

Thanks to all the professional software developers and computer science students for participating in my user studies. Without them I simply could not have made the research findings and completed this thesis.

Special thanks to Petra Isenberg whose work this thesis builds upon. I am grateful to Petra for the time she spent discussing with me about her thesis and words of advice she provided.

Thanks to my examiners for examining the thesis and providing valuable feedback: T.C. Nicholas Graham, Stuart Charters, and Taehyun Rhee.

Thanks to the wonderful people in the Elvis - Software Design Research Group, fellow students, and colleagues who have made graduate school an enjoyable experience: Alex Potanin, David Pearce, Ian Welch, Radu Muschevici, Rashina Hoda, Siva Dorairaj, Pippin Barr, Rilla Khaled, Angela Martin, Keith Cassell, Adam Clarke, Nicholas Cameron, Stephen Nelson, Vipul Delwadia, Roman Klapaukh, Paley Li, Jan Larres, Frank Schmager, Hugh Davenport, Jennifer Ferreira, Matthew Duignan, Michael Waterman, Stephen Nelson, Kyle Chard, Siva Dorairaj, Diane Strode, Michael Homer, Tim Jones, Kourosh Nesthatian, Achim Gadke, Ben Palmer, Urvesh Bhowan, Rohitash Chandra, Van Lam Le, Hui Ma, Pavle Mogin, Thomas Kuehne, Marco Servetto, Peter Andreae.

Thanks to the following other students whom I helped with their project but I also gained valuable insight which helped me with my project: Jeremy Shipman, Esther Ng, Joshua Lindsay, Nick Vause, Hien Tran, Haowei Ruan, Yi-Jing Chung, Daniel Cope, Matthew Crisp, Marco Costantini, and Fahmi Abdulhamid.

Thanks to the programming, technical, and administration teams within the School of Engineering and Computer Science who have always been very generous and helpful: Roger Cliffe, Mark Davies, Royce Brown, Duncan McEwan, Ray Brownrigg, Kevin Buckley, Monique Damitio, Christo Muller, Tim Exley, Jason Edwards, Sean Anderson, Sue Hall, Amanda Holdaway, Prema Ram, Kelsey Firmin, and Ginny Whatarau. Thanks to staff from ITS: Laureen Jones, Sue Creese, Raymond Hutchinson, Fletcher Hanscomb, Michael Hikuroa, and Johny Flutey.

Thanks to the help of various people throughout the local computer science community in New Zealand who have helped in different aspects: Jens Dietrich, John Hosking, John Grundy, Jevon Wright, Graham Jenson, Rachel Blagojevic, Beryl Plimmer, Paul Schmeider, Carl Schultz, Alyona Medelyen, Andreas Loew, Paul Hunkin, Jason Alexander, Neville Churcher, Warwick Irwin, Tim Bell, Ian Witten, Craig Schock, and Robert Amor.

Along the journey I have met some interesting people who have helped me with my research at different stages: Shane Markstrum, Emerson Murphy-Hill, Keith Andrews, Peter Eades, Don Brutzman, James Hill, Donna Malayeri, Ciera Jaspan, Yoav Zibin, Wes Kendall, Fernanda Viegas, John Stasko, Chris Weaver, Jason Dykes, Jeff Heer, Michele Lanza, Stephan Diehl, Carsten Georg, Fabian Beck, Alex Telea, Tobias Isenberg, Neal Glew, Matthias Hauswirth, Josh Bloch, Dirk Riehle, Raimund Dachselt, Matthias Frisch, Adrian Kuhn, Richard Wettel, Kevin Andressend, Gordon MacDonald, John Newton, Ignas Kukenys, Alain Forget, Sonia Chiasson, Stevenson Gossage, Jeff Wilson, Miguel Nacenta, Nicolai Marquardt, Marian Doerk, Uta Hinrichs, Christopher Collins, Melanie Tory, Jennifer Baldwin, Philippe Kruchten, Michael Wybrow, Luc Vlaming, Andrew Clayphan, Christopher Ackad, Dominikus Baur, Victor Pascual, Michael Sedlmair, Andreas Butz, Peter Kinnaird, Mary-Beth Rosson, Steve Tanimoto, Susan Wiedenbeck, Judith Good, Martin Erwig, Sriram Subramanian, Hrvoje Benko, Patrick Baudisch, Kevin Groke, and Heidi Esplin.

Thanks to all the people in the NUI Community Group who have helped me: Jordan Hochenbaum, Owen Vallis, Memo Akten, Nolan Ramseyer, Jim Lyst, Seth Sandler, Chris Rasmussen (Fraunhofer and MT4j), and Bill Evans.

This work is supported by the New Zealand Foundation for Research Science and Technology (now called the Ministry of Business, Innovation, and Employment) for the Software Process and Product Improvement (SPPI) project, a Telstra Clear Post-Graduate Scholarship, and a Victoria University of Wellington PhD Submission Scholarship.

Finally, thanks to my family and friends for supporting me throughout this degree: Mum, Dad, Kate, Cherie, Brigitte, Brett, Rob, Raelene, Amber-Rose, Ruby-Anne, Ralphie, Toby, Frank, and James.

Publications

Materials, ideas, tables, and figures in this thesis have appeared previously in the publications below.

Peer Reviewed Conference Papers

- **Craig Anslow**, Stuart Marshall, James Noble, Robert Biddle. SourceVis: Collaborative Software Visualization for Co-located Environments. In *Proceedings of the IEEE Working Conference on Software Visualization (VISSOFT)*, Eindhoven, Netherlands, 2013.
- Keith Cassell, **Craig Anslow**, Lindsay Groves, Peter Andreae. Visualizing the Refactoring of Classes via Clustering. In *Proceedings of the Australasian Computer Science Conference (ACSC)*, Perth, Australia, 2011.
- **Craig Anslow**, James Noble, Stuart Marshall, Ewan Tempero, and Robert Biddle. User Evaluation of Polymetric Views Using a Large Visualization Wall. In *Proceedings of the ACM Symposium on Software Visualization (SoftVis)*, Salt Lake City, UT, USA, 2010.
- Haowei Ruan, **Craig Anslow**, Stuart Marshall, and James Noble. Exploring the Inventor's Paradox: Applying Jigsaw to Software Visualization. In *Proceedings of the ACM Symposium on Software Visualization (SoftVis)*, Salt Lake City, UT, USA, 2010.
- 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 *Proceedings of the Asia Pacific Software Engineering Conference (APSEC)*, Sydney, Australia, 2010.

Peer Reviewed Workshop Papers

- **Craig Anslow**, Stuart Marshall, James Noble, Robert Biddle. Exploring Collaborative Software Visualization with Multi-touch Tables. In *Proceedings of the Workshop on Collaboration meets Interactive Surfaces: Walls, Tables, Tablets, and Phones at the at the ACM International Conference on Interactive Tabletops and Surfaces (ITS)*, St Andrews, Scotland, 2013.
- **Craig Anslow**, Stuart Marshall, James Noble, and Robert Biddle. Interactive Multi-touch Surfaces for Software Visualization. In *Proceedings of the Workshop on Data Exploration for Interactive Surfaces (DEXIS) at the ACM International Conference on Interactive Tabletops and Surfaces (ITS)*, Kobe, Japan, 2011.
- **Craig Anslow**, Stuart Marshall, James Noble, and Robert Biddle. Co-located Collaborative Software Visualization. In *Proceedings of the Workshop on Human Aspects of Software Engineering (HAoSE2010)* at SPLASH, Reno/Tahoe, Nevada, USA, 2010.

Peer Reviewed Posters

- **Craig Anslow**, Stuart Marshall, James Noble, and Robert Biddle. SourceVis: A Tool for Multi-touch Software Visualization. In *Proceedings of the International Conference on Interactive Tabletops and Surfaces (ITS)*, Kobe, Japan, 2011.
- **Craig Anslow**, James Noble, Stuart Marshall, and Ewan Tempero. Visualizing the Word Structure of Java Class Names. In *Companion to the ACM SIGPLAN International Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPLSA)*, Nashville, Tennessee, USA, 2008.
- **Craig Anslow**, James Noble, Stuart Marshall, and Robert Biddle. Web Software Visualization Using Extensible 3D (X3D) Graphics. In *Proceedings of the ACM Symposium on Software Visualization (SoftVis)*, Herrsching am Ammersee, Germany, 2008.
- Bennett Thompson, David Pearce, Gary Haggard, and **Craig Anslow**. Visualizing the Computation Tree of the Tutte Polynomial. In *Proceedings of the ACM Symposium on Software Visualization (SoftVis)*, Herrsching am Ammersee, Germany, 2008.

Doctoral Symposiums

- **Craig Anslow**. Multi-touch Table User Interfaces for Co-located Collaborative Software Visualization. In *Proceedings of the Doctoral Symposium at the ACM International Conference on Interactive Tabletops and Surfaces (ITS)*, Saarbrucken, Germany, 2010.
- **Craig Anslow**, James Noble, Stuart Marshall, and Ewan Tempero. Towards Visual Software Analytics. In *Proceedings of the Australasian Computing Doctoral Consortium (ACDC)*, Wellington, New Zealand, 2009.
- **Craig Anslow**, James Noble, Stuart Marshall, and Ewan Tempero. Towards End-User Web Software Visualization. In *Proceedings of the Graduate Consortium at the IEEE Symposium on Visual Languages and Human Centric Computing (VLHCC)*, Herrsching am Ammersee, Germany, 2008.

Other

- Hien Tran, **Craig Anslow**, Stuart Marshall, Alex Potanin, Mairead De Roiste. Lessons Learnt from Collaboratively Creating Maps on a Touch Table. In *Proceedings of the ACM New Zealand Conference on Computer-Human Interaction (CHINZ)*, Hamilton, New Zealand, 2011. (short paper)
- **Craig Anslow**, James Noble, Stuart Marshall, and Ewan Tempero. Visualizing the Size of the Java Standard API. In *Proceedings of the New Zealand Computer Science Research Student Conference (NZCSRSC)*, Wellington, New Zealand, 2010.
- **Craig Anslow**, James Noble, Stuart Marshall, and Ewan Tempero. Web Software Visualization Via Google's Visualization API. In *Proceedings of the New Zealand Computer Science Research Student Conference (NZCSRSC)*, Auckland, New Zealand, 2009. (short paper)

Contents

Part I

Introduction and Background

Chapter 1

Introduction

> *"People who are really serious about software should make their own hardware."*
>
> Alan Kay. 1982

Software systems are becoming increasingly complex to design and support. Software maintenance is reported to be about 70% of the total cost of a software product [52, 307]. Understanding what causes these costs is an ongoing research problem in software engineering. Developers face the task of understanding software when they want to maintain, reuse, reverse engineer, or re-engineer a piece of software. Visualizing the source code and run-time of software can give a greater insight into the structure, behaviour, and evolution of software, to aid developers in software understanding [77]. *This thesis addresses exploring collaborative software visualization on multi-touch tables for co-located software development teams.*

Software maintenance is often a social activity and involves developers working within co-located environments (same room and time). Developers quite often work in pairs within a larger team. These pairs carry out tasks including: programming, code reviews, refactoring, and visualization of work flow. Some of these activities use digital (e.g. code) and physical (e.g. post-it notes) artifacts. Most software development tools and applications that support these activities often involve analysis and visualization features.

Most development tools and applications are designed from a single-user perspective such as Integrated Development Environments (IDEs) (e.g. Eclipse, Microsoft Visual Studio). These tools make it hard for developers to analyse and interact with software artifacts collaboratively using the same tool. For example, in pair programming there is only one keyboard and mouse for input which is controlled by the driver [22, 363]. When the observer wants to interact they have to either swap positions or obtain the keyboard and mouse from the driver.

Most software analysis applications contain a small range of visualization techniques. If applications supported multiple visualizations this would allow developers to visualize aspects of systems from different perspectives and reduce the overhead of installing multiple applications. For example, in code reviews developers often have to use multiple applications to explore different aspects of software including test coverage, class dependencies, and class diagrams.

Most computer applications are viewed on displays that are oriented vertically. The displays are usually only designed to support vertical orientation as opposed to horizontal. Developers quite often have dual vertical displays with a large number of pixels. When working in co-located teams it is hard to interact and collaborate with information across multiple digital and physical devices. For example, in visualizing work flows many development teams use physical devices like white boards and physical artifacts like post-it notes. Visualizing work flow is commonly referred to in the Agile software development community as "information radiators" [67], "visible big charts" [147], and "KanBan boards" [123]. As post-it notes are physical, manipulating and transferring this information into a digital context can be cumbersome.

Most computer user interfaces are designed for single-users and traditionally have keyboard and mouse input. The advent of new technologies like touch sensing and digital pens has enabled computers to have multimodal input for devices including smart phones, tablets, and touch screens. These devices are targeted towards single-users and are too small for a multi-user co-located environment. Large multi-user and multi-input devices that are potentially suitable for co-located collaborative software development teams include interactive surfaces like horizontal or vertical touch displays or multi-touch tables and high resolution visualization display walls.

Existing software visualization research has primarily focused on Graphical User Interfaces (GUI) and Virtual Reality [77]. To date there has been limited research on how tools support collaborative software maintenance [170, 214], and even less has explored how visualizations can support collaborative software maintenance. Storey et al. propose that collaborative software visualization can improve team software maintenance [316]. Little research has been conducted on how collaborative devices such as multi-touch tables and visualization walls help support development teams with software development and analysis tasks.

This thesis addresses exploring collaborative software visualization on multi-touch tables for co-located software development teams. We built our own large multi-touch tables and developed a collaborative software visualization application to run on the tables. We conducted user studies with professional software developers using our multi-touch table and software visualization application.

1.1 Research Context

We adopt a general definition for information visualization from Card et al. [56].

> "Information visualization is the use of computer-supported, interactive, visual representations of abstract data to amplify cognition" [56].

Software visualization is a field that is derived from information visualization and is defined as follows.

> "The use of the craft of typography, graphic design, animation, and cinematography with modern human computer interaction and computer graphics technology to facilitate both the human understanding and effective use of computer software" [312].

> "Software visualization is the application of information visualization in software engineering and can show the structure of software, runtime behaviour, and representation of source code" [77].

None of these above definitions take into consideration collaboration or multiple users. Isenberg et al. [137, 139] defines collaborative information visualization based upon the general definition for information visualization [56].

> "Collaborative visualization is the shared use of computer-supported, (interactive) visual representations of data by more than one person with the common goal of contribution to joint information processing activities" [137, 139].

We build upon these definitions for software visualization and collaborative information visualization and use the term **collaborative software visualization** to define our research context in the following way:

> "Collaborative software visualization is the shared use of computer-supported interactive visual representations of data to understand the structure, behaviour, and evolution of software by more than one person with the common goal of contribution to joint information processing activities."

1.2 Research Scope

The research topic of this thesis lies at the intersection of three main computer science research fields as illustrated in Figure 1.1: Software Engineering (SoftEng), Information Visualization (InfoVis), and Computer Support Cooperative Work (CSCW). The research scope of this thesis is a subset of collaborative software visualization aspects:

Software Engineering (SoftEng):

- Software maintenance with the goal to analyze and understand existing software systems focusing on the structure and evolution.
- Identification of possible refactoring opportunities in a software system.
- We deem other software maintenance tasks such as debugging, adding new features to a code base, and performing actual refactorings out of scope of this thesis.

Information Visualization (InfoVis):

- Representing different aspects of the software systems focusing on software metrics for structure and evolution [96].
- Presenting multiple visualizations of the software systems (e.g. using diagrams, charts, Polymetric Views [185]).
- Viewing the aspects of the software systems from different view points and orientations.
- Interacting and exploring the aspects of the software systems in the visualizations.

Computer Supported Cooperative Work (CSCW):

- Co-located environments, where several software developers work for the same organisation and in the same team.
- Developers who work together, where collaborative software understanding and programming occurs at the same time.
- Developers who share a single workspace for meetings and discussion such as a white board, interactive table, or wall display.
- Developers working in small groups typically 2-3 individuals.

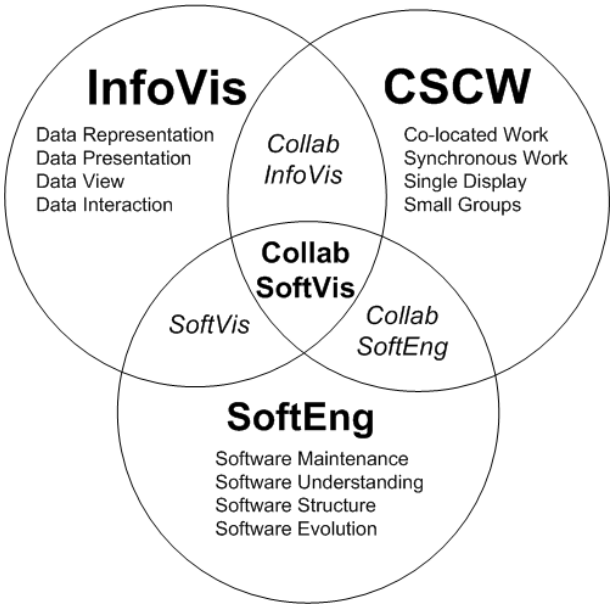


Figure 1.1: The research topic of this thesis lies at the intersection of Software Engineering, InfoVis, and CSCW. Diagram adapted from [137, 316].

1.3 Research Goals

Our research goal is to investigate how pairs from a co-located software development team might use a collaborative software visualization application on a large multi-touch table to understand existing software systems. We examine how pairs work at the same time either as a group or as individuals in parallel. In particular we wanted to address the following research questions:

Q1 What are the strengths of the visualizations and the multi-touch table?

Q2 What are the weaknesses of the visualizations and the multi-touch table?

Q3 What improvements could be made for the visualizations and the multi-touch table?

Q4 Does the multi-touch table help with team collaboration, and if so how?

Q5 Did the pairs favour working as a group or as individuals in parallel?

Q6 Which coupling categories did the participants use when interacting?

Q7 Which coupling style strategies did the participants use?

Q8 Which physical arrangement styles did the participants use with the table?

Q9 Which visualization techniques did the participants perceive to be the most effective?

1.4 Research Approach

Our research approach has been to first look at the background literature within our research scope. Based on the literature we used an existing set of design considerations for collaborative information visualization [137, 138] and applied them for designing co-located collaborative software visualizations.

Following an iterative design process we built two multi-touch tables to validate the design considerations for a physical collaborative environment. We designed a collaborative software visualization application for multi-touch tables to validate the design considerations for collaborative information visualization.

To evaluate our software visualization application we followed a qualitative grounded evaluation of information visualization process during the development lifecycle [144]. This process used mixed methods and is suited for giving an overview of a situation and to examine how and why users behave in certain environments [71]. Other researchers have adopted a similar approach to understand how groups of participants collaboratively use a multi-touch table in a co-located environment [137, 141, 291].

Our evaluation process involved conducting preliminary user studies with computer science students working in pairs during different stages of the development life-cycle of the application. We then conducted a much larger study with professional software developers. The aim of the user studies was to determine how groups and individuals use our multi-touch table and software visualization application and to assess the perceived effectiveness of the visualizations.

We used the following techniques as part of the evaluation process: observational studies, video recordings, interviews, and questionnaires. We asked participants to think aloud during our studies so that we could capture their thoughts about their actions, perceptions, and expectations regarding the application's interface and functionality [224]. Getting participants to talk about their actions and thoughts enabled us to gain insight into how each user views the computer system, identify their misconceptions, and determine which parts of the interface cause problems.

1.5 Research Contributions

The thesis contributes a richer understanding of how pairs of software developers make use of shared software visualizations on large interactive multi-touch tables to gain insight into how existing software systems are structured and how they have evolved over different versions. Specifically this thesis makes three contributions.

Designing Collaborative Software Visualizations. We built two multi-touch tables (28 and 48 inches) (§3). We conducted preliminary user studies with 18 computer science students (§5) using the 48 inch multi-touch table and prototypes of our software visualization application. The user studies led to a protocol which we used for the professional user study.

SourceVis: Software Visualization Application. We designed a collaborative software visualization application, called SourceVis, for use on large multi-touch tables (§4). SourceVis allows multiple users to interact simultaneously or separately with the table either as a group or as individuals.

Evaluation of Collaborative Software Visualization. To evaluate the large multi-touch table and SourceVis we conducted a user study (§6) with 44 software developers working in pairs performing software understanding tasks. We wanted to find out if participants preferred to work as a group or as individuals. We observed what collaborative coupling categories, coupling styles, and physical arrangement styles participants used and how much time was spent in these styles. We asked participants to provide feedback on what the strengths, weaknesses, and improvements of SourceVis and the multi-touch table were. We asked participants how the multi-touch table helped with team collaboration and what visualization techniques they perceived to be the most effective. We obtained both qualitative (§7) and quantitive findings (§8).

1.6 Key Findings

Our research revealed key findings for collaborative group work, designing collaborative software visualizations, and designing multi-touch tables.

1.6.1 Collaborative Group Work

Design for joint group work over parallel individual work. We found that all pairs preferred joint group work over parallel individual work and chose to do Section 3 of the user tasks as a group.

Support a flexible variety of coupling styles. We found that pairs used a variety of coupling styles during both joint group work and parallel individual work, but used more coupling styles when they were closely coupled. We found that pairs spent more time in closely coupled styles.

Support fluid transitions between coupling and arrangement styles. We found that when pairs were closely coupled they were more closely arranged, and when they were loosely coupled they were loosely arranged. We found that pairs regularly transitioned between many closely coupled styles and closely arranged styles, but transitioned less between loosely coupled and loosely arranged styles.

1.6.2 Designing Collaborative Software Visualizations

Design visualizations for closely coupled arrangements with rotation features. We found that visualizations were primarily viewed from the long side of the table where both participants were closely arranged next to each other. When visualizations needed to be viewed from different view points (such as in parallel individual work) participants used lightweight features for rotating visualization windows to face different directions.

Provide functionality in the appropriate locality. We found that when pairs were displaying a visualization at full screen and wanted to launch a new overview visualization they they had to navigate to the start up screen rather than having a menu on hand. Pairs, however, could launch new detailed visualizations through the pie menu on elements in a visualization. When a visualization was at full screen and a participant were on one side of the table and wanted to select an option from a menu on the other side of the table they had to ask their colleague to perform the action.

Provide consistent user interactions and visual interface design. We found that the navigation gestures behaved differently across the 13 visualization techniques in SourceVis. Some participants found manipulating visualization windows difficult. We found at times some menus were displayed at different zoom levels and occasionally displayed half off the screen. We found participants having difficulty entering text using the virtual keyboard. We found that some participants occasionally lost context of what visualization they were currently looking at and could not always remember what the visual encodings meant.

1.6.3 Designing Multi-touch Tables

Provide high resolution workspace. We found that visualization windows often overlapped each other which caused interference. We found that text was readable when visualizations were at full screen, but reading text when displayed in visualization windows was difficult as the visualization was reduced in size. Providing a high resolution workspace would reduce windows overlapping and make text easier to read.

Provide appropriate table space. We found that when pairs were performing joint group work the size of the table was appropriate. We found that when pairs were performing parallel individual work the size of the table was not appropriate as the table forced them to primarily be physically arranged at opposite ends of the table and visualization windows had to be reduced in size. Depending on the task and number of users interacting there should be appropriate table space.

Differentiate between simultaneous user interactions. We found that when participants performed simultaneous navigation gestures on the canvas of a visualization or manipulated the same element within a visualization the system was confused as to what action to perform. The system should differentiate between simultaneous user interactions.

1.7 Thesis Structure

The remainder of this thesis is structured as follows.

Chapter 2: Background. We present background literature to our research including information visualization, software visualization, collaborative software visualization, and multi-touch technologies.

Chapter 3: Large Interactive Multi-touch Tables. We present our large interactive portable multi-touch tables for co-located collaborative environments. We describe our experience at designing and building our initial early prototypes to our proof of concept working prototypes.

Chapter 4: SourceVis: A Collaborative Software Visualization Application. We present our collaborative multi-touch software visualization application, SourceVis. We describe the design, implementation, interaction features, and illustrate the individual visualizations.

Chapter 5: Preliminary User Studies. We present three preliminary user studies of computer science students using SourceVis at different stages of the development life-cycle. We present qualitative and quantitive findings from the studies. The studies led us to develop a protocol which we used for a study with professional software developers.

Chapter 6: Professional User Study. We present our user study of professional software developers using our multi-touch table and SourceVis. We describe the design, participants, procedure and limitations of our user study.

Chapter 7: Professional User Study — Qualitative Findings. We present the qualitative findings from the user study with professional software developers.

Chapter 8: Professional User Study — Quantitative Findings. We present the quantitative findings from the user study with professional software developers.

Chapter 9: Conclusions. We conclude by presenting the contributions of this thesis. We then present the key findings from this thesis. We discuss limitations of our research. Finally, we suggest directions in which our research could be extended in future.

Chapter 2

Background

Contents

In this chapter we present the background literature. We begin with an overview of information visualization (§2.1). We describe the different areas of software visualization (§2.2). We describe multi-touch hardware technologies (§2.3). We describe optical based multi-touch tables (§2.4).

2.1 Information Visualization

Card et. al [56] define information visualization as:

> "Information visualization is the use of computer-supported, interactive, visual representations of abstract data to amplify cognition."

After producing a visual representation, the following issues must be addressed: exploration, navigation, and interpolation of the data [57]. Several overviews on information visualization exist [25, 56, 62, 119, 309, 352].

The theory of the visual display of quantitative information [337] consists of principles that generate design options and that guide choices among the design options. Tufte [337] describes graphical excellence of quantitative information, as the well designed presentation of interesting data – a matter of substance, of statistics, and of design. Graphical excellence consists of complex ideas communicated with clarity, precision, and efficiency. This results in a visualization displaying the greatest number of ideas, in the shortest time and in the smallest space possible.

An early example of graphical excellence is the original London Underground map designed by Harry Beck in 1933 (see Figure 2.1(a)). Most people will use the map as a visualization tool for planning a journey from one station to another. People may memorise their route by colour or the direction of the lines involved, which is known as a cognitive map [309]. Beck based the map on electrical circuit diagrams which does not reflect the geography of the city above.

Ben Shneiderman [301] created a visual design guideline called the *visual information seeking mantra* which says show an overview of the data first, then zoom and filter, and finally show details-on-demand. He then proposed a task by data type taxonomy which has seven data types (1-, 2-, 3-dimensional data, temporal and multi-dimensional data, and tree and network data) and seven tasks which a user can perform (overview, zoom, filter, details-on-demand, relate, history, and extract). This visual information seeking mantra is one of the very few methodical guidelines for designing information visualizations and it is the most widely cited [70]. There are other user task heuristics but they are not as useful for evaluating usability, focus on low level tasks, or are domain specific [5, 6, 189, 340, 379, 380].

(a) Harry Beck's 1933 original London Underground map, reproduced by kind permission of London's Transport Museum ©Transport for London [105].

(b) Many Eyes, Barrack Obama's inaugural speech [343].

Figure 2.1: Information Visualization Examples.

2.1.1 Collaborative Information Visualization

Collaborative information visualization can occur in many scenarios. Using a space and time matrix, collaborative information visualization can be broadly categorized according to where the visualizations occur in space (distributed vs. co-located) and time (synchronous vs. asynchronous) [80, 137] (see Table 2.1). Collaborative visualization may occur on different levels of engagement with the visualizations including how users: view the information, interact or explore within a visualization, and share discoveries about the visualizations [139].

Table 2.1: Time and space matrix [80] and adapted to collaborative information visualization [137].

	Same Place (Co-Located)	**Different Place** (Distributed)
Same Time (Synchronous)	Face-to Face Conversation Meeting room	Video conference
Different Time (Asynchronous)	Sticky Shift work	Email, web based, revision control systems

For distributed collaborative information visualization tools the Web is a common platform for sharing visualizations. Toolkits such as D3 [44] make deploying visualizations over the web an easy process. Many Eyes [73, 342, 343] is a web site provided by IBM research that has collaborative visualization services, see Figure 2.1(b). The web site allows users to upload ASCII data sets, visualize them, comment on each other's visualizations, and then discuss their discoveries with other people. Other similar information visualization web tools include Swivel[1], Data360[2], and DataPlace[3].

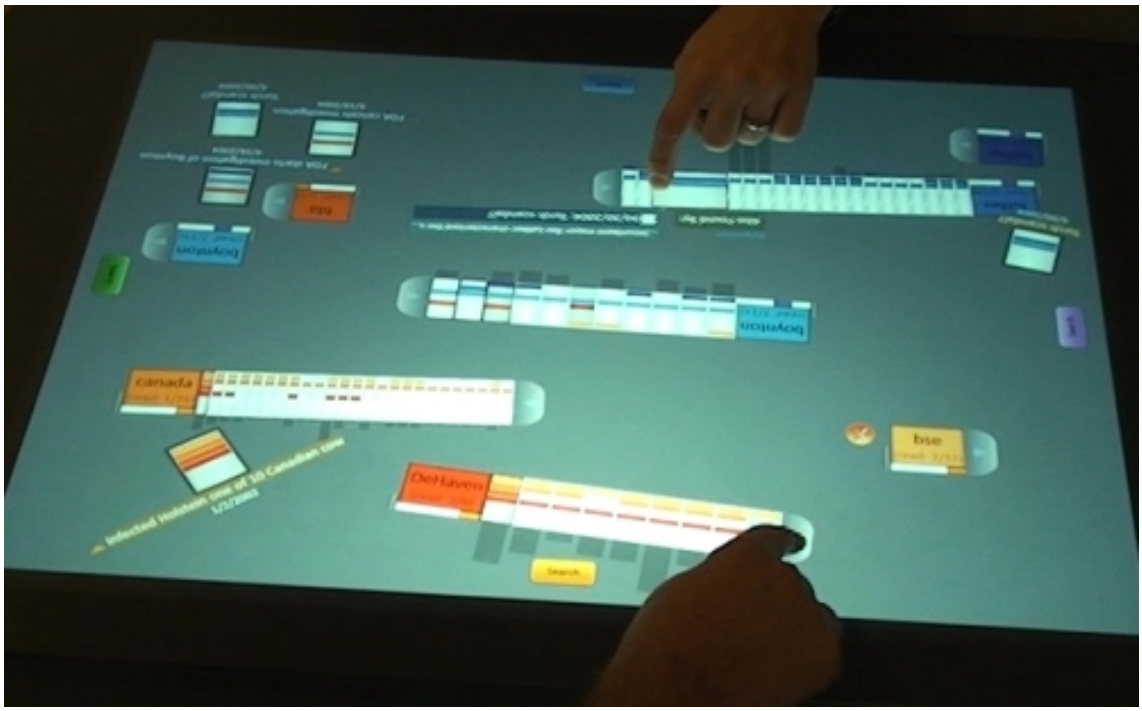
Isenberg [137, 138] suggest aspects to consider for designing co-located collaborative information visualization. For the collaborative environment the size of the display should be an appropriate for the number of users. The display configuration should accommodate groups' work practices, task, and goals. The type of input will impact the possible interactions. The resolution size needs to be considered for both the input and display. The collaborative information visualizations should support different communication strategies such as awareness cues when data changes across different views. The visualizations should support coordinating between different roles with features for sharing and rotating data views. For

[1]http://www.swivel.com
[2]http://www.data360.org
[3]http://www.dataplace.org

designing collaborative information visualizations multiple representation types should be supported to allow integrated reasoning and sensemaking. Data should be presented to allow for group access, copies of the same data, accommodation of input methods, and compensations for the display resolution. Data should be interpreted from multiple viewpoints and orientations. The visualizations should support simultaneous actions by users for interacting and data manipulation.

For co-located collaborative information visualization multi-touch tables have been explored. Cambiera [140, 141] is a system for information foraging activities on a multi-touch tabletop display (see Figure 2.2(a)). Group members can individually search for documents, browse through search results, and read documents. Cambiera includes a number of features (collaborative brushing and linking) that highlight where collaborators have found or read similar documents. Lark [330] is a system that facilitates the coordination of interactions with information visualizations on multi-touch tabletop displays using an information visualization pipeline (see Figure 2.2(b)). Hugin [164] is a graphical framework for mixed-presence synchronous collaborative visualization over multi-touch tabletop displays. Branch-Merge-Explore [209] propose a new collaboration protocol for explicitly supporting varying degrees of coupling styles in co-located collaborative visualization. They use a multi-touch tabletop display for a public view of the data and tablets for private views.

For co-located collaborative information visualization multi-display environments have been explored. Yost et al. [374] conducted an evaluation to explore the effect of using large visualization walls for information visualization. Their results showed that performance on most tasks was more efficient and sometimes more accurate because of the additional data that could be displayed, despite the physical navigation. Andrews et al. [7] conducted a study which demonstrated how large displays support sensemaking. Ball et al. [14] identified that physical navigation helped improve user performance with large displays. Bi et al. [32] discuss how interior bezels affect user behaviours, and suggest guidelines for effectively using tiled-monitor large displays and designing user interfaces suited to them. Other researchers have investigated large displays for daily desktop computing tasks [33] such as navigation tasks [319] and window and task management [280].



(a) Cambiera [140, 141].

(b) Lark [330].

Figure 2.2: Co-located Collaborative Information Visualization with Tabletop Displays Examples.

2.1.2 Evaluation of Information Visualization

Researchers have largely focused on evaluating information visualizations using quantitative approaches which focus on performance evaluations. This style of approach makes use of evaluation metrics such as task time completion and number

of errors made. These evaluations of information visualization commonly occur in controlled lab environments. These methods appear insufficient to quantify the quality of an information visualization system since users are far removed from their actual workplaces [144].

A workshop[4] aims to explore novel information visualization evaluation methods beyond time and errors, and to structure the knowledge on evaluation in information visualization around a schema, where researchers can easily identify unsolved problems and research gaps.

Plaisant [248] argues there is a need to try and ensure that the evaluations are grounded in the context in which they strive to assist users. Isenberg et al. [144] propose a grounded evaluation of information visualization as a process. They advocate for increased attention to the field of qualitative inquiry early in the information visualization development life-cycle, as it tries to achieve a richer understanding by using a more holistic approach considering the interplay between factors that influence visualizations, their development, and their use. There are several papers that propose for more qualitative evaluations and complementary qualitative and quantitative approaches for evaluating information visualizations [93, 208, 332].

2.2 Software Visualization

Diehl [77] defines software visualization as:

> "Software visualization is the application of information visualization in software engineering and can show the structure of software, runtime behaviour, and representation of source code."

The goal of software visualization is to help users comprehend software systems and to improve the productivity of the software development process [77]. Software visualization is essentially situated at the intersection of information visualization, software engineering, human computer interaction, graphics, and cognitive psychology [201]. Several overviews on software visualization exist [76, 77, 312, 378].

We give an overview of some of the tools that have been used to investigate the three aspects of software including structure, behaviour, and evolution. We then present systems that have explored collaborative software development and visualization, and some approaches to evaluation of software visualization.

[4] BEyond time and errors: novel evaLuation methods for Information Visualization - http://www.beliv.org

2.2.1 Structure Visualization

Structure refers to the static parts and relations of software. In this section we present source code and static structure, software metrics, 3D, and UML.

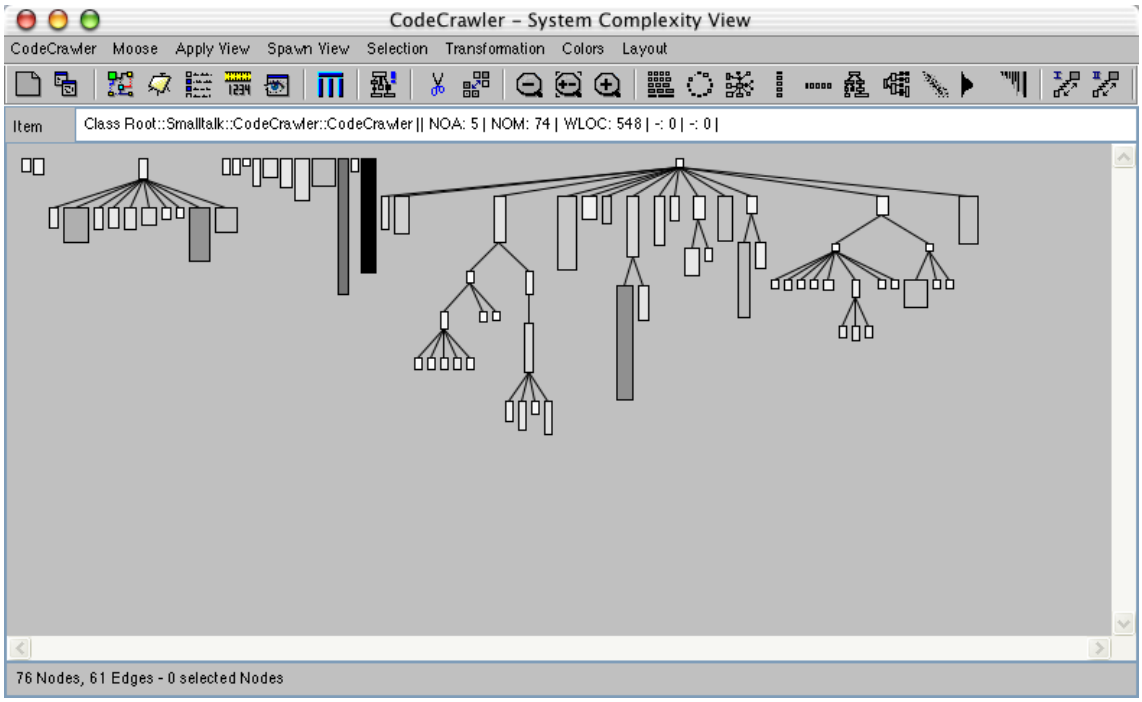
Source Code and Static Structure

Many systems have looked at visualizing the source code and the static structure of software. SolidFX [322, 324] is an IDE for reverse engineering of C/C++ programs and provides many advanced visualization techniques to explore attributes of a code base such as call graphs, metrics, and UML diagrams. J3Browser [1, 2, 3] explores Java class relations and their other tool VisMOOS [102, 103] is an Eclipse plug-in. Hierarchical Net [15, 16] visualizes the structure of large software systems as software landscapes. VizzAnalyzer [195] is a framework designed for reverse engineering. VizzAnalyzer has a built-in tool Vizz3D [233], which can be used for visualizing class and package interaction, program evolution, and program quality. NosePrints [235, 236] visualizes code smells. Enhance [297] provides information about exception handling constructs and exceptions' flow from the quantitative, the flow, and the contextual perspectives. Telea et al. [323] provide an open toolkit for visualizing telecommunications software for the purposes of reverse engineering. Clack [358] visualizes the structure of network routers. Barrio [78] visualizes class dependencies using clustering techniques.

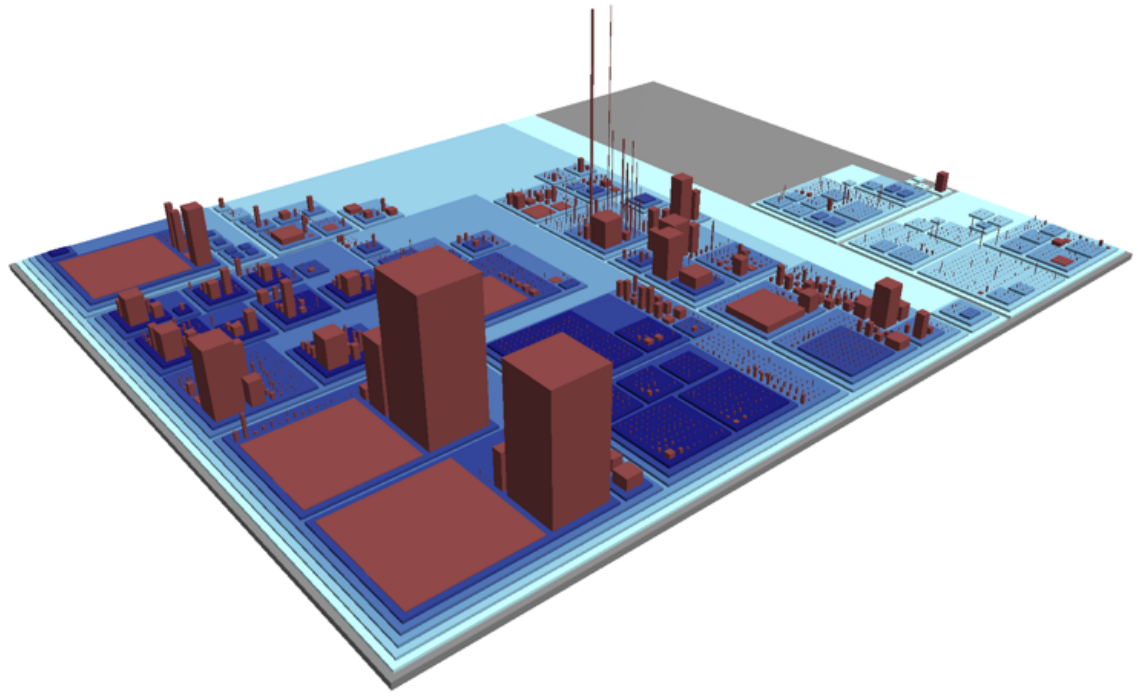
Software Metrics

A software metric measures some property of a piece of software such as the number of lines of code [96]. Applying software metrics can help determine the quality of software [187]. Chidamber and Kemerer [64] provide the most widely cited suite of metrics. These include WMC (Weighted Methods per Class), DIT (Depth of Inheritance Tree), NOC (Number of Children), CBO (Coupling Between Objects), RFC (Response for a Class), and LCOM (Lack of Cohesion in Methods).

Code Crawler [184, 186] is a language independent reverse engineering tool which combines metrics and software visualization techniques and was the first tool to use Polymetric Views [185], see Figure 2.3(a). CodeCity [360] a tool that stems from Code Crawler and Polymetric Views uses a 3D city metaphor to display additional kinds of metric information (see Figure 2.3(b)). A further study [361] extended CodeCity to focus on disharmony maps to look at the quality of the system design by focusing on design flaws. The Mondrian toolkit [213] aims to bring the Polymetric Views closer to the code by extending existing programming languages to use embedded scripts in their programs to create



(a) Code Crawler - Metrics Visualization [187].



(b) Code City - Metrics Visualization [360].

Figure 2.3: Software Structure Visualization Examples.

the visualizations as opposed to using another tool to generate the visualizations. Bergel et al. [27] extended the Mondrian toolkit by displaying dynamic information about CPU consumption in Class Blueprints. Softwarenaut [196, 197] also uses Polymetric Views but focuses on the dependencies between modules. Lagrein

is a tool that supports a number of Polymetric Views and augments them with software requirements and change history information [148, 149, 150].

Churcher et al. [66, 162] visualize object-oriented metrics with a focus on inheritance structures with cone trees, inheritance structures with metrics, hierarchies with tree maps [151, 300], web sites [120], class cohesion [65], and class clusters [136]. Various other systems and research groups have also looked at visualizing object-oriented metrics including CrocCosmos [192, 193], MetaViz [277, 278, 279], and CocoViz [37, 38, 39, 40].

3D Software Visualization

Teyseyre and Campo [326] provide an overview of 3D software visualization. Koike et al. [171, 172, 173, 175] described the significance of visualizing software information in three dimensional space and the problems of 2D visualization. This work also introduced the concept of a 3D class library browser to show method inheritance. The class hierarchy was represented as a tree in the X-Y plane and methods of each class were shown in the Z axis with the same X-Y coordinates.

Other early research has been done in 3D for visualizing Lisp programs [194], different features of a program [255, 256, 257], the layout and structuring of object oriented software in three dimensions as directed graphs (GraphVisualizer3D [97, 353, 354] and NestedVision3D [234]), SELF programs [82, 83, 122], call graphs [375], design patterns [55], and software architectures [95].

Some researchers have even explored different 3D visualization metaphors for source code comprehension. These metaphors include 3D cities (Software World [167, 168, 169], Component City [61], 3D City [232], Verso [181, 182], and CodeCity [360]), a 3D solar system metaphor [111], 3D self organizing maps [47], 3D file maps (sv3D [201, 202, 203]), and 3D computer game engines (Quake3 [178]).

UML Diagrams

Many systems have looked to reverse engineer systems in order to generate UML diagrams of large software systems [77]. Recent systems have taken a similar approach but given that software is more complex these days the software visualization research has been aimed at improving the understanding of UML diagrams by augmenting the UML diagrams with new features including areas of interest [53], textures [54], automatic layout of UML use case diagrams [91], semantic zooming [99], and digital pens and paper to create UML diagrams which can then be transferred to tabletop displays [72]. Sharif and Maletic [298] have also studied the effect of layout on the comprehension of UML class diagrams.

Visualizations of various UML diagrams such as class, object, sequence, and collaboration diagrams have been explored in 3D [75, 86, 107, 108, 210, 211, 253, 376]. Rather than representing strict UML diagrams in 3D, some research has been conducted that represents UML diagrams as 3D geon diagrams [58, 133, 134, 135]. The geon diagrams are made from 3D primitives such as cones, spheres, cylinders, and boxes. Another related area is visualizing CRC cards in 3D [285].

2.2.2 Behaviour Visualization

Behaviour refers to the execution of the program with real and abstract data. Behaviour data is collected at run time by instrumenting the code, debugger interfaces, byte code injection or extending virtual machines. In this section we cover algorithm animation and execution trace visualizations.

Algorithm Animation and Program Visualization

Algorithm animation and program visualization are useful for education and research into the design and analysis of algorithms and programs [312]. *Sorting Out Sorting* [11, 12] was the first teaching film on algorithm animation and described nine sorting algorithms.

There are various algorithm animation and program visualization systems that have been produced and some early examples include: Balsa [48, 51] (the first real-time interactive algorithm animation system), Zeus [49] (follow up to Balsa), Tango [311, 310], Polka [313] (a follow up system to Tango), Pavane [282], Tarraingím [225, 226, 227] (a tool for visualizing Self programs).

Some more recent examples include Blumenkrants et al. [36] which look at narrative algorithm visualizations, Alspaugh et al. [4] explain algorithms using scenario visualizations, SIV [118] visualizes inter-dependencies between scenarios, Lumière [28] for visualizing scheduling based algorithms, visualization of the computation tree of the Tutte Polynomial algorithm [328], and HDPV [318] for visualizing C/C++ and Java programs to understand recursion and the effect of programming errors such as buffer overflow. Python Tutor [114] visualizes the execution of Python programs online.

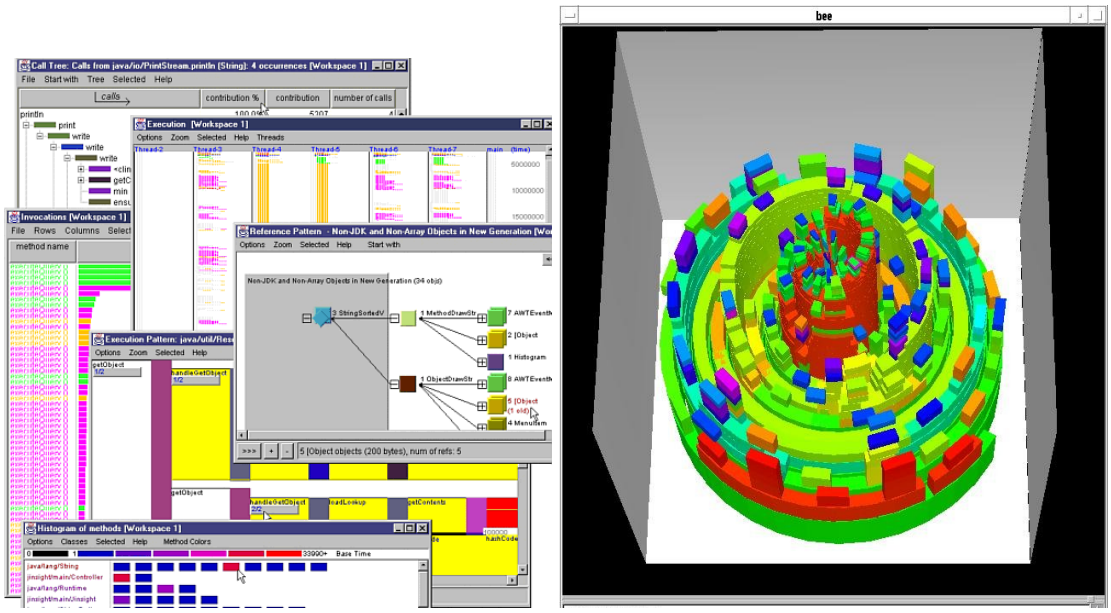
Stasko and Wehrli [314] identified the need for 3D in program visualization. They list the basic requirements for 3D computation visualization, define three categories for characterising visualizations, and discuss their system for supporting 3D animation development by programmers. Other systems have explored the use of 3D graphics for program visualization. Some of these systems include Pavane [69, 282], Polka3D [314], Zeus3D [50], 3D-AAPE [110], JCAT [220, 221], and Alice [74].

Execution Trace Visualization

Jinsight [239, 240, 241, 242, 244, 245, 296], which stems from a wide range of work from IBM, is a tool for visualizing and analysing the execution of Java programs and is useful for performance analysis, memory leak diagnosis, debugging, or any task in which a user needs to better understand what a Java program is really doing. Figure 2.4(a) shows a brief overview of some of the visualizations produced by Jinsight. Follow up systems have looked at visualizing the execution patterns of web services [243] and streaming applications [238].

BLOOM [258, 259, 260, 261, 270, 271, 272] is a framework for understanding software through visualization (see Figure 2.4(b)). BLOOM provides facilities for static and dynamic data collection and offers a wide range of data analysis. The system includes a visual query language for specifying what information should be visualized. All these are used in conjunction with a back end that supports a variety of 2D and 3D visualization strategies. Other systems include JIVE [262, 263] for visualizing Java programs in action and JOVE [273] which provides slightly more detailed information about where execution is occurring. A follow up system focuses on more specific user abstractions [264]. Another system [303] focuses on virtual machine code (IBM's Jikes RVM) and how to optimise it as opposed to user code. DYPER [265, 266, 267] does controlled performance analysis of Java systems and can obtain a variety of performance metrics including CPU usage, IO, sockets, heap utilization, memory allocations, phase analysis, and reaction analysis. DYMEM [268, 269] provides a visualization of object ownership from the memory of a running process.

Some researchers have explored using 3D for visualizing execution traces. TraceCrawler [112, 113] and CCJUN [373] explore visualizing feature traces in 3D of object instantiations and method sends to find which classes and objects are most active during the execution of a feature, what are the patterns of activity that are common in feature behaviour and which are specific to one feature. Bohnet et al. [42, 43, 344] also do dynamic analysis to look for features in C/C++ programs. Koike et al. [171, 172, 173, 175] has looked at visualizing large trace files in 3D of computer processes from a number of computers running in parallel and communicating with each other. Storer et al. [315] have developed a tool for teaching object-oriented programming concepts to introductory level computer science courses. The tool provides Java3D visualizations of the execution of Java programs including representation of classes, objects, references, and method execution. The Rube [130, 166] framework uses VRML to produce finite state machines of programs. We previously created VARE-3D [8, 9] to create visualizations from execution traces using X3D and deployed over the web.



(a) Jinsight - Debugging Visualization [239]. (b) BLOOM, Spiral views of the stack (sampled during execution) [259].

Figure 2.4: Software Behaviour Visualization Examples.

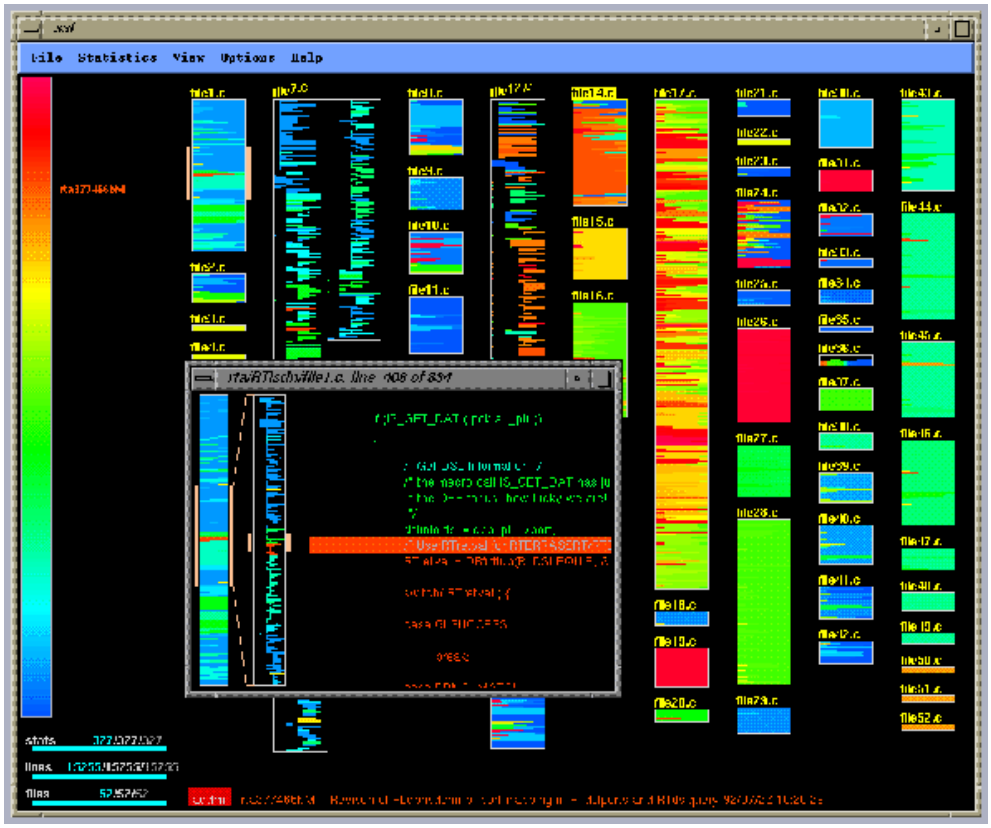
2.2.3 Evolution Visualization

Evolution refers to the process of developing software and focuses on the changes of the program code over time to improve the software and eliminate bugs. In this section we cover visualizing software archives.

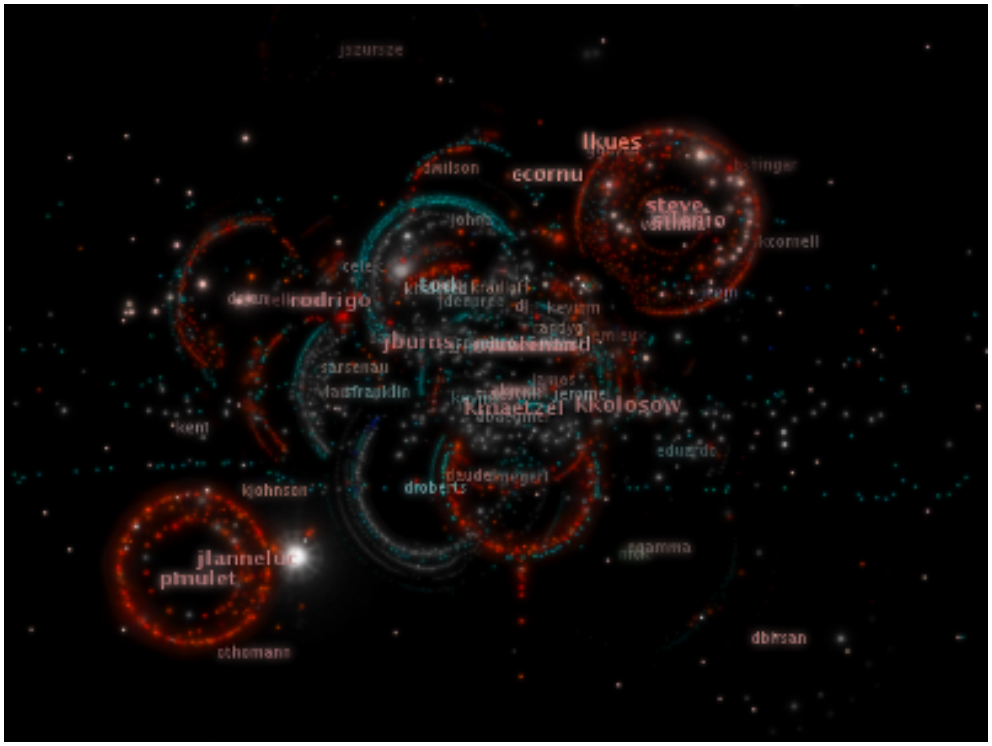
SeeSoft [92] and SeeSys [13] visualize various textual features of evolving large and complex software systems using the space filling technique which tries to convey as much information as possible with as few pixels as possible. The features include software metrics, number and scope of modifications, number and types of bugs, and dynamic program slices (see Figure 2.5(a)). The tools support a number of different views including line, pixel, file summary, and hierarchical representations.

Code Swarm [229] organically visualizes the commit history of open source projects using animations programmed in Processing [104, 254] and displayed as videos. Figure 2.5(b) shows a snapshot in time of visualizing the Eclipse IDE project. The developers and files of a project are represented as moving elements. Files are coloured differently for source code and documentation. Non-active files or developers eventually fade away. A bar chart at the bottom left is a reminder of the history of events. Another similar more recent system is Gource [59] which uses a forced directed graph layout.

Other tools also visualize software archives. Voinea et at. [345, 346, 347, 348]



(a) SeeSoft visualizing [92].



(b) Code Swarm visualizing the revision history of Eclipse [229].

Figure 2.5: Software Evolution Visualization Examples.

describe a suite of tools CVSGrab and CVSscan for mining software artifacts which display various artifacts using some advanced visualization techniques. CCVisu [29, 30, 31] use a method for computing clustering layouts of software systems for which the change history is available. WhiteCoats [212] visualizes the evolution of software from CVS repositories. VRCS [174] visualizes software revision histories using the Z axis as a time axis to represent the different revisions of each file. Panas [231] visualizes the evolution of the signatures of software binaries to find malicious code. Theron et al. [327] visualize the evolution of baselines and revisions of artifacts from software repositories. Verso [183] uses different views and animation to show structural and control version metrics of evolving software. Other evolutionary work has included the evolution of UML diagrams such as class diagrams [154, 155, 339] and model transformations [349].

2.2.4 Collaborative Software Development and Visualization

Ko et al. [170] explored how tools support collaborative software understanding for co-located software development teams but did not focus on software visualization per se. Of particular interest to us are software visualization tools and applications that support co-located collaboration.

Storey et al. [316] propose that collaborative software visualization can improve team software maintenance. They reviewed a number of existing software visualization tools and found that most of them rarely support any form of collaborative authoring and sharing of views. They recommend that designers of software visualization tools for software maintenance consider the social and collaborative aspects when building tools to help improve collaboration and usability, and adopt Computer Supported Cooperative Work (CSCW) [368] methodologies for evaluating collaborative visualizations.

Some prototypes have explored using different interactive devices to support collaborative software development. FastDash [35] is an ambient visualization system displayed on a projector for providing awareness about developer activities in software teams. IMPROMPTU [34] is a framework for sharing artifacts in a multi-display environment and a study was conducted with software development teams. CodeSpace [46] uses shared touch screens, mobile touch devices, and Kinect sensors to share information during developer meetings. CodePad [237] uses peripheral interactive devices ranging from portable tablets to tables to support developers in maintaining their concentration. CoffeeTable [117] is a visual system that uses digital pens and Wii Remotes for interaction to assist with software development processes such as what developers are working on, a summary of the architecture, and work flow activities. CREWW [45] is an interactive CRC card

system that uses Wii Remotes for collaborative requirements engineering.

Some prototypes have explored using multi-touch tables to support collaborative software development. MasePlanner [215], DAP [216], and Agile Planner [106, 350, 351] have explored planning software projects using horizontal touch tables for Agile software teams. SmellTagger [218] uses a small multi-touch table for code reviews to identify code smells in software with lightweight visualizations and software metrics. Mueller et al. [219] use a small multi-touch table to explore software modelling with CRC cards. Boccuzzo and Gall adapted their 3D software exploration tool, CocoViz [40], to multi-touch tables [41]. Soro et al. [308] compared observations of user behaviour in pair programming performed at a traditional desktop computer versus a multi-touch table. MT-CollabUML [17, 18, 19, 20] is a UML tool for multi-touch tables and a user study was conducted between the tool on a multi-touch table and desktop computer.

These multi-touch table prototypes are focused on exploration, modeling, and architecture of systems and use small tables. Any evaluation of these tables was done with non professional software developers. Our research focuses on bringing software visualization techniques such as visualizing the structure and internals of software systems to large multi-touch tables. We also exclusively evaluate our software visualization designs with professional software developers.

2.2.5 Evaluation of Software Visualization

There is no perfect method for evaluating software visualizations [121] nor any benchmarks [198] to determine how effective a software visualization is. There are various information visualization and software visualization design guidelines [223, 301], taxonomies [161, 230, 250, 249, 251, 281], and frameworks [8, 9, 84, 85, 199, 317] that can be used to evaluate algorithm animations, software visualizations, and software visualization tools.

In a survey [176, 177] based on questionnaires completed by 111 researchers from software maintenance, re-engineering and reverse engineering, 40% found software visualization absolutely necessary for their work and another 42% found it important but not critical. 7% think that software visualization is at least relevant and 6% that they can do without it but it is nice to have. Only 1% believe software visualization is not an issue at all. Finally, 4% did not answer the question. From the same survey relatively few people consider software visualization their primary research (11%) or at least a substantial part of their research (18%). Many people are doing software visualization research every now and then (20%), however most people are primarily using or integrating existing software visualization tools developed by others (33%).

Some recent studies [184, 286, 293, 294, 295] classified desirable features and lessons learned from a number of software visualization tools for software maintenance. Several features were strongly desired by all users: IDE integration, scalability, multiple views, and query support. 3D and animation were less desirable.

2.3 Multi-Touch Technologies

Multi-touch is an interactive technique that allows single or multiple users to control graphical displays with more than one finger or mouse pointer on various kinds of surfaces and devices. Multi-touch technology has been around for approximately 25 years. Bill Buxton created an overview of multi-touch systems[5]. In this section we describe some example multi-touch systems, research projects, evaluation of multi-touch technologies, and multi-touch hardware approaches.

2.3.1 Multi-Touch Commercial Systems

Exposure of multi-touch technology to consumers occurred in 2007 with the release of the Apple iPhone[6] followed by the iPad[7] in 2010. Google released Android[8] in 2007 a touch mobile phone operating system. Microsoft released the Windows 7 in 2009 and Windows 8 in 2012 which both support multi-touch interaction. HP and Dell both have laptops and desktops that support multi-touch interaction. A number of companies have released multi-touch tables which are aimed at supporting multiple users. Mitsubishi produced one of the earliest commercial multi-touch tables the DiamondTouch table[9] [79] (42 inches) in 2001 and is now sold by CircleTwelve. Microsoft produced the Surface multi-touch table (30 inches) in 2007 and now produce the Samsung SUR40 with Microsoft Pixel Sense[10] (40 inches) since 2011. Microsoft subsequently acquired Perceptive Pixel[11] which produce a multi-touch wall (81 inches x 48 inches) and multi-touch workstations (48 inches x 27 inches) [116]. Smart Technologies[12] produce touch screen products oriented towards the education market and produce a multi-touch table (42 inches).

[5] http://www.billbuxton.com/multitouchOverview.html
[6] http://www.apple.com/iphone/
[7] http://www.apple.com/ipad/
[8] http://www.android.com
[9] http://www.circletwelve.com/
[10] http://www.microsoft.com/en-us/pixelsense/default.aspx
[11] http://www.perceptivepixel.com/
[12] http://www.smarttech.com/

Ideum[13] produce a large mult-touch table (50 inches). Evoluce[14] produce multi-touch tables ranging in size (40–70 inches). Flat Frog[15] produce a small multi-touch table (32 inches). Some companies produce touch overlays which are mounted to different sized TV screens including NextWindow[16] and PQ Labs[17].

2.3.2 Multi-touch Research Projects

Jeff Han [115] (see Figure 2.6(a)) helped invigorate research into multi-touch user interfaces with his low cost multi-touch table, presented at the TED Conference[18]. Han's multi-touch user interface design philosophy is based on the following principles: that no instructions are required for touch interfaces, no manual, and almost no traditional desktop interface. Some other early research systems include: VideoPlace [179], DigitalDesk [356, 357], HoloWall [207, 275], TouchLight [364], Visual Touchpad [200], and PlayAnywhere [365].

A number of research projects have begun to explore what role multi-touch table environments can play in different domains. These domains include browsing photos [204], collaborative play [163], simulating origami [60], computer games [247], and musical interfaces with robots [126, 127].

Integrating and adapting existing technology with multi-touch tables and applications is another area that research groups are exploring. These technologies include integrating digital pens for UML diagram sketching [98], integrating mobile devices with tabletop displays [90, 367], integrating Access Grid technology with multi-touch tables [63, 131], and adapting X3D for multi-touch [156].

Some of the multi-touch research has looked at providing different kinds of input other than just fingers. These include integrating speech and gestures with geospatial systems on multi-user tabletops [336], adapting multi-finger gestures to emulate mice in a multi-touch environment [205], configuring a multi-touch wall to support both hand and foot input [288], and new ways to enter text [165].

Another input technique for multi-touch systems is physical tangible objects. The Reactable[19] [152, 153, 159, 160] (see Figure 2.6(b)) is one of the first multi-touch musical instruments. The Reactable allows many users to play electronic music collaboratively by manipulating tangible objects to control different elements like synthesizers, effects, sample loops or control elements in order to create a

[13] http://www.ideum.com
[14] http://www.evoluce.com
[15] http://www.flatfrog.com/
[16] http://www.nextwindow.com
[17] http://www.multitouch.com
[18] http://www.ted.com/talks/jeff_han_demos_his_breakthrough_touchscreen.html
[19] http://www.reactable.com

(a) Jeff Han, FTIR approach [115].

(b) Reactable [159, 152].

Figure 2.6: Recent pioneering multi-touch table examples.

composition. Lumino [21] is another system that created 3D physical tangible objects made out of glass fibre bundles which are used as building blocks for applications such as checkers. Other tangible object projects include GeoTUI [68], metaDESK [145], PhotoHelix [124], and SLAP Widgets [355].

Some approaches have tried to differentiate between multiple users touching a multi-touch table simultaneously. These approaches include detecting hand tracking [81], shadows of the arms of users [377], shoes of users [276], fingerprints [129], skin sensing [274], and assigning users to specific physical seats [79].

2.3.3 Evaluation of Multi-touch Technologies

Researchers have begun observing how people use multi-touch applications in different settings, analysing what gestures people use to control multi-touch user interfaces, and measuring the performance of the multi-touch system itself.

Peltonen et al. [246] analysed detailed observations of people using a large multi-touch display as part of the CityWall project[20]. They found that strangers acted mostly separately, but courteously, in parallel, and interacted with each other mostly after a conflict. Other research has explored collaborative coupling [320], public and private workspaces [306], casual to focused usage scenarios [284], reach in table tops [331], and personal, group and storage territories [289, 290, 292, 291].

Researchers at Microsoft [369] established a taxonomy of multi-touch surface gestures based on analysing 1080 user defined gestures. The gestures covered a wide range of user tasks including selecting, grouping, and manipulating objects. A number of other researchers have also explored analysing user gestures [94, 228, 329, 371, 372] and user gestures with virtual physical objects [125, 366].

Laurence Muller [217] conducted a series of user studies into the performance of multi-touch table surfaces compared with desktop computer mice. These studies ranged from pointing, object manipulation, collaborative sorting, and collaborative and selecting tasks. He concluded that while multi-touch is capable of performing some tasks faster than a mouse, multi-touch should not be considered a replacement. When tasks require precision, the multi-touch device shows longer task completion times with higher error rates. When the number of users were increased the test results showed that the multi-touch table had significant improvements than a single mouse.

[20] http://citywall.org/

2.3.4 Multi-touch Hardware Approaches

There are a number of multi-touch hardware approaches that can be used to build a multi-touch surface. These include resistance based, capacitive based, surface wave based, and optical based. Further details can be found elsewhere [287, 321].

Resistance Based Multi-touch

Resistance based touch surfaces have a number of layers. There are two important thin metallic layers that are separated by an insulated air gap and are electrically conductive [287]. When a finger or object touches the outer touch surface the two layers are pressed together which establishes the precise location of the touch input [287]. DualTouch [206] is a resistance based example.

Surface Acoustic Wave Based Multi-touch

Surface Acoustic Wave (SAW) based uses ultrasonic waves that pass over the touch surface. Two transducers (one receiving and one sending) are placed along the X and Y axes of the surface and are directed by reflectors. The position of a touch event can be calculated by processing the electronic signals and observing the changes.

Capacitive Based Multi-touch

Capacitive based touch surfaces were originally designed for single touch. Capacitive based touch surfaces have high clairty, durability, and reliability but are expensive to produce. The touch surface consists of an insulator such as glass and coated with a transparent conductor such as an indium tin oxide [287]. Capacitive based systems are well suited for robust environments such as public information kiosks, point of sale controls, and smart phones. Some example systems include DiamondTouch [79], SmartSkin [274], and work by Wayne Westerman [359] that led to the Apple iPhone.

Capacitive based touch surfaces can either be surface or projected based. In surface capacitive based only one side of the insulator is coated which contains electrodes that sets up a uniform electric field across the conductive later [287]. Once a finger touches the uncoated surface a capacitor is dynamically created and the sensor controller can then determine the location of the touch from the change in the capacitive as measured from the corners of the surface [287].

In projected capacitive based a very thin grid of microphone wires is installed between two protective glass layers. When the surface is touched by a finger capacitance is formed between the finger and the grid layer. The touch point can

then be calculated based on the measured electrical properties of the grid layer. The disadvantages of capacitive based surfaces are they have limited resolution, are prone to false signals from capacitive coupling, and need to be calibrated when manufactured.

Optical Based Multi-touch

Optical based multi-touch tables require the following hardware components: some kind of surface projection material, surface lighting technique, infrared (IR) light source, cameras, and a display source (e.g. data projector).

Figure 2.7 illustrates the basic arrangement of the components required for an optical based multi-touch surface. When a user touches the surface material the camera detects a change (e.g. blob) in the illuminated infrared surface and relays that information back to the tracking software. There are various options for the kind of surface material used and surface lighting technique (§2.4.1). The infrared light source is used to illuminate the surface and dependent on the surface lighting technique (§2.4.2). The optical based cameras are used for finger and object detection and only detect IR light (§2.4.3). The display source can be a

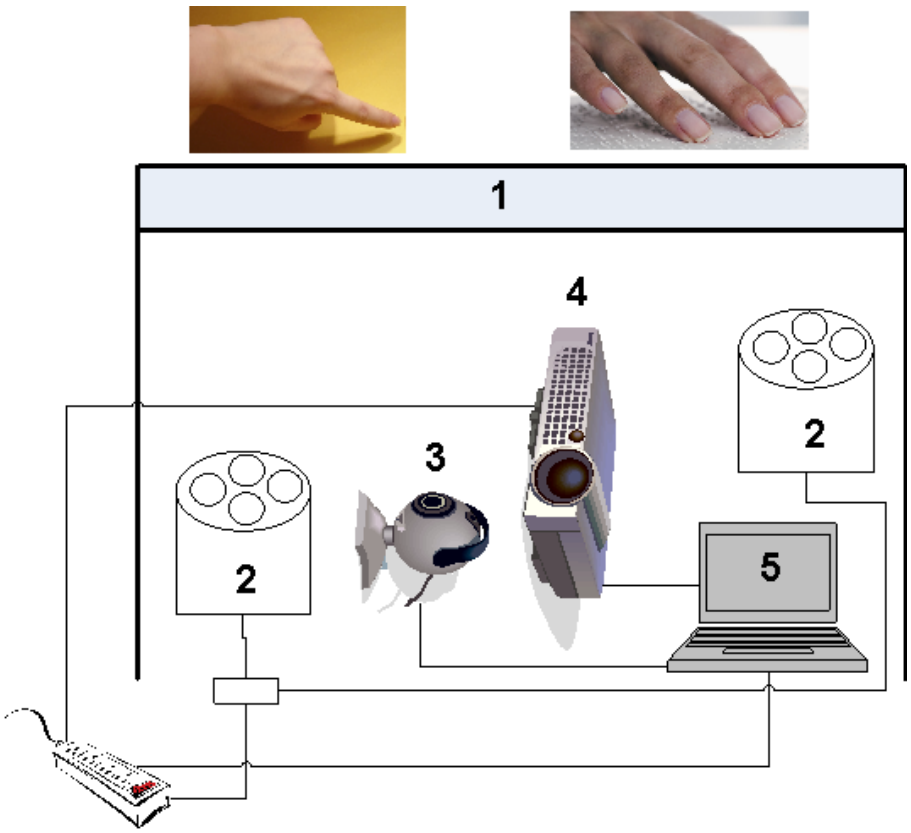


Figure 2.7: Optical based multi-touch table - basic arrangement of components.

data projector or LCD display (§2.4.4). There are some different kinds of software used for finger and object detection, and software for client applications (§2.4.5).

Resistance, surface wave, and capacitive based multi-touch hardware require industrial quality fabrication facilities and are out of scope for this thesis. Optical based multi-touch tables are cheaper to build than the other solutions, require less specialised equipment, require less electronics knowledge to build, and one can build a much larger surface.

2.4 Optical Based Multi-touch Tables

We give an overview of the the necessary hardware required to build an optical based multi-touch table, further details can be found elsewhere [287, 321] and from forums on the Natural User Interface (NUI) Community Group[21].

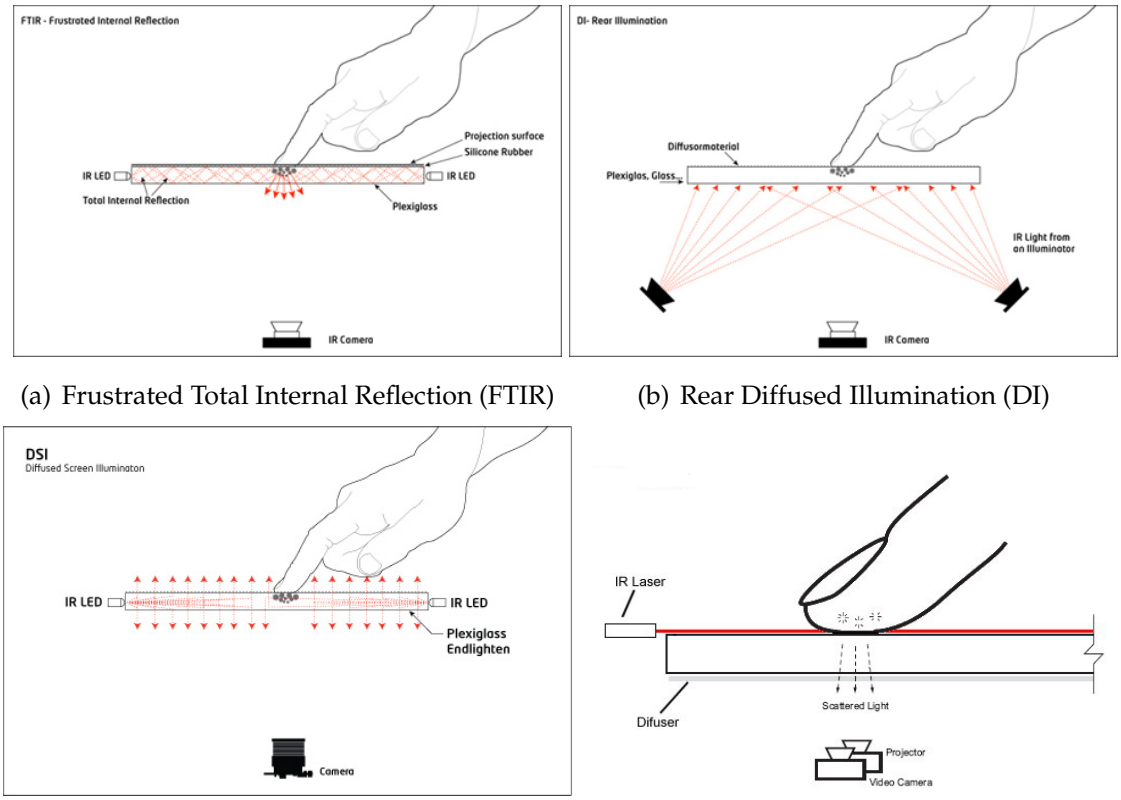
2.4.1 Surface Lighting Techniques

There are a number of lighting techniques for building an optical based multi-touch surface. Figure 2.8 illustrates some of these techniques. Table 2.2 compares the different surface lighting techniques for multi-touch tables. Depending on the requirements for a multi-touch environment, different surface lighting techniques each have their advantages and disadvantages. The surface lighting technique used impacts the infrared lighting source required to illuminate the surface (§2.4.2). Some of these surface lighting techniques include Frustrated Total Internal Reflection (FTIR), Diffused Illumination (DI), Diffused Surface Illumination (DSI), Laser Light Plane Illumination (LLP), and LCD Monitors.

Frustrated Total Internal Reflection (FTIR)

Frustrated Total Internal Reflection (FTIR)[22] is a technique created by Han [115] (see Figure 2.8(a)). The FTIR technique uses Total Internal Reflection which is a condition present in certain materials when light enters one material from another material with a higher refractive index, at an angle of incidence greater than a specific angle. The specific angle at which this occurs depends on the refractive indexes of both materials, and is known as the critical angle. When this happens, no refraction occurs in the material, and the light beam is totally reflected. The FTIR technique floods the inside of a piece of acrylic with IR light by trapping the light rays within the acrylic. When a user comes into contact with the surface, the

[21] http://nuigroup.com
[22] http://wiki.nuigroup.com/FTIR



(a) Frustrated Total Internal Reflection (FTIR)

(b) Rear Diffused Illumination (DI)

(c) Surface Diffused Illumination (DSI)

(d) Laser Light Plane (LLP)

Figure 2.8: Multi-touch surface lighting techniques [321].

light rays are said to be frustrated, since they can pass through into the contact material (e.g. skin), and the reflection is no longer total at that point. The light is then sent downwards to a camera which picks up these blobs and sends the information to the tracking software.

The clear acrylic needs to be about 8-10mm and it is important to ensure the sides of the acrylic are polished to improve the spread of the illumination. A baffle is required to hide the light leaking from the sides of the light emitting diodes (LEDs) and can be built out of any material. The more sides trapping the light will help improve the lighting conditions. A compliant layer is required to increase the brightness of the touch points. There is lots of experimentation ongoing within the NUI Community Group as what is the perfect compliant layer using many types of materials including rubber, silicon, and fabrics. A diffused layer is required to stop the projector from beaming further into the ceiling and for the camera to only see the touches, all other objects behind the surface will not be seen. There are many types of projection screens and diffusers that can be used for the touch surfaces. Depending on the requirements some diffusers are better than others.

Diffused Illumination (DI)

Diffused Illumination (DI)[23] has two variants, front and rear which use the same basic principles (see Figure 2.8(b)). The only difference between front and rear DI is where the IR lights are shone at the screen. For Rear DI IR light is shone at the screen from below the touch surface. A diffuser is placed on top or on the bottom of the touch surface. When an object touches the surface it reflects more light than the diffuser or objects in the background; the extra light is sensed by a camera. This method can also detect hover and objects placed on the surface.

Diffused Surface Illumination (DSI)

Diffused Surface Illumination (DSI)[24] uses a special acrylic to distribute the IR evenly across the surface (see Figure 2.8(c)). DSI basically uses the standard FTIR setup with an LED Frame (no compliant silicone surface is needed) and a special piece of acrylic. This acrylic uses small particles that are inside the material, acting like thousands of small mirrors. When IR light is shone into the edges of the acrylic the light gets redirected and spread through the surface. The effect is similar to DI, but with even illumination, no IR hot-spots, and has the same setup process as FTIR.

Laser Light Plane Illumination (LLP)

Laser Light Plane Illumination (LLP)[25] has an infrared light from a laser(s) shone just above the surface (see Figure 2.8(d)). The laser plane of light is about 1mm thick and is positioned right above the surface, when a finger tip just touches the laser plane the finger will register as an IR blob.

LED Laser Light Plane Illumination (LED-LLP)

LED-LLP is similar to LLP but uses LEDs to shine light over the surface. The LEDs are setup in a bezel frame similar to that of FTIR, but above the acrylic. The LEDs create a much thicker plane than that of LLP. When a user or object touches this plane of light it is reflected to the camera as a bright blob.

LCD Monitors

Alternative optical based techniques use LCD monitors to create multi-touch surfaces. The advantages of LCD monitors over the earlier discussed projector based

[23] http://wiki.nuigroup.com/Diffused_Illumination
[24] http://wiki.nuigroup.com/Diffused_Surface_Illumination
[25] http://wiki.nuigroup.com/Laser_Light_Plane_Illumination_(LLP)

techniques is that they provide a higher display resolution for a much lower cost. LCD monitors make it easier to embed within tabletop structures and therefore one does not have to worry about throw distance nor key-stoning to get a perfect displayed image. There are two common techniques for using LCD monitors with optical sensing, Bezel-IR for LCD and Matrix of IR Transceivers [287].

The Bezel-IR for LCD technique uses IR LEDS around the bezel of the LCD to shine IR light across the top of the surface of the screen. When a finger touches the surface light reflects off the finger and continues through the monitor whereupon it is captured by an IR sensitive camera. This technique can also be accomplished using IR laser LEDs. The advantages of this technique is that it is cheaper to purchase a LCD monitor than a projector, no compliant surface is required, and is scalable to 32 inch screens. The disadvantages of Bezel-IR is that no tangible tracking of objects is possible, more filtering required to eliminate false touches, and modifying the LCD is difficult. Echtler et al. [87] use an inverted FTIR approach with acrylic glass in front of the LCD display and place a camera in front of the screen.

The Matrix for IR Transceivers technique creates a matrix of IR transceivers behind the LCD screen. The transceivers contain an IR emitter and detector. The emitter pulses IR light at a frequency which the sensor can detect. Once a finger touches the LCD screen the finger reflects back the light which is detected by the sensor. It is possible to cover the entire LCD screen with the transceivers. The amount, size, and distance between the sensors determines the accuracy and resolution of this technique. The advantages of this technique is that it allows thin form factors which are similar to that of LCD screens and that tangible object tracking is possible. The disadvantages is that this technique requires specialist electronics skills and that it is not scalable beyond 32 inches due to the increase in cost and latency. An example system includes ThinSight [128, 146].

2.4.2 Infrared Light Sources

For the IR light source FTIR, DSI, and LED-LP require IR LEDs, while DI needs an IR illuminator which may have IR LEDs inside. LLP uses IR lasers. LEDs can be bought as single LEDs, LED ribbons, or within IR illuminators. Single LEDs are cheap but require being electrically wired and soldered together. The most common through-holed IR LEDs used within the NUI Community Group are Osram SFH4250 (SMD) and Osram SFH485 (5mm). LED ribbons come in the form of already soldered single LEDs and come with an adhesive side which can be stuck to the acrylic to form a continuous ribbon. IR emitters and illuminators come in grouped LEDs such as IR security lights or IR LED bars. The wavelengths of the

Table 2.2: Comparison of Multi-touch Surface Lighting Techniques [321].

Technique	**Positive**	**Negative**
FTIR	- Enclosed box not required - Blobs have strong contrast - Allows for varying blob pressure - With a compliant surface can be used with something as small as a pen tip	- Need to build LED frame - Requires compliant surface - Can't recognise objects or fiducial makers - Can't use a glass surface
Rear DI	- No need for compliant surface - Need diffuser/projection surface - Can use any transparent material (e.g. glass or acrylic) - No LED frame required - No soldering of IR LEDs - Prepackaged IR spotlights are fine - Can track objects, fingers, fiducial markers, hovering	- Difficult to get even illumination - Blobs have lower contrast - Greater chance of false blobs - Enclosed box is required
Front DI	- No need for compliant surface - Need diffuser/projection surface - Can use any transparent material (e.g. glass or acrylic) - No LED frame required - No soldering of IR LEDs - Prepackaged IR spotlights are fine - Can track fingers and hovering - Enclosed box not required	- Can't track objects and fiducial markers - Difficult to get even illumnination - Greater chance of false blobs - Not as reliable as relies heavily on ambient lighting environment
DSI	- No compliant surface required - Can easily switch between DSI and FTIR - Can detect objects, hovering, and fiducials - Is pressure sensitive - No IR hot-spots - Even finger/object illumination throughout the surface	- Endlighten acrylic costs more than regular acrylic - Blobs have lower contrast than FTIR and LLP - Possible size restrictions due to acrylic softness
LLP	- No compliant surface required - Can use any transparent material (e.g. glass or acrylic) - No LED frame required - Enclosed box not required	- Can't track objects or fiducial markers - Not pressure sensitive - Can cause occlusion if only using 1/2 lasers where light hitting one finger blocks another finger from receiving light
LED-LP	- No compliant surface - Can use any transparent material (e.g. glass or acrylic) - No LED frame required - Enclosed box not required	- Hovering might be detected - Can't track objects or fiducial markers - LED frame (soldering) required) - Only narrow-beam LEDs can be used, no ribbons

IR LEDs suitable for most cameras are in the range between 780–940 nanometers with 850nm being the most common.

For DI and FTIR it is important to achieve even IR illumination on the surface to allow the camera to detect all areas of the surface effectively. This can be accomplished by using wide angle LEDs and bouncing IR light of the insides of the table. If IR LEDs are pointed directly at the surface this can cause IR hotspots in certain areas which prevent the camera from detecting any fingers or objects.

2.4.3 Cameras

In an optical based multi-touch environment a camera is needed to detect fingers and tangible objects touching the surface. The camera needs to be modified to see only the IR spectrum so that it will only see fingers or tangible objects and not images from the projector. Basic web cameras can work for multi-touch but since they block out IR light they must first be modified to detect only infrared light. This can be done by removing their IR block filter and adding an IR bandpass filter. Basic web cameras usually have the IR block filter as a small component that can be removed but more expensive cameras have the filter attached to the lens. The purpose of the IR bandpass filter is to allow only light from a specific wavelength to pass through such as 850nm. Since bandpass filters are usually quite expensive an alternative is to use overexposed developed film negatives.

The greater the resolution of the camera the more pixels can be used to detect fingers or tangible objects. To maintain the precision of finger and object detection and tracking low resolution cameras such as 320 x 240 pixels are suitable for small surfaces but larger surfaces require a resolution of 640x480 or greater. To handle basic detection and manipulating objects cameras with a minimum frame rate of 30 FPS is recommended, but to get better performance for tracking of fast movements such as gestures and dragging objects rapidly 60 FPS is recommended [287, 321].

It is important for the lens of the camera to be able to see the whole table surface in order to detect fingers and tangible objects. Depending on the distance of the camera from the touch surface a lens with the appropriate focal length must be selected. Some cameras can use mounts to interchange between different lenses. If a lens has a low focal length this may affect the image distortion and hence make the calibration of the tracking software more complex.

Cameras with FireWire connections are more expensive than USB web cameras. FireWire cameras have a number of benefits over USB web cameras. They provide the least overhead for transferring the image to the computer, have a higher frame rate, can capture larger image sizes, and have less overhead for the driver due to less compression.

2.4.4 Display Sources

The data projector is used to display the visual output from the client software running on the computer. Two types of projection techniques include Liquid Crystal Displays (LCD)[26] and Digital Light Processing (DLP)[27]. LCD are made of up of a grid of dots that go on and off as needed and is the same technology as used in computer display screens. This technique is very sharp and has a very strong colour. DLP works by the use of thousands of tiny mirrors moving back and forth and then colour is created by a spinning colour wheel. DLP projectors have very good contrast ratio and are small in physical size.

The brightness of projectors are measured in ANSI lumens[28] and most projectors range between 1000-3000 lumens. The larger the number of lumens the brighter the image will be. In some cases a bright image can produce hot-spots so a not so bright projector maybe required.

The throw distance of the projector affects how far the projector needs to be from the projection surface. If a multi-touch table is to be created in a tightly confined space then a mirror can be used to bounce the image onto the projection surface. An alternative is to use a more expensive and less available short throw projector which can produce a much larger image at a closer distance. Using a short-throw projector makes the design of the table simpler and alleviates any complications with bouncing images off mirrors and ghosting issues.

The resolution and aspect ratio of the projector affects the shape of the projection surface. It is important to use the projector at the native resolution as this achieves the best display results. Common resolutions of projectors used in multi-touch projects described within the NUI Group Community include 1024x768 (4:3 aspect ratio) and 1280x800 (8:5 aspect ratio).

2.4.5 Multi-Touch Detection Software

We now discuss the software architecture for multi-touch software applications and describe some multi-touch vision detection software.

Multi-touch Detection Software Architecture

Figure 2.9 shows the software architecture for a multi-touch environment. The protocol used for detecting touch events (either human or by tangible objects) and transferring them to a client application is TUIO (Tangible User Interface

[26] http://en.wikipedia.org/wiki/LCD_projector
[27] http://en.wikipedia.org/wiki/Digital_Light_Processing
[28] http://en.wikipedia.org/wiki/Lumen_(unit)

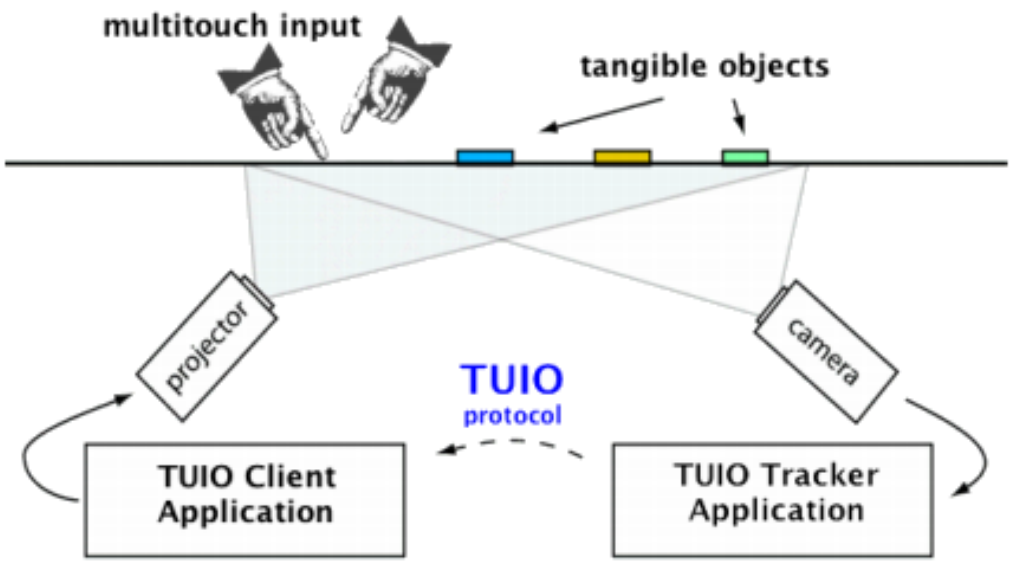


Figure 2.9: TUIO Architecture [158].

Objects)[29] [26, 157, 158]. The tracker and client applications can be located on the same machine if required. The TUIO protocol defines common properties of controller objects on the table surface and finger gestures performed by the user.

The TUIO protocol is based on OpenSound Control (OSC)[30] which is a protocol for the communication between controllers and sound synthesizers. OSC was primarily designed as a replacement for MIDI in order to overcome the bandwidth and speed limitations of this standard protocol for digital musical instruments. The transport method most commonly implemented is UDP and TUIO uses port 3333 for delivery of binary OSC data. Flash does not support UDP so an alternative TUIO/LC (shared memory) method has been implemented within a dedicated Flash/AS3 library. There also exists an additional TUIO/TCP transport method.

Echtler and Klinker [89] present a more comprehensive multi-touch software architecture with an implementation called Tangible Interactive Surfaces for Collaboration between Humans (TISCH)[31] [88]. Figure 2.10 shows the four layers of the architecture, where all data between layers is transported using UDP. The hardware abstraction layer takes raw data from the input of hands, fingers, and tangible objects and generates data packets containing these input positions. The transformation layer then converts the data into screen coordinates and outputs transformed data packets. The interpretation layer reads the input positions and converts them to gesture events. Finally, the widget layer registers for gesture events and generates the visual output to the user.

[29] http://www.tuio.org
[30] http://opensoundcontrol.org
[31] http://tisch.sourceforge.net/

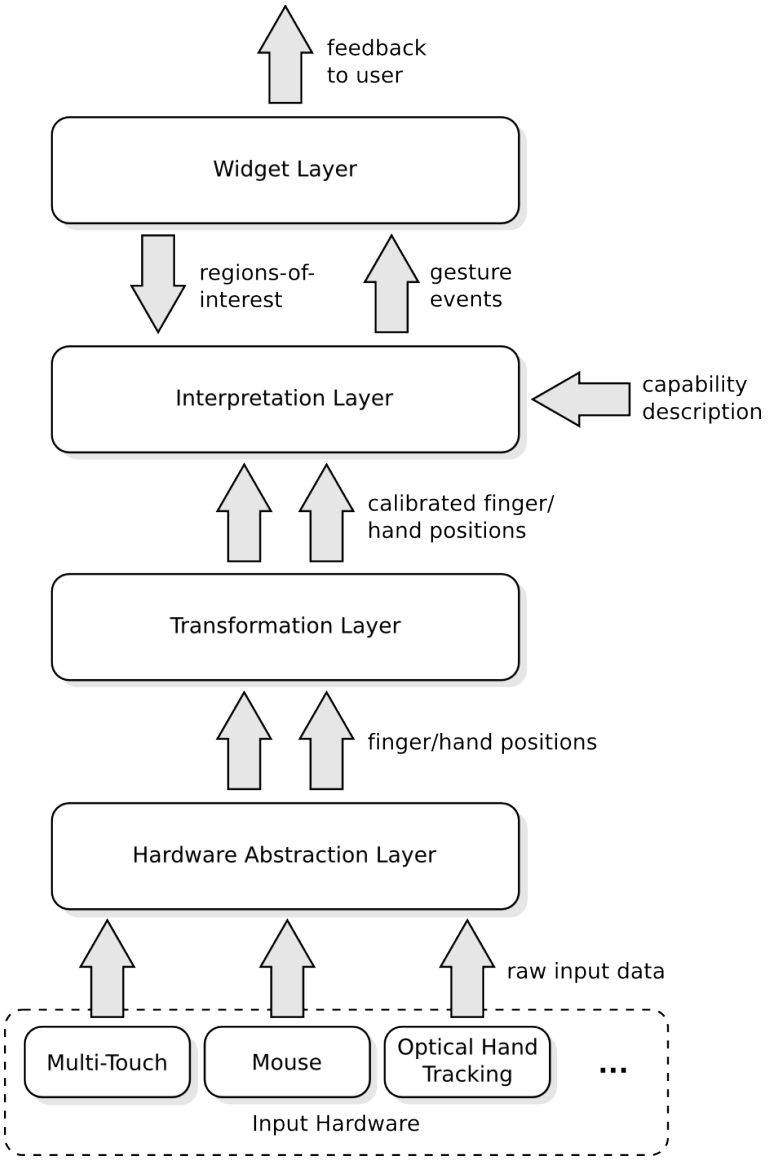


Figure 2.10: Echtler and Klinker [89] multi-touch software architecture.

Vision Detection Software

There exist some blob detection and finger/object tracking software that can be used for a multi-touch user interface. Some of the more popular multi-touch software include Community Core Vision (CCV)[32] (formerly known as tbeta), reacTIVision[33] [26, 157], Touchlib[34]. Others include Touch[35], BBTouch[36], Bespoke Multi-touch Framework[37], LightTracker [109], EquisFTIR [304, 370], and MPX [132].

[32]http://ccv.nuigroup.com/
[33]http://reactivision.sourceforge.net/
[34]http://nuigroup.com/touchlib/
[35]http://gkaindl.com/software/touche
[36]http://benbritten.com/software/bbtouch-quick-start/
[37]http://www.bespokesoftware.org/multi-touch/

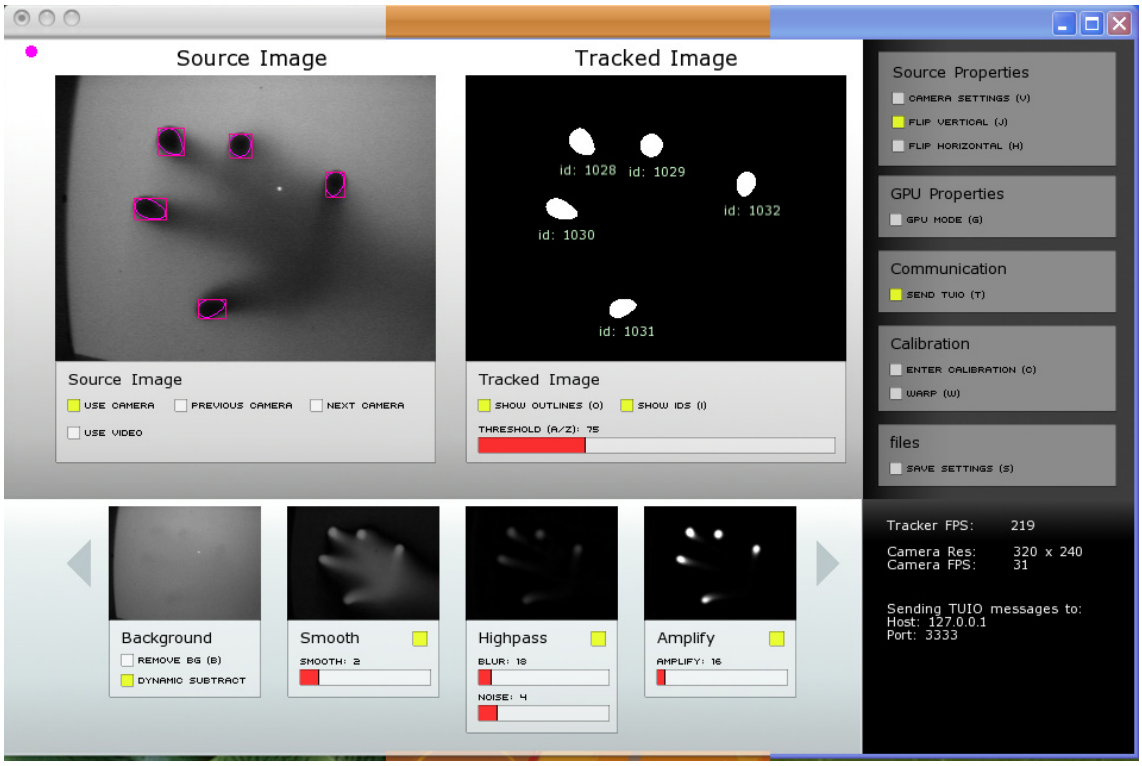


Figure 2.11: CCV multi-touch detection and tracking software.

CCV is an open source cross-platform solution for computer vision and machine sensing. CCV takes a video input stream (e.g. from web cameras) and outputs tracking data (e.g. coordinates and blob size) and events (e.g. finger down, moved, and released). The events and tracking data can then be used to build multi-touch applications. Various web cameras and video devices are supported. CCV can accept connections from TUIO and OSC applications. CCV supports the following multi-touch surface lighting techniques: FTIR, DI, DSI, and LLP.

Figure 2.11 shows a screen-shot of CCV with the source image of the raw video from a camera on the left and the tracked image on the right after image filtering. ID numbers are present for each tracked blob. There are a number of filters and settings that can be adjusted which have an effect on the quality of the blob detection. The filters are required to be set differently according to the surface lighting technique being used and the physical lighting environment. CCV has a calibration process which calibrates the software with the camera and projector. The calibration process aligns touch points on the screen in a grid formation and then a user manually touches each individual calibration point. The reason for calibration is to gain accurate detection of touch events with respect to the image being displayed by the projector.

reacTIVision [26, 157] is an open source cross-platform computer vision framework for the fast and robust tracking of fiducial markers attached onto physical objects. It also supports multi-touch finger tracking but this was added later. reacTIVision has been developed as part of the reacTable project [152, 153, 159, 160] (see Figure 2.6(b)). The software comes with a set of fiducial symbols which are a set of patterns that the camera sees and the software responds to actions upon seeing the symbol on a fiducial marker. The symbol labels can be attached to any object users want to track.

Touchlib is a library for creating multi-touch user interfaces and was one of the first libraries. Touchlib was originally designed for Windows but there have since been ports to MacOSX and Ubuntu. Touchlib has limited support for fiducial marker identification and tracking. Touchlib supports the TUIO and OSC protocols. Touchlib works with most types of web-cams and video capture devices. Touchlib does not provide a developer with any graphical or front end capabilities, it simply passes the touch events to a client application.

2.4.6 Multi-Touch Application Software

Several general purpose and domain specific programming languages support the TUIO protocol for writing multi-touch client applications. These programming languages include Action Script 3, C, C++, C#, Flash, Java, ObjectiveC, Processing, Python, and Smalltalk. More information can be found elsewhere [287, 321].

There are number of open source toolkits for building multi-touch applications: Multi-touch for Java (MT4j)[38] [188], TUIO Zones[39] a Processing library, Kivy[40] a Python library, OpenFrameworks[41] a C++ library, Cinder[42] a C++ library, and Libavg[43] a C++ library. There are some commercial toolkits for building multi-touch applications. Diamond Spin[44] is a toolkit for the Diamond Touch table [299]. The Microsoft Surface SDK[45] is a toolkit for the Microsoft Pixel Sense table. GestureWorks[46] is a C++ and Flash toolkit for the Ideum table. There are various commercial multi-touch software applications such as Omnitapps[47] for content management in presentation displays and Emulator[48] a DJ application.

[38] http://www.mt4j.org
[39] http://jlyst.com/tz
[40] http://kivy.org/
[41] http://www.openframeworks.cc/
[42] http://libcinder.org/
[43] http://www.libavg.de
[44] http://diamondspin.free.fr
[45] http://www.microsoft.com/en-us/pixelsense/softwareplatform.aspx
[46] http://gestureworks.com
[47] http://www.multitouch-software.com
[48] http://www.smithsonmartin.com

There is some software for simulating multi-touch user interfaces on desktops which can be used for development purposes. They essentially simulate mouse click events as touch events. These simulators include: SimTouch[49] which uses the Adobe Air run-time and provides a transparent background that touch events can be simulated on, TUIO Simulator[50] which is designed for simulating reacTIVision applications, QMTSim[51] which is built using the Qt toolkit, and BSQ Simulator[52] for converting TUIO messages to Windows 7 touch messages.

2.5 Summary

In this chapter we have presented background material to this thesis. We gave an overview of information visualization, collaborative information visualization, and evaluation of information visualization. We described software visualization systems and techniques based on three areas: structure, behaviour, and evolution. We presented collaborative development and software visualization and evaluation of software visualization systems. We presented multi-touch technologies including systems, research projects, evaluation of multi-touch technologies, and approaches to building multi-touch hardware. We gave an overview of building optical based multi-touch tables, multi-touch detection software, and multi-touch application software.

In the next chapter (§3) we describe our experience of building large interactive portable multi-touch tables from scratch for co-located collaborative software visualization.

[49] http://code.google.com/p/simtouch
[50] http://reactivision.sourceforge.net/#files
[51] http://code.google.com/p/qmtsim
[52] https://code.google.com/p/bsqsimulator

Part II

Hardware and Software Infrastructure

Chapter 3

Large Interactive Multi-touch Tables

Contents

In this chapter we describe our experience of building large interactive portable multi-touch tables from scratch for co-located collaborative software visualization. We illustrate from our initial early prototypes to our working prototypes. We first experimented with various techniques to understand the technologies behind touch screens (§3.1). We then built a medium sized (28 inches) multi-touch table prototype (§3.2). Finally, we built a much larger sized (48 inches) multi-touch table (§3.3).

3.1 Multi-Touch Prototypes

In order to construct a multi-touch table we first needed to understand the technology behind touch screens. We experimented with two multi-touch designs based on work by Seth Sandler[1].

3.1.1 MT Mini

The first prototype was a multi-touch pad called *MT Mini*[2]. Figure 3.1 shows our prototype which consists of an empty cardboard box, acrylic glass surface with drafting paper stuck to the top, an unmodified web camera, and CCV (§2.4.5) running on a laptop. There was no displayed image nor any infrared illumination.

The MT Mini gave us an idea of what a large multi-touch screen was like. The MT mini was essentially a large track pad as opposed to a multi-touch table, so users could not manipulate images on the surface. The MT Mini was hard to configure for different physical lighting environments. It was hard for users to manipulate objects on a separate display screen when the object being manipulated was not visible on the physical touch surface.

3.1.2 MT Biggie

The second prototype was a medium sized multi-touch table called *MT Biggie*[3]. Figure 3.2 shows the parts used in our MT Biggie. We used a computer desk with

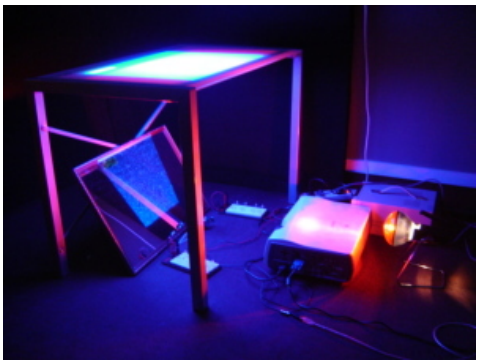
[1] http://sethsandler.com
[2] http://sethsandler.com/multitouch/mtmini/
[3] http://sethsandler.com/multitouch/mtbiggie/

(a) MT-Mini Touch Pad. (b) MT-Mini Web Camera.

Figure 3.1: MT-Mini Touch Pad prototype.

a frosted glass top (originally purchased from a local furniture company) which was approximately 1200mm wide and 800mm high. The frosting proved to be too opaque for the IR lights we used to shine through. The issue with glass and touch is that we wanted the solution to be robust, and eliminate the possibility of smashing the glass surface. We tried using safety glass with drafting film, tracing paper, transparent projection paper, and different thicknesses of paper. These approaches seemed to work better than our frosted glass table as they were not as opaque to IR light. The drafting film worked best. Finally, we tried clear acrylic glass with the different kinds of film and paper and this made our prototype safer. The downside to all of these solutions though was that the diffused material was not diffuse enough and that either the glass or acrylic glass would suffer IR hot-spots, where fingers were not tracked on the surface.

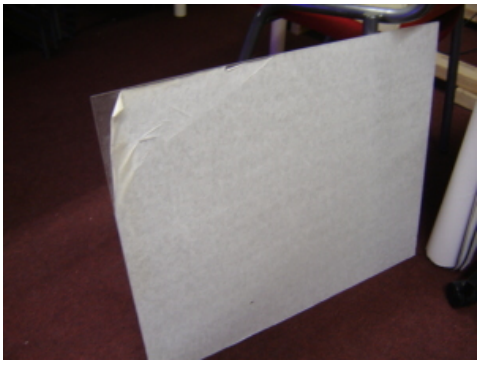
We used a basic web camera with a band pass filter lens to detect IR images. The camera supported up to 30FPS, but this was too slow for tracking fingers. We needed at least 60FPS to effectively track fast finger movements. We built two electronic circuit bread boards each with eight IR LEDs and powered by a bench top power supply. These IR lights ended up not dispersing the light evenly over the surface as they were too directional. We also tried an IR heat lamp, but this was too bright and made the room much warmer.



(a) Glass table.

(b) Safety glass.

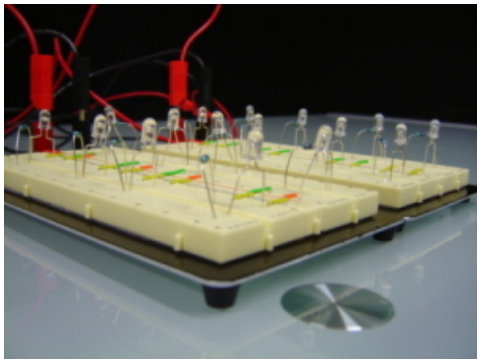
(c) Tracing paper.



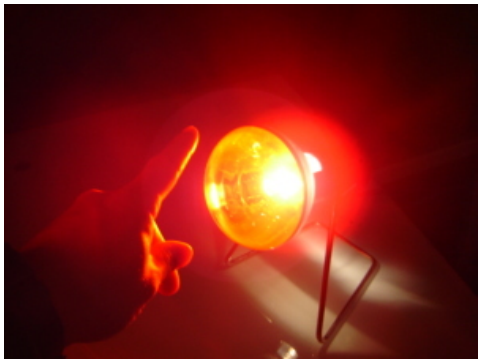
(d) Clear acrylic glass.

(e) Basic web camera.

(f) IR band pass filter lens.



(g) IR LED lights on bread board.



(h) IR heat lamp.

Figure 3.2: MT Biggie - parts.

3.2 Black Multi-touch Table

Based on our earlier experience we built the Black Multi-touch Table as a proof of concept prototype, which has a display screen size of 28 inches (see Figure 3.3). Figure 3.3(a) shows the demo photo browser application implemented in Flash that comes packaged with CCV. The image shows two users collaborating on the table to browse through a range of photos. One user has zoomed in on an image while the other user is rotating another image.

3.2.1 Physical Table Frame

Upon deciding not to use the glass computer desk from the MTBiggie for the table frame we had the option of either making a frame or adjusting an existing table. Since we wanted to create a proof of concept quickly we decided to adjust an existing table (see Figure 3.3(b)). We acquired a portable steel trolley table that had wheels from our university's central IT services department to act as the table frame. We modified the table by removing the shelves and the top surface. We then ground away the steel pieces that held these wooden shelves in place as they were an obstruction. We extended the table by adding in a separate compartment at the back that housed the computer, data projector, power plugs, and wires (see Figure 3.3(c)). The size of the table after the extensions was 760mm wide, 980mm high, and 850mm deep. We painted the table black so that it was of uniform colour and blended with other office furniture.

3.2.2 Surface Lighting Technique

For surface lighting we followed a Rear DI approach (§2.4.1) as we wanted a complete enclosed box that was portable. For the surface diffuser material we used a special piece of acrylic, called Plexiglas RP 99561 which is designed especially for rear projection displays. The 99561 material is 3mm thick. We added five millimetres of clear acrylic glass on top of the 99561 material to give it support, so that the table does not flex when a user puts their hands in the middle of the surface (Figure 3.3(d)).

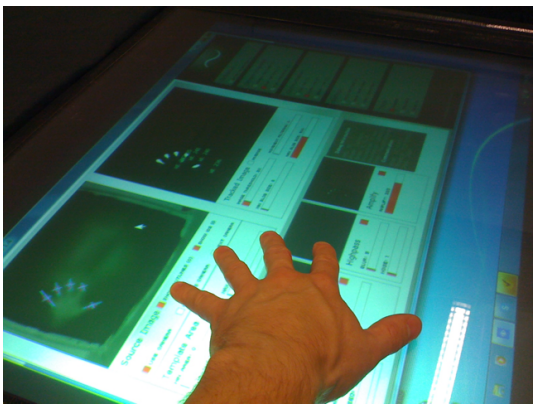
3.2.3 Infrared Lighting Source

For the IR lights we initially used two IR security spotlights purchased from a local electronics company. They provided enough IR light at 850 nanometers. We added a power source and had to slightly modify the lights by hiding the daytime detection sensor with tape. Rear-DI requires an enclosure for the IR light. We

(a) Demo test of photo browser application.

(b) Front of table.

(c) Back of table.



(d) Surface of table with CCV.

(e) Inside of table.

Figure 3.3: The Black Multi-touch Table.

initially used some old black cloth material which was wrapped around the sides of the table and later added in wooden sides where one side acted as a sliding door to access the internals. To provide an even spread of light we subsequently used four IR LED bars each with 12 LEDs at 850nm instead of the IR security spotlights and placed the bars around the inside of the table (Figure 3.3(e)). The IR LED bars emitted light at a 110 degree viewing angle as opposed to the spotlights which were more directional and had a much smaller viewing angle. We also painted the inside of the table white to help disperse the IR light.

3.2.4 Camera and Lens

For the camera we purchased a modified Sony Play Station 3 (PS3) camera with a m12 lens mount, 3.6mm lens, and a 850nm band pass filter which is compatible with the IR lights. We also purchased a range of lens to enable positioning the camera at different distances from the surface. We used a case to house the camera to point vertically as opposed to the OEM horizontal case. In order for our camera to work with CCV we required customized device drivers for which we used Code Laboratories CL Eye Driver[4].

3.2.5 Display Source

For the display source we used a Sony VPL-PX11 XGA data projector at 1024x768 (4:3) resolution which we acquired from our department. The display surface was 420mm by 570mm which is approximately 28 inches. The projector was mounted vertically and slightly angled to beam an image down onto a mirror which then reflects the image back up onto the surface material located above (Figure 3.3(e)). We only had to use minor vertical key-stoning to get a perfect rectangular image on the surface, but that required many hours of adjusting different parts of our frame. We also digitally flipped the image horizontally and vertically so that the user can see the displayed image the correct way up when facing the table front on. The projector was held in place by a steel frame mount bracket with machine screws from behind, and elastic stretch cords from the front.

Given the size of the table, the amount of heat generated by the projector and lights we had to provide enough ventilation so that the projector and other electronic components did not overheat. We created holes at the top of the sides and added louvers for the heat to rise and escape. We added silver ventilation ducting to the projector and vented the heat out the bottom of the table. We also added a fan which brought colder air from outside the table to inside.

[4]http://codelaboratories.com/products/eye/driver/

3.2.6 Computer Hardware

We initially used a MacBook Pro with OSX (10.5.8), 2.4GHz Intel Core Duo, 4GB RAM, and GeForce 8600M GT with 256MB RAM running CCV 1.2 to drive the multi-touch applications. We subsequently upgraded to a faster machine so that we could use later versions of CCV (i.e. 1.3-1.5). This was a Dell Optiplex 990 with Windows 7 Enterprise 64 bit, Intel Core i5-2500 3.3 GHz, 4GB Ram, and integrated graphics on the Intel motherboard. We connected desktop speakers to provide audio support and a wireless network adapter for Wifi connectivity.

3.3 Blue Multi-touch Table

The resolution of the Black Multi-touch Table was unfortunately too low to do anything other than entertainment based applications. Even though the Black Table was medium size it was still a bit small for multiple users to do collaborative work. Hence we decided to build the Blue Multi-touch Table which was much larger and had a display screen size of 48 inches (see Figure 3.4). Figure 3.4(a) shows two users interacting with the MSA Fluids example from MT4j which shows different coloured particles being displayed when a user touches the surface [188].

3.3.1 Physical Table Frame

We designed a steel table frame which we outsourced to our workshop department to build. The table was 1200mm wide, 920 mm height, and 780mm deep (see Figure 3.4(b)). The table had four wheels and a base made of ply wood which was 10mm thick. The front side of the table was fixed with two sliding doors on the sides. The sides of the table were made of 5mm glossy shiny hardwood. The back of the table was an adjustable black cloth material that was attached to the top, tucked underneath the bottom, to provide sufficient ventilation.

3.3.2 Surface Lighting Technique

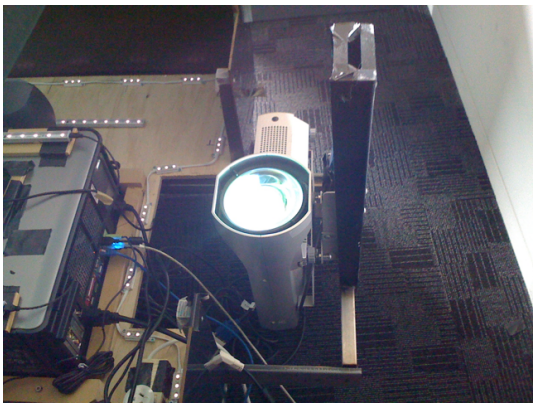
For the surface lighting we again followed a Rear DI approach (§2.4.1). The surface diffuser material used was rear projection acrylic called Plexiglas RP 7D006 and was 5mm thick (see Figure 3.4(c)). This material had a greater viewing angle compared with the 99561 material. We had to import a large sheet of the material from the manufacturers in Germany and cut it to the size of the table. We put a piece of clear acrylic (3mm) on top, but the touch sensing was more effective with only the 7D006. The down side to using just the 5mm 7D006 was that the middle of the surface flexed with hard hand pressure in the middle.

(a) MSA Fluids application [188].

(b) Front of table.

(c) Surface of table with maps.

(d) Inside of table.



(e) Projector on sliding draw.

Figure 3.4: Blue Multi-Touch Table.

3.3.3 Infrared Lighting Source

We used were four IR LED bars each with 12 LEDs at 850nm. The bars were placed evenly inside the middle of the table. One 25 module IR LED set (each module had three LEDs) was also used and placed around the perimeter of the inside of the table. The four IR bars and module set emitted light at a 110 degree viewing angle which gave us the best spread of IR light (see Figure 3.4(d)).

3.3.4 Camera and Lens

The camera was a modified PS3 camera with a CS lens mount, 850nm band pass filter, and housed inside a case (see Figure 3.4(d)). The lens was a Theia 1.67mm which was adjustable, and had a wide angle 113 degree field of view. Hence we could place the camera relatively close to the surface and still view a large surface size. There was no distortion so the camera could detect effective blobs even in the corners of the display which can sometimes be problematic. We again used Code Laboratories CL Eye Driver for our camera to work with CCV.

3.3.5 Display Source

The mirror on the Black Table caused ghosting effects on the displayed image so we eliminated the mirror in the Blue Table and used a short throw data projector that pointed directly at the surface. We used a Sanyo PLC-WXL46 at 1280x800 (16:10) resolution. At the time of purchase there were only eight projectors that were short throw and at that resolution available according to Projector Central[5]. At the time of submission of this thesis there were 83 projectors that were short throw and 1280x800 resolution. The display surface is 1077mm x 673mm which is approximately 48 inches. The projector was mounted to a sliding draw and a steel post frame and slid out for when in use (see Figure 3.4(e)). The projector can also be slid inside the table for transportation so that it can fit through doors.

3.3.6 Computer Hardware

We used a Dell Optiplex 760 with Windows 7 Enterprise 64 bit, Intel Core 2 Duo 3.0 GHz, 8GB Ram, and ATI Radeon HD 3400 graphics card. We needed a separate graphics card and at least 8GB of ram to successfully run both CCV and the multi-touch applications. We connected some desktop speakers to provide audio support and a wireless network adapter for Wifi connectivity when the table did not have access to ethernet connectivity.

[5]http://www.projectorcentral.com

3.4 Discussion

We now discuss the most important aspects of the tables considered against the design considerations for collaborative information visualization in co-located environments [137]. These aspects include: the size of the display, display resolution, how the display can be configured, and different types of input. We also discuss other aspects such as portability, how the tables performed in the physical environment, and the cost to make the tables.

3.4.1 Display Size

The Black Table is of medium size (28 inches) and is not effective for sharing information on the display amongst users when orientated horizontally. The Blue Table is large (48 inches) which makes it easier to share information amongst users as there is a much larger display space, and more physical space around the table. As our applications are focused on the work place, a vertical orientation of the surface would enable more information to be shared at a distance. Making the horizontal display much larger will increase the height of the table but also make objects harder to reach and manipulate [331].

3.4.2 Display Resolution

The Black Table is low resolution at 1024x768 and the Blue Table medium resolution at 1280x800. Both of these resolutions are significantly smaller than what users currently get from their laptop and desktop screens. The resolution on both tables impacted precise and accurate manipulation of data for work based applications. It would be preferable to design tables that had higher resolution (e.g. full high definition 1920x1080) and be of large size. This would lead to building tables that could utilize large TV display screens.

3.4.3 Display Configuration

Each of the multi-touch tables only had one shared display surface that users could interact with. A separate display screen was actually connected to the computer or the projector as output and was located on another physical table next to the touch tables. When the display screen was connected to the computer the screen was used to help configure the table and for providing additional display information. When the display screen was connected to the projector the screen just mirrored what was already being displayed by the projector.

3.4.4 Input Type

Teams are increasingly using different kinds of input for group work. Our prototypes only support touch, keyboard, and mouse input. We are also interested in tracking objects other than fingers such as fiducial markers. Rear DI is the only optical based technique that supports detecting fiducial markers. Ideally we would like to add support for digital pens as this is likely to increase precision and accuracy for work place tasks. Others have explored commercial digital pens and touch displays [100].

Both tables support multiple users interacting at once since we used the CCV software. The Black Table is of medium size and best supports multiple users for entertainment based applications. When the Black Table is used for work based applications it is better suited for a single user due to the display screen size. The Blue Table is large and easily allows multiple users to interact. The projector side of the table is not intended to be used as most of our applications are designed with a lot of text and orientating data continuously can be become quite burdensome. Hence only three sides of the table are primarily available for use.

Our tables do not distinguish between users interacting with the table. That is when multiple people touch the table at the same time the detection software can not detect who is touching what location. If we could distinguish between users that would provide a better user experience and allow us to track different users in the software. Others have tried distinguishing between users by detecting shadows of a user's arms based on their physical position [377].

Users interact with our tables by standing up. We did not provide any furniture for users to sit on when interacting with our tables. This was due to the tables being boxes and there was no space for users to position their legs underneath the tables. Using stools next to the tables would have been problematic as the position of the users would be far away from the table and interaction would be awkward.

3.4.5 Portability

Both tables can easily be moved to different locations within a building as they are on wheels. They were both designed to fit through doorways that are less than 800mm wide. The Black Table does not require calibration of CCV after relocation. The Blue Table, however, requires re-calibration of CCV after relocation as the projector has to be slid inside for transportation and then slid out for display, even though it is slid out to the same position each time. Both tables have also been transported to other locations within our city and required shuttle van transportation. Neither table is suitable for overseas travel. Others researchers have explored creating more portable and collapsible solutions for overseas travel [126].

3.4.6 Performance

An issue with Rear DI setups is the tables being exposed to natural light and fluorescent lights which makes it harder for detecting and tracking fingers with CCV. Because our tables were enclosed and the type of IR lights we used, the tables could be operated in rooms that had fluorescent lights, and directly below fluorescent lights as well. When working in rooms that have fluorescent lights or large amounts of natural light we needed to reconfigure CCV by increasing the amplify setting to effectively detect fingers and objects. Both tables worked more effectively in dimly lighted rooms.

3.4.7 Cost

Large multi-touch tables are expensive to mass manufacture and purchase due to the high quality fabrication facilities required. At the time this project started the cost of commercial multi-touch tables was greater than $30,000 New Zealand Dollars (NZD). Due to this large expense we sought to build our own low cost large portable multi-touch tables. The total cost of parts for the Black Multi-touch Table was $1,674 NZD and the Blue Multi-touch Table $6,193.50 NZD. Building the Blue Table required significantly more effort than the Black Table. These costs did not include the many hours to design and assemble the parts.

Table 3.1 outlines the break down of costs for the different parts used in the hardware for both the multi-touch tables in NZD. All parts were purchased from Peau Productions[6], local hardware, electronic and projector stores, and Dell. The specialist multi-touch materials (cameras, filters, cases, lens, IR LED light emitter bars and set) were purchased from Peau Productions. Some parts were designed and made from scratch. The parts that were purchased from other countries have had their values converted into NZD. For the Black Multi-touch table most of the parts were acquired from our department or local companies. The cost of the Blue Table could have been reduced if we had built the table frame ourselves as there was significant time spent implementing our design.

[6]http://www.peauproductions.com

Table 3.1: Parts used in the Black and Blue Multi-touch Tables.

Part	**Black Table**	**Blue Table**
Table Frame	Free	$1,050.00
Surface Material	Free	$800.00
IR Lights	$250.00	$510.00
Camera, Filter, and Case	$188.00	$188.00
Lens	$88.00	$354.00
Projector	Free	$1,700.00
Projector Frame Mount	Free	$431.50
Computer	$838.00	$1,100.00
Other Materials	$310.00	$60.00
Total Cost (NZD):	**$1,674**	**$6,193.50**

3.5 Summary

In this chapter we presented our large portable multi-touch tables for co-located collaborative work. We described our experience from building early prototypes to our two multi-touch table prototypes (28 and 48 inches). Based on our experience we recommend future multi-touch tables to focus on higher resolution displays (i.e. greater than 1280x800), portability, allowing for vertical or horizontal orientation of the display, and providing seating so users do not become fatigued during prolonged use.

In the next chapter (§4) we present our software visualization application that we developed for use on the Blue Multi-touch Table.

Chapter 4

SourceVis: A Collaborative Software Visualization Application

Contents

In this chapter we present our contribution *SourceVis* – which is a multi-user interactive collaborative software visualization application for use on large multi-touch tables within co-located environments. SourceVis is an application for exploring, and visualizing two key areas of software systems: the *structure* and *evolution*. In this chapter we give an overview where we describe the design, implementation, and interaction features of SourceVis. We then present the suite of 13 visualization techniques that SourceVis supports which cover three areas of software visualization: exploration, structure, and evolution.

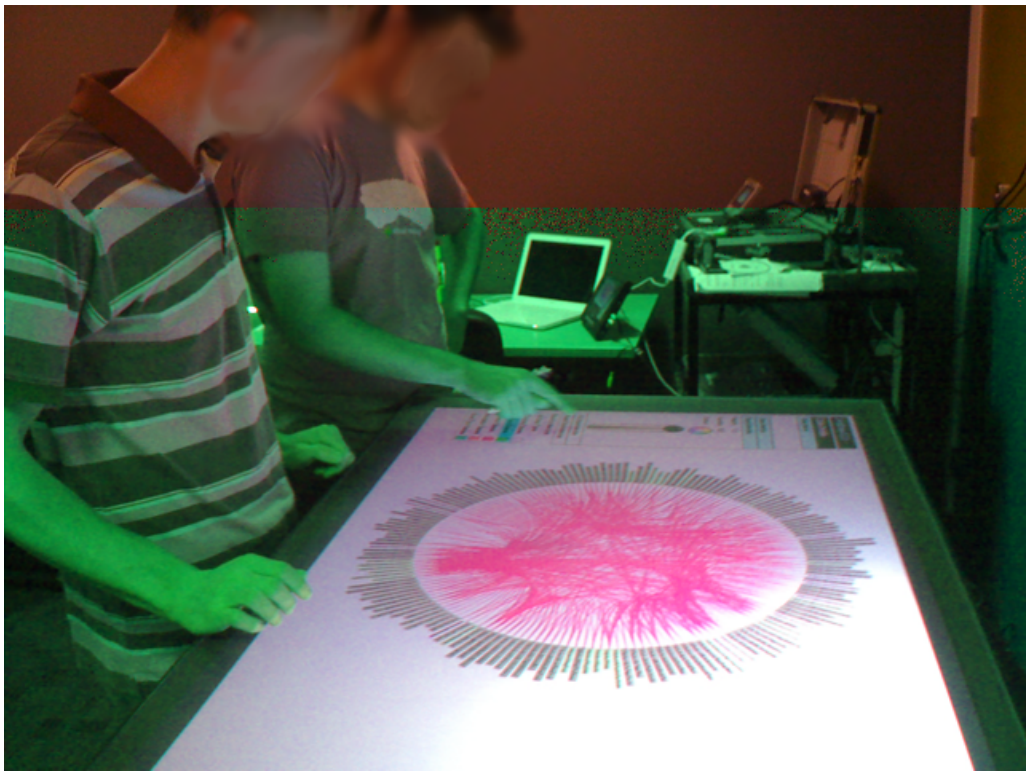
4.1 Overview

Many researchers have found that developers often ask questions about how software has changed such as "when, how; by whom; and why was this code changed or inserted?" [101, 170, 333, 334, 302]. SourceVis aims to help support answering some of these questions using visualization techniques.

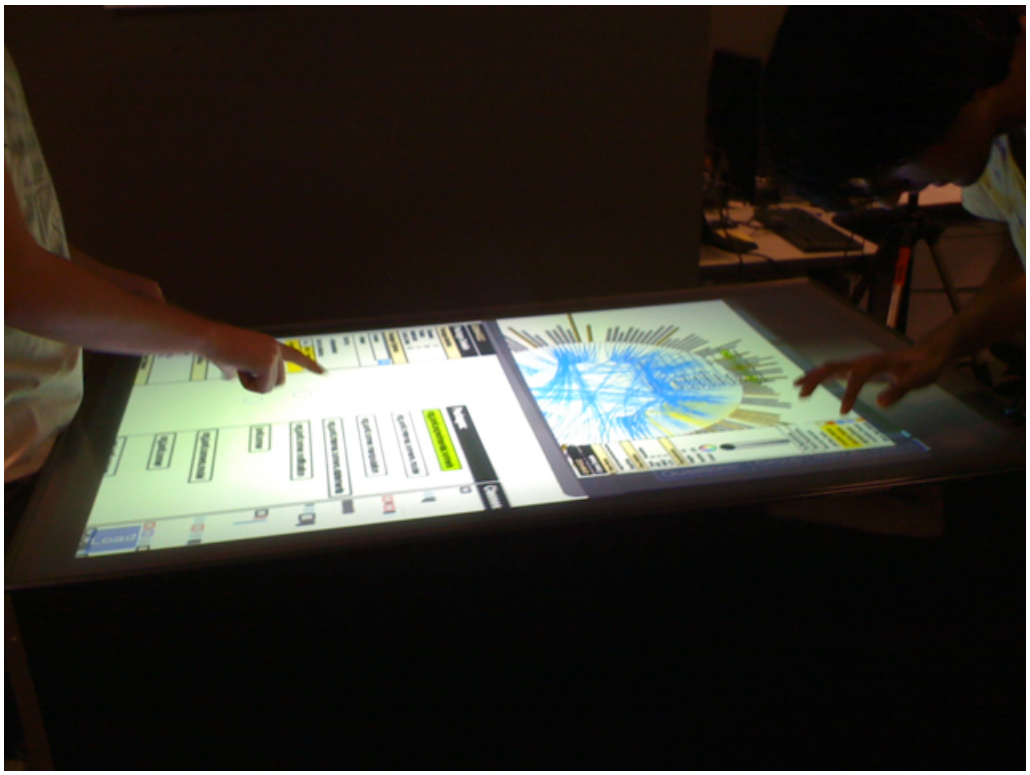
SourceVis builds upon previous research and solely focuses on software visualization whereas previous research did not focus on software visualization with large multi-touch tables (§2.1.1 and §2.2.4). We focus on visualizing the structure and internals of software systems while previous research focused on exploration, modeling, and architecture of systems on smaller touch tables.

SourceVis is an interactive collaborative application for exploring, and visualizing the structure and evolution of software systems. SourceVis supports multiple visualization techniques. The visualizations can help identify what parts of a system are large and likely need to be refactored, dependencies between entities, and how the structure of systems have evolved. The aim of SourceVis is to help developers working in co-located teams to explore a software system. SourceVis is designed for multiple users to interact simultaneously or separately.

SourceVis has been designed to be used on our Blue Multi-touch Table (§3.3) and built upon MT4j [188]. Visualizations can be displayed at full screen, within scalable windows, and at any orientation on the table. Multiple visualizations can be displayed at once. Figure 4.1(a) shows two users interacting with a full screen visualization. Figure 4.1(b) shows two users working on two different visualizations in separate windows at the same time.



(a) Two users working together with a visualization which is displayed at full screen.



(b) Different visualizations displayed in separate windows.

Figure 4.1: SourceVis - multi-user collaborative software visualization.

4.1.1 Design

To address the design considerations for collaborative information visualization in co-located environments [137, 138] we present the design goals of SourceVis.

Representation. When visualizing software it is important to be able to represent different aspects of the underlying source code. We designed SourceVis to represent the software structure and how the structure has changed over time by using software metrics, dependency, and inheritance information.

Presentation. When exploring a software system it is important to have a large range of visualizations to present different views of a system. We designed SourceVis to support multiple techniques for visualizing the structure and evolution of software systems. We selected a sample of existing techniques from the information visualization and software visualization literature based on our prior experience and empirical user studies [56, 77]. For the visualizations to support multi-touch and multi-user interaction we had to adapt the techniques in a number of ways such as input from finger and hand touch gestures. We designed SourceVis in a modular fashion to allow new visualizations and extensions to be added in the future.

View. When developers are working collaboratively with an application it is important for them to be able to view the visualizations clearly irrespective of where they are physically located. We designed SourceVis to display visualizations on large shared interactive horizontal surfaces that are portable walk-up user interfaces and within a co-located environment. Visualizations can be displayed at full screen, within scalable windows, and at any orientation on the horizontal plane. Multiple visualizations can also be displayed at once. SourceVis has been designed for use on our horizontal Blue Multi-touch Table (§3.3).

User Interaction. When developers are working together it is important for both to be able to interact easily with an application at the same time. SourceVis has been designed for multiple users to interact at the same time using multi-touch within a co-located environment.

4.1.2 Visualizations

The visualization techniques selected have been adapted from existing techniques in the information and software visualization literature [56, 77], and from our previous visualization wall user study. In our previous visualization study which evaluated one of the more common visualization techniques Polymetric Views and found that the System Hotspots View was very effective at identifying large classes [10]. The visualization techniques in SourceVis have been modified substantially to support multi-touch and multi-user interaction.

SourceVis contains thirteen visualizations grouped into three categories. These categories are: Exploration, Structure, and Evolution. The visualizations allow users to explore a system (Exploration), see the structure of a system (Structure), and see how the structure of a system has evolved over time (Evolution).

The exploration category contains visualizations that show a list of entities, metrics about systems, packages, and classes, and vocabulary employed in entities. The lists are common to looking at documentation such as JavaDoc but enable touch scrolling. Popular wordle [341] like techniques have been adopted to highlight large entities. Charts have been used as they are common in spreadsheet applications and are well known information visualization techniques [337, 338].

The structure category adapts Polymetric Views [185] (software metrics based) to multi-touch including the System Hotspots View and Class Blueprint. Polymetric Views are a common technique within software visualization and many other tools implement these techniques hence why we decided to adapt them to multi-touch (§2.2.1). The structure category also contains graph based visualizations as node link diagrams to show dependencies between class entities which is a common technique for showing relationships among items [56].

The evolution category shows how a system has evolved over time focusing on structural changes such as the size of different versions of a system, packages, and classes using Polymetric View encodings and charts. Again we use the common technique of Polymetric View encoding to represent the different entities in a system. We describe each of the individual visualizations from the three categories in more detail later (§4.2, §4.3, §4.4).

4.1.3 Interaction

Users first load a system before any visualizations can be displayed. They load a system by selecting a system name from the load menu at the bottom of the startup screen (see Figure 4.2). Only one system can be loaded at a time. Visualizations can then be launched by tapping a visualization icon. Tapping and holding on a visualization icon brings up a help message that describes the purpose of the visualization and what will be shown when launched as seen on the right hand side of Figure 4.2 which shows a red bordered box with text for the Toxicity Chart. The visualizations in the startup screen are overview visualizations and users can then select individual entity visualizations from these overviews to get more specific details about different entities as denoted with an asterisk in Figure 4.3.

Individual visualizations are displayed in rotatable and scalable windows. Having individual visualizations in separate windows allows for multiple visualizations to be displayed on the tabletop at once either next to each other,

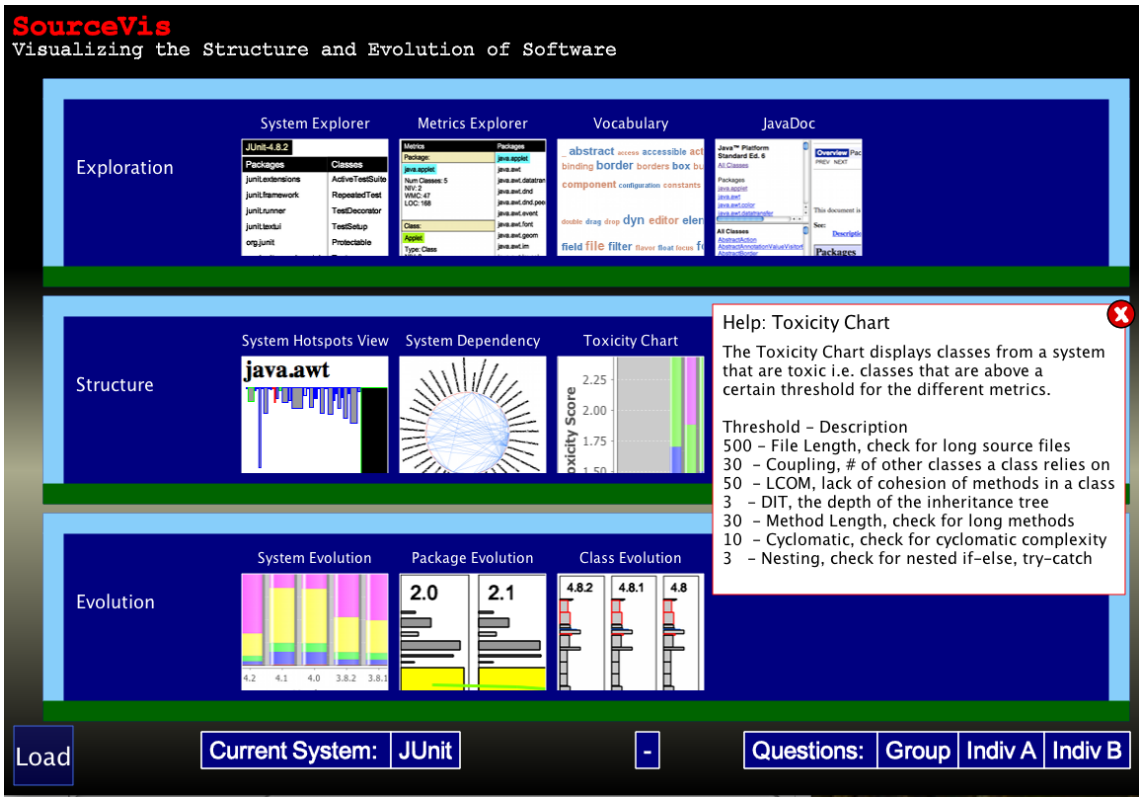


Figure 4.2: SourceVis startup screen.

overlapping one another, or within side another visualization window. When a visualization is first started it is displayed in a separate window. On the border of the window there are options located on the right hand side to close or maximize the visualization. Once a visualization is displayed at full screen there is an option in the bottom right of the display as seen in Figure 4.5 that allows closing (denoted as a cross) or making a visualization smaller (denoted as windows overlapping).

Users can interact with elements in visualizations by tapping for select, dragging elements with one finger, and rotating and resizing elements with two fingers (of either the same hand or different hands). Tapping and holding an element displays properties about that element in help message boxes, or for element entities, displays options in a pie menu. Multiple entity elements can be grouped by drawing a shape around them using a lasso gesture. These grouped elements can then be moved the around the visualization. Zooming uses a pinch gesture with one or two hands, and panning by scrolling with two fingers in any direction on the canvas of the visualization.

As SourceVis is designed for a horizontal display it is important for users to be able to move around the tabletop and orient the information to where they are physically standing. On the start screen users can orient the categories and the visualization scene icons and text to each of the four directions of the tabletop.

The current direction is represented by the solid green bar on the border of the category. Figure 4.2 shows the categories pointing to the bottom of the display. Tapping one of the cyan coloured borders orients the categories in that border direction. Since visualizations are displayed in separate scalable windows this allows users to manipulate windows to orient visualizations to the location of where they or their colleague are physically standing.

For a multi-user scenario a visualization could be displayed at full screen where one user is focused on the overview of a system while another is focused on a certain aspect of the same visualization (see Figure 4.1(a)). For another scenario users could be viewing two separate visualizations at the same time. One visualization could be at full screen while the other a smaller visualization inside or on top of the larger one. Alternatively two separate visualizations could be displayed at opposite ends of the table for the different users (see Figure 4.1(b)).

There are a number of menus available for each visualization including: system version, pie, and options menus. The system version menu displays the name of the current version being displayed in the top left of the visualization. Tapping on the current version makes a scrollable list of all versions for that system. When a new version is selected the data for that system is then loaded and the visualization updated. This only affects the current visualization being displayed. System evolution type visualizations do not have the system version menu as these visualizations contain all versions of a system.

The pie menu is displayed when a user performs a tap and hold gesture on an individual entity such as a package or a class (see Figure 4.5). The pie menu has options for displaying metrics properties about an entity (e.g. name, version, type, and specific metrics, i.e. the light yellow shaded box in Figure 4.5), or options for displaying a new individual visualization type for that entity (e.g. Class Blueprint, Class Dependency, Class Evolution). Only one instance of each individual visualization type from the pie menu can be displayed at once.

The options menu displays information about entities such as metrics and is predominantly located the left hand side of the visualizations. There are features for manipulating the element entities in the visualizations such as sorting (e.g. alphabetical, and ascending or descending by metrics), filtering (e.g. by class type or slider to show entities greater than a threshold), and searching (e.g. via displaying keyboard). To allow for more screen real estate for the visualizations the options menu and sub menus can be hidden by tapping on the menu labels. For the options menu (coloured black with white text, e.g. Options, Package Options, Class Details) when tapped are collapsed into a single plus sign, while the sub menus (coloured light yellow with black text, e.g. Class Options, Class Metrics) hide the contents in that sub menu.

4.1.4 Architecture

Figure 4.3 illustrates the architecture of SourceVis. In order to build visualizations from the underlying software structure we created a meta-model to represent entities from the Java programming language. We created a JavaMetricsEntity Interface which other classes can implement. The methods the interface contains include setting and getting properties about a Java entity such as the name of the entity, type, and which software version the entity belongs to.

We have concrete classes for: JavaMetricsSystem, JavaMetricsPackage, JavaMetricsClass, JavaMetricsMethod, JavaMetricsField. Each of these classes extends the MT4j MTTextArea class so that actual text can be displayed or the entity can be represented as a rectangle with no text. The MTTextArea class extends the MTRectangle class. We also have some specialized classes for some of the visualizations as they required a slightly different class such as JavaMetricsWord and JavaMetricsDependency.

There are visual aspects for the entities including whether properties are being displayed for a certain entity and domain specific visualization features like the weight of an edge, what layer an entity belongs to, and what colour an entity is. When comparing different entities we created a comparator class (JavaMetricsEntityComparator) that compares two classes based on their entity name.

To launch SourceVis the SourceVisShell class is executed which inherits from the MTApplication and is the main applet from the MT4j application. Upon launching the SourceVisCategoriesScene is displayed. This is the startup screen for SourceVis (see Figure 4.2).

The SourceVisCategoriesScene contains all the loaded software systems as VisSystems. A VisSystem contains the name of the system, a list of all the versions, the current version, and associated data metrics files.

The SourceVisCategoriesScene has three kinds of visualization Categories: exploration, structure, and evolution. Each Category has many VisScenes which are images to represent a visualization and other properties. A VisScene contains one SourceVisScene (Abstract Class) which represents a visualization in the application.

The SourceVisScene inherits from the MT4j AbstractScene and is the actual scene that is drawn on the canvas in the MTApplication. The SourceVisScene can be any one of the different concrete software visualizations in the application. Some visualizations inherit from the ChartScene which integrates the JFreeChart library and supports bar, line, and area charts. The SourceVisScene has menus for displaying different system versions, and options for manipulating the data in the visualizations.

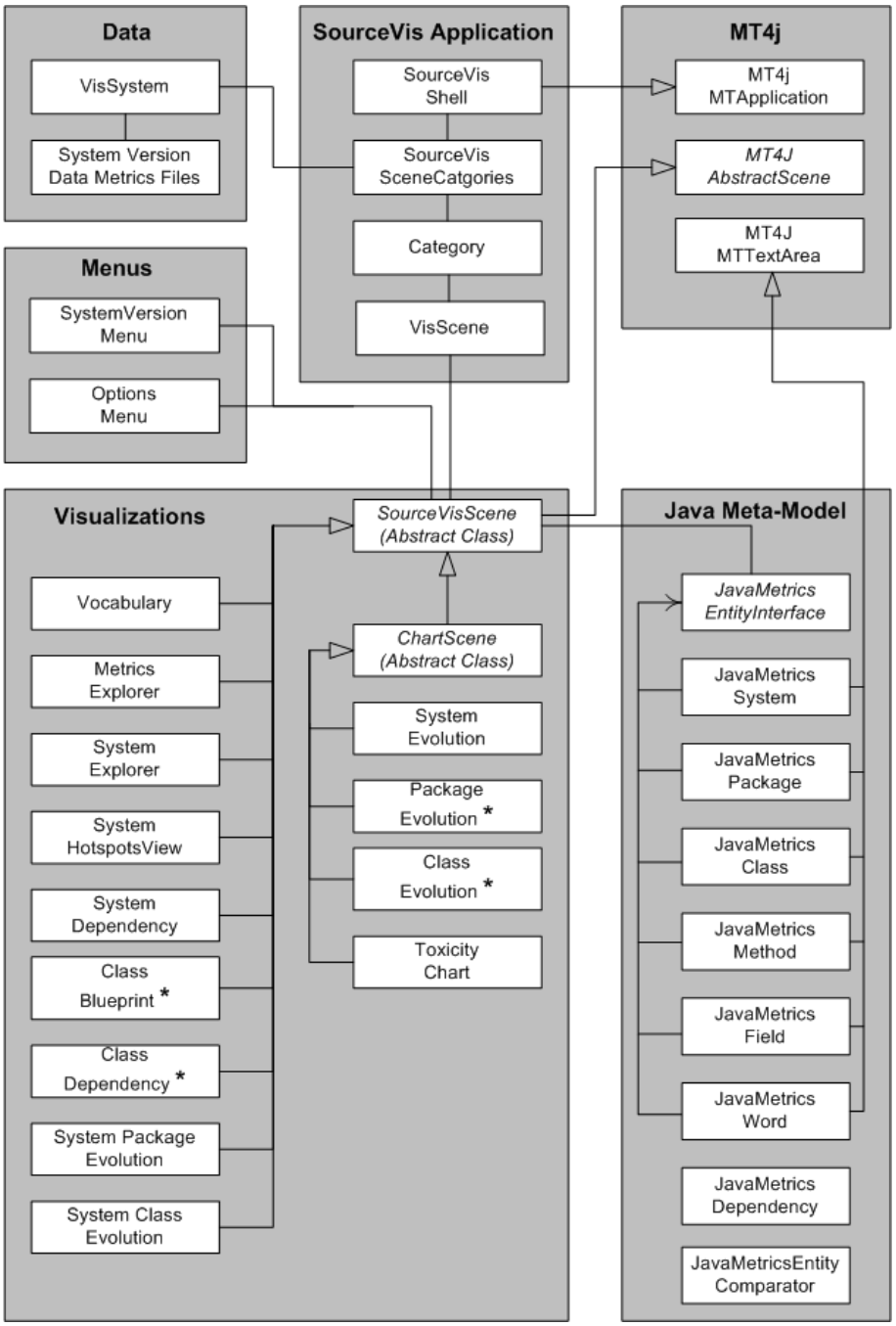


Figure 4.3: SourceVis Architecture. The SourceVis application section represents the classes to make the application start. The MT4j section represents the classes we build upon to create SourceVis. The Data section represents the metrics data files. The Menus section represents the graphical menus in SourceVis. The Java Meta-Model section represents the classes to model the entities from the Java programming language. The Visualizations section represents the visualizations in SourceVis. An * denotes individual entity visualizations.

4.1.5 Implementation

We wanted to develop SourceVis with a purpose-designed tabletop toolkit that was cross-platform, well supported, contained substantial documentation, and allowed for extensions. For this reason we developed on top of the open source MT4j toolkit and integrated other third party Java libraries for specific visualizations [188]. As opposed to developing upon a multiple device toolkit such as jQMultiTouch [222].

To create visualizations from the source code we had to either generate the data and display the visualizations at run-time, or pre-process the code to generate data files and then display the visualizations. We chose the later option as it would allow us to separate the data creation step from the display of the visualizations, make the visualization display / update faster, and allow for other developers to integrate with our application in the future.

To generate data for our visualizations we required a static analysis tool that was either a plugin to Eclipse or a stand alone tool, well documented, had libraries to customize, and was well supported. Upon exploration of tools by trial and error, and also from our previous experience we selected a commercial stand alone tool called "Understand" by Scitools[1], for which we obtained an educational licence.

Figure 4.4 illustrates the visualization pipeline of SourceVis. The source code of the systems to be visualized are first loaded into Understand. The systems were then analysed from which metrics reports can then be exported. In order to collect appropriate data for the visualizations in SourceVis we developed a number of Perl scripts that were executed within Understand to obtain custom metrics, dependency, and inheritance data. Once the scripts have been executed the data files are imported into the SourceVis application. The data files are then used within the visualizations which are displayed on the multi-touch table.

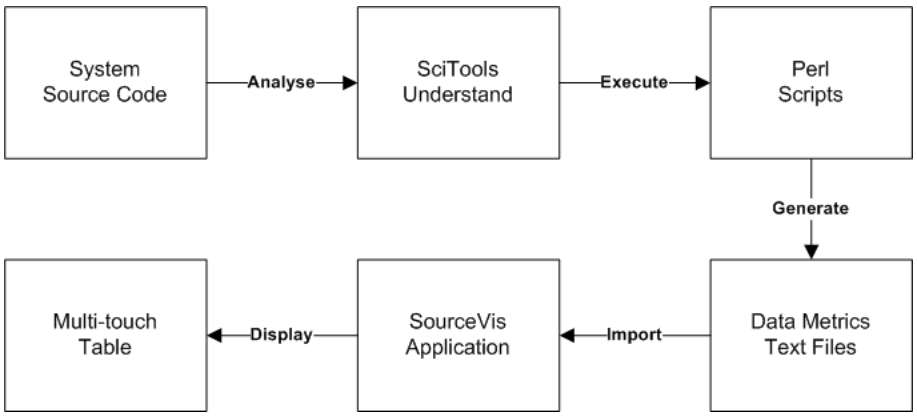


Figure 4.4: SourceVis - visualization pipeline.

[1]http://www.scitools.com/

To demonstrate SourceVis we required some readily available software systems. We are interested in understanding Java software systems so we used a sample of open source systems from the Qualitas Corpus (Version 20101126) [325]. The Qualitas Corpus is a curated collection of software systems intended to be used for empirical studies of code artifacts. We selected systems ranging in size from small to very large determined by us based on the number of lines of code and number of classes (see Table 4.1). We also selected some systems that contained more than 10 versions to demonstrate the evolution features of SourceVis.

We now present each of the separate visualizations from the different categories: Exploration (§4.2), Structure (§4.3), and Evolution (§4.4).

Table 4.1: Software systems visualized by SourceVis from the Qualitas Corpus (Version 20101126) [325]. The systems are ordered by size based on number of lines of code and number of classes. The data for lines of code, and number of classes is for the latest version of the system.

System	Lines of code	Number of Classes	Versions	Size
Azureus	453433	7249	51	V Large
Weka	224356	2099	49	V Large
ArgoUML	194859	2905	10	Large
FindBugs	109096	1744	2	Large
JHotDraw	75958	1070	6	Medium
GanttProject	47051	1058	2	Medium
JUnit	6164	209	21	Small
SquirrelSQL	6944	2211	2	Small

4.2 Exploration Visualizations

The exploration visualizations are designed for a user to be able to browse a system to get an overview of the most important aspects of a system. From an exploration visualization users can quickly navigate to an entity within a system and display an individual visualization.

4.2.1 System Explorer

The System Explorer (see Figure 4.5) is a visualization that shows all the packages and classes from a system. The left hand side shows packages and the right hand side classes in scrollable lists. Each package and class entity is selectable. The scrollable lists support inertia scrolling, also known as the rubber band effect. Initially all classes in the system are displayed. When a package is tapped only the classes from that package are displayed and the package name is highlighted green. All classes can be displayed by tapping on the classes heading label.

In the Figure the `junit.framework` package and `junit.framework.TestSuite` class have been selected. The metrics properties for the TestSuite class are being displayed along with a pie menu to display a new visualization type for the class. There are options in the bottom right for closing or minimizing the visualization.

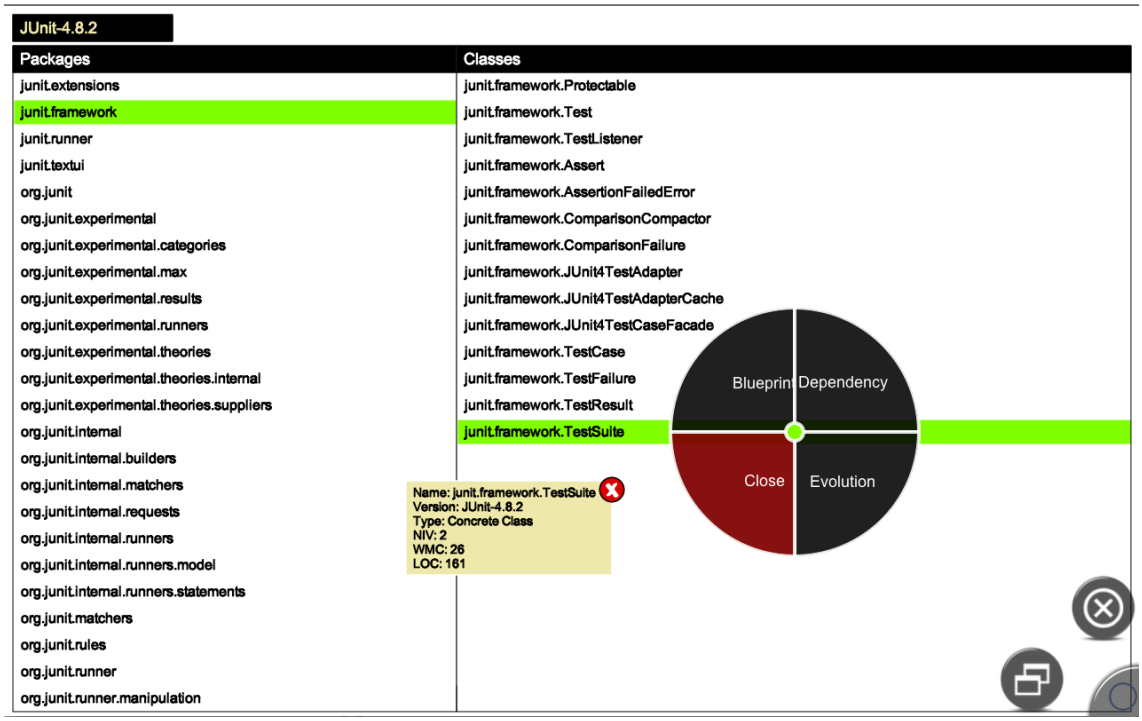


Figure 4.5: System Explorer. With options in the bottom right for closing or minimizing the visualization.

4.2.2 Metrics Explorer

The Metrics Explorer (see Figure 4.6) shows metrics about the different entities 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 [96]. All the packages in a system are initially displayed alphabetically. Tapping a package displays the metrics about the package in the metrics pane of the package details options menu and the name of the package is highlighted green. Subsequently the list of classes from a package is displayed in the classes pane. Tapping a class displays the metrics about a class in the metrics pane of the class details options menu and is highlighted green.

In the Figure a user has sorted the packages in descending order using the NIV, WMC, and LOC metrics and selected the `junit.framework` package. The package shows that there are 14 classes of which 10 classes are concrete, three interfaces, and one abstract class. The classes have also been sorted in descending order using the same metrics and the `junit.framework.TestSuite` class has been selected.

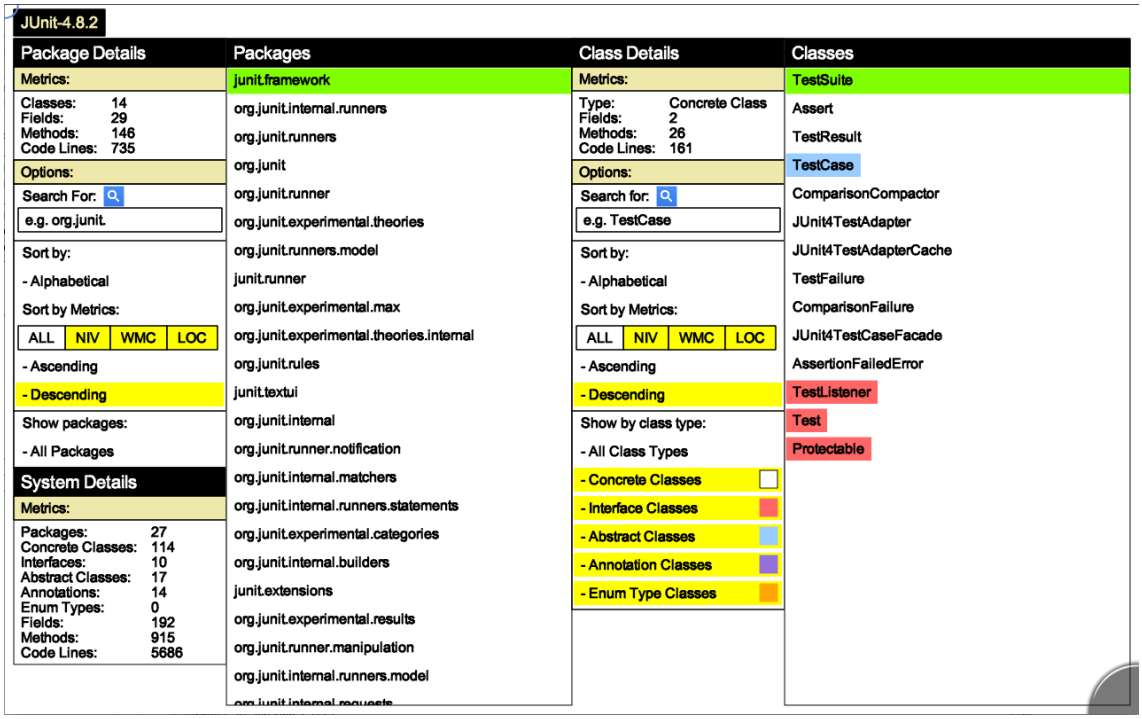


Figure 4.6: Metrics Explorer.

4.2.3 Vocabulary

The Vocabulary visualization (see Figure 4.7) uses a word cloud / wordle [341] representation to provide an overview of the vocabulary used in the names of entities (i.e. packages, classes, methods). The purpose of this visualization is to help understand the coding standards employed in the entities of a system. The visualization can also show what packages and classes are large and likely need to be refactored. The packages and classes use metrics (e.g. NIV, WMC, LOC) to determine the font size for the name of the entity. The visualization integrates the OpenCloud[2] Java library.

In the Figure the size of classes is being displayed and the `org.junit.Assert` class has been selected. A number of metrics properties for classes are also being displayed. Some classes have been filtered out leaving a total of 20 classes on display.

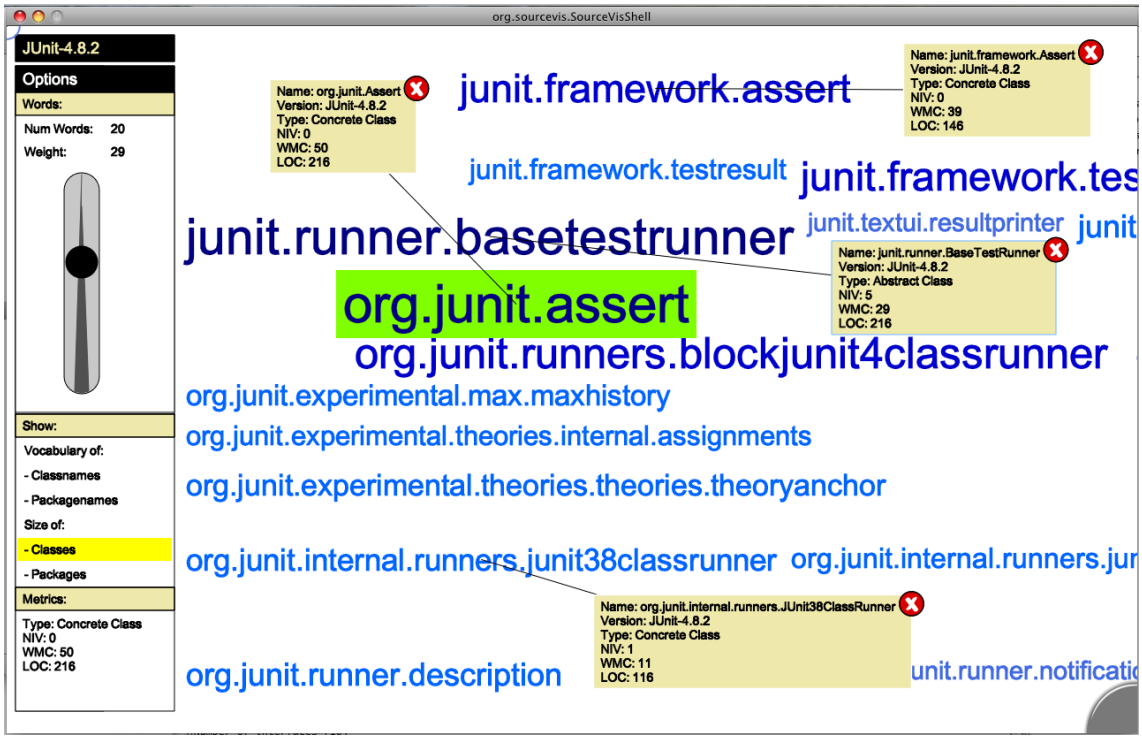


Figure 4.7: Vocabulary.

[2]http://opencloud.mcavallo.org/

4.2.4 Toxicity Chart

The Toxicity Chart adapted from Erik Doernenburg[3] (see Figure 4.8) shows which classes are toxic in a system. Each bar in the chart represents one class and the height of the bar shows the toxicity score for that class. The score is based on several metrics. The higher the score the more toxic the class is. In our adaptation we use the following metrics: file length, coupling, lack of cohesion, depth of inheritance tree, method length, cyclomatic complexity, and method nesting [96]. The individual metrics of the score are colour coded and displayed in the legend. The visualization shows at a glance not only how toxic a system is but also how the problems are distributed. Classes that score zero points are not included in the visualization. The data in the visualization can be filtered by selecting all or some of the metrics and using a slider to show classes greater than the toxicity score threshold slider. Individual elements in the chart cannot be selected. The chart can be resized using a scale gesture.

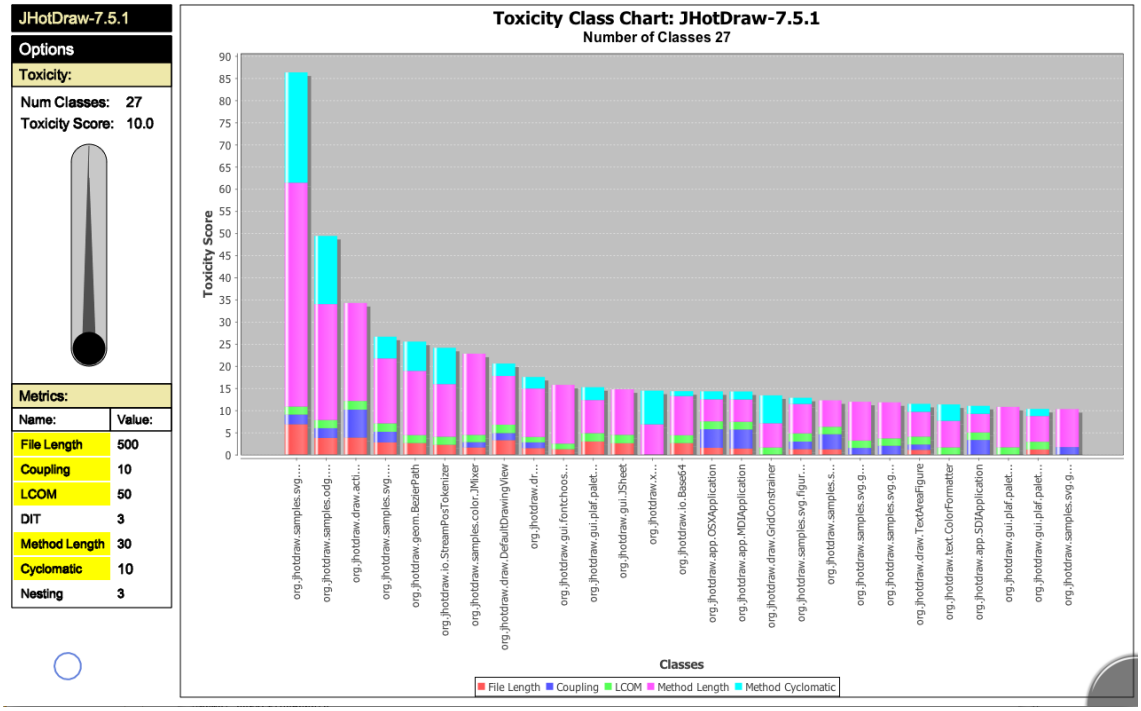


Figure 4.8: Toxicity Chart.

[3]http://erik.doernenburg.com/2008/11/how-toxic-is-your-code/

4.3 Structure Visualizations

The structure visualizations are designed for a user to explore how a software system is structured and to see how classes are dependent on each other, by examining the individual entities within a system.

4.3.1 System Hotspots View

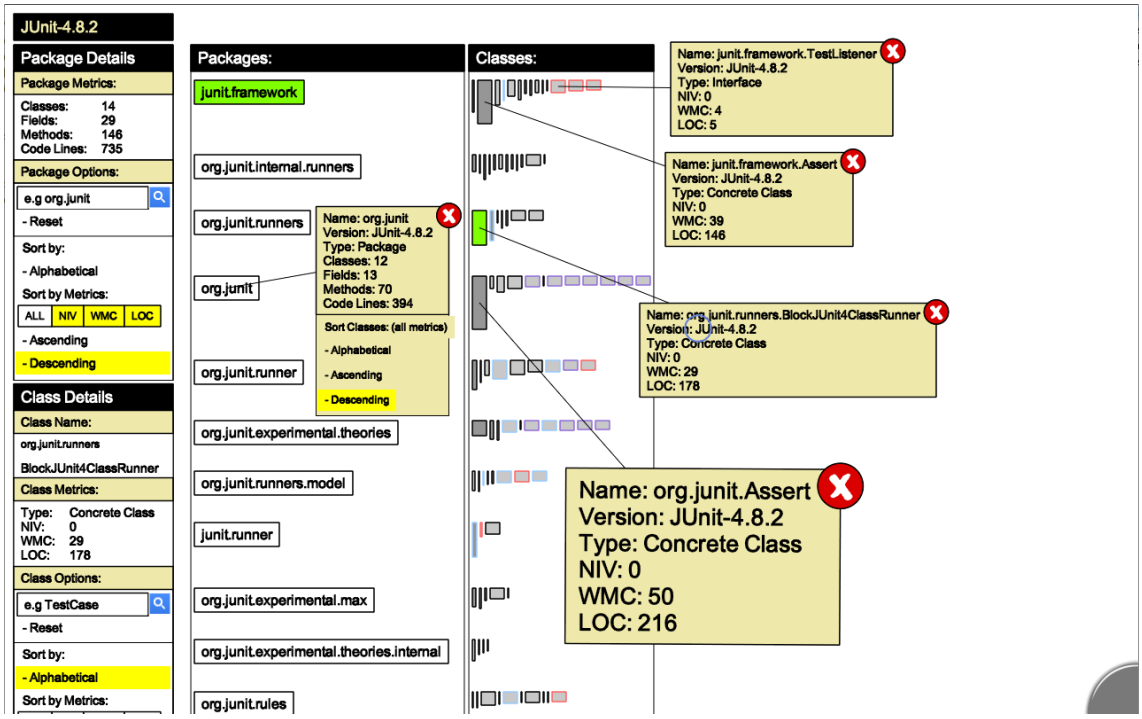


Figure 4.9: Systems Hotspots View.

The System Hotspots View (see Figure 4.9) (adapted from Lanza et al.) aims to highlight large packages and classes in a system [185]. In our adaptation packages are displayed down the Y axis in the packages pane, and classes from each package along the X axis in the classes pane. The metrics used for a package include the total metrics for classes, fields, methods and lines of code. The package properties display has options for visually sorting the classes in the package alphabetically, ascending, or descending by individual metrics or groups of metrics. Each class is represented as a rectangle using the Polymetric View encoding where the width indicates the NIV and height the WMC. The colour shading of a class represents the number of LOC. The darker the class, the more lines of code the class contains. Different border colours represent the type of class (i.e. black border = concrete class, red = interface, blue = abstract class, purple = annotation, and orange = enum type). Classes can be moved around the visualization to be compared with other classes

and can be grouped together using the lasso gesture to move more than one class at a time. In the figure the packages and classes have been sorted in descending order using all the metrics (e.g. NIV, WMC, LOC) and the `junit.framework` package and `org.junit.runners.BlockJUnit4ClassRunner` class have been selected. Other class properties are displayed too.

4.3.2 System Dependency View

The System Dependency View (see Figures 4.1(a) and 4.10) shows all the classes in the system displayed in a circle with bezier curved edges to represent what other classes they depend on. Tapping a class entity displays the class name, metric values, and highlights the dependencies between the selected class and the dependent classes. A slider option allows filtering class dependencies according to the slider weight value. Classes can also be filtered according to class type. Classes that have no dependencies and or no references can be displayed which are potentially redundant classes in a system. As this visualization is displayed on a table the entire visualization can be rotated by doing a rotation gesture in the middle of the circle. In the Figure `org.junit.internal.runners.JUnit4ClassRunner` class is selected which has eight dependencies.

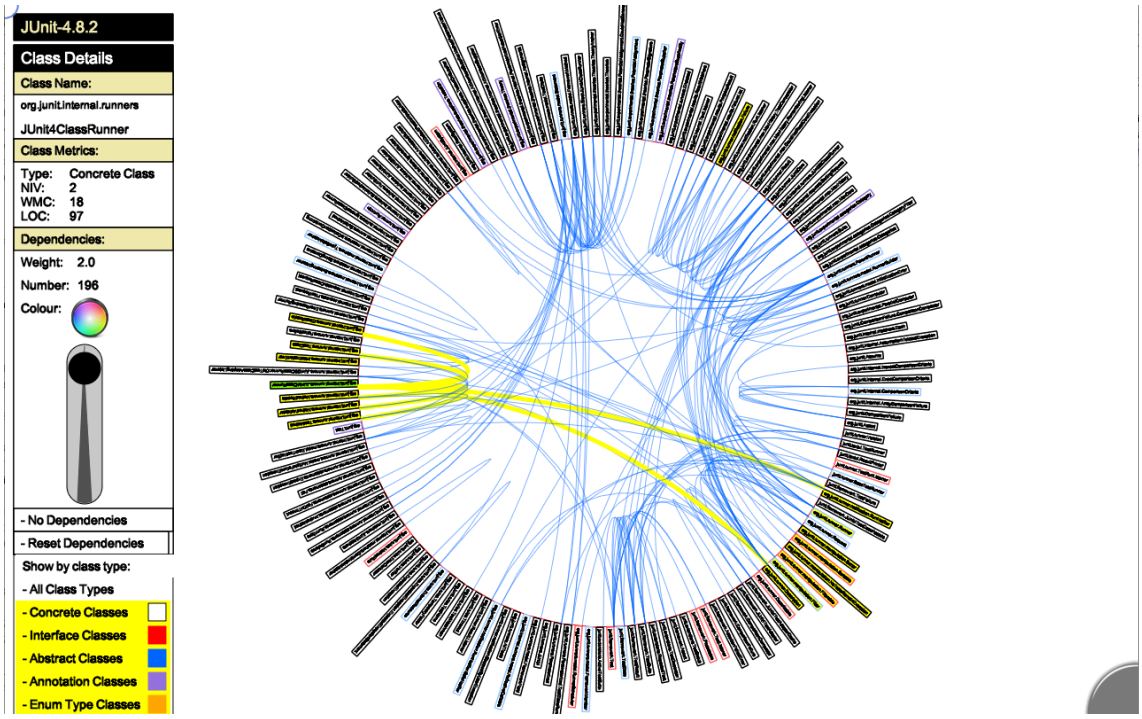


Figure 4.10: System Dependency View.

4.3.3 Class Dependency View

The Class Dependency View (see Figure 4.11) shows what classes a class depends on. The main class is coloured in yellow with blue edges to dependent classes. Dependent classes are coloured according to class type. Edges are displayed to show the dependencies between the main class in the middle of the visualization and dependent classes on the outer circle. Edges are weighted according to the number of times each class is dependent on the main class (e.g. object creation, method calls, imports). The stronger the dependency the thicker the edge weight. Weight labels are displayed on the edges and can be hidden. The main class and dependent classes can be moved around the visualization while preserving the edge dependencies. Dependent classes can be filtered by adjusting the slider weight threshold. In the Figure the class `org.jhotdraw.draw.DefaultDrawingView` has 17 dependent classes. Two of the dependent classes have a high dependency on the main class (78 and 68), are interfaces, and are displaying class properties. These interfaces are `org.jhotdraw.draw.Figure` and `org.jhotdraw.draw.handle.Handle`.

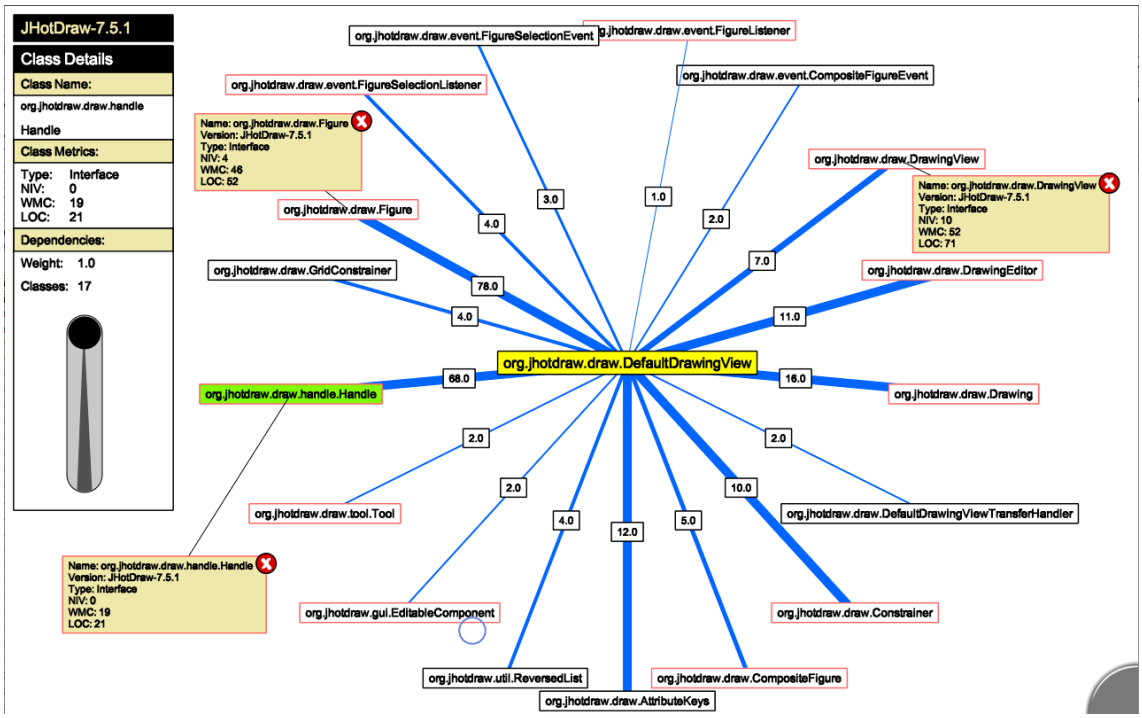


Figure 4.11: Class Dependency View.

4.3.4 Class Blueprint View

The Class Blueprint (see Figure 4.12) (adapted from Lanza et al.) shows the references between methods and attributes within a class [185]. The visualization has 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 the 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 the slider to highlight edge colouring. Methods that call themselves have circle edges at the end of the method name. In this visualization it is useful to be able to select multiple methods or attributes at once which highlights the edge references. The Figure shows the `org.jhotdraw.draw.DefaultDrawingView` class which has 80 methods and 30 fields. There is one initialization method, 42 public methods, four private methods, 21 accessor methods, 12 mutator methods. Only two of the fields are actually called. The public method `repaintHandles()` has been selected which has references with one private method and two accessor methods.

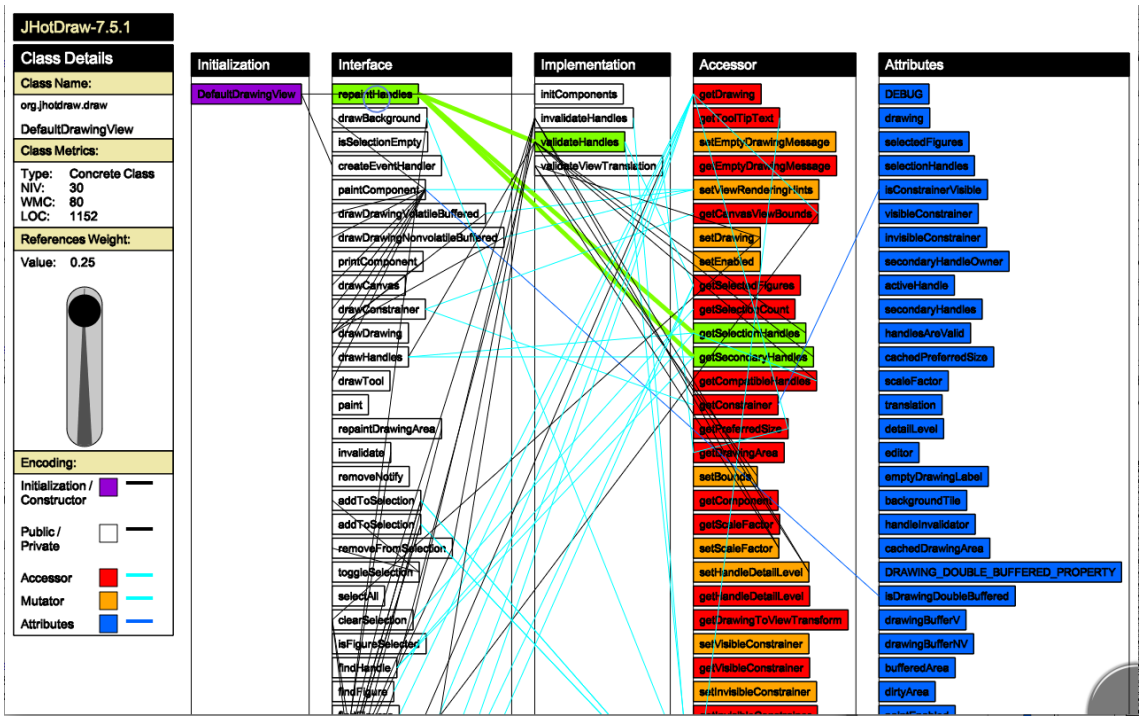


Figure 4.12: Class Blueprint View.

4.4 Evolution Visualizations

The evolution visualizations are designed for a user to see how a software system has evolved over time focusing on structural changes.

4.4.1 System Evolution View

The System Evolution View (see Figure 4.13) shows how a system has evolved over different versions using the ChartScene. The metrics used are the total number of packages, classes, methods, fields, and lines of code. The chart can be updated by selecting different metrics. Either all versions or only major versions can be displayed. Major versions are defined as versions ending in zero in the number. In the Figure JUnit 4.0 and 4.1 are the largest versions, then the next version 4.2 dramatically drops in size most likely due to major refactoring. On closer inspection we found that version 4.0 and 4.1 included the test case packages where other versions did not.

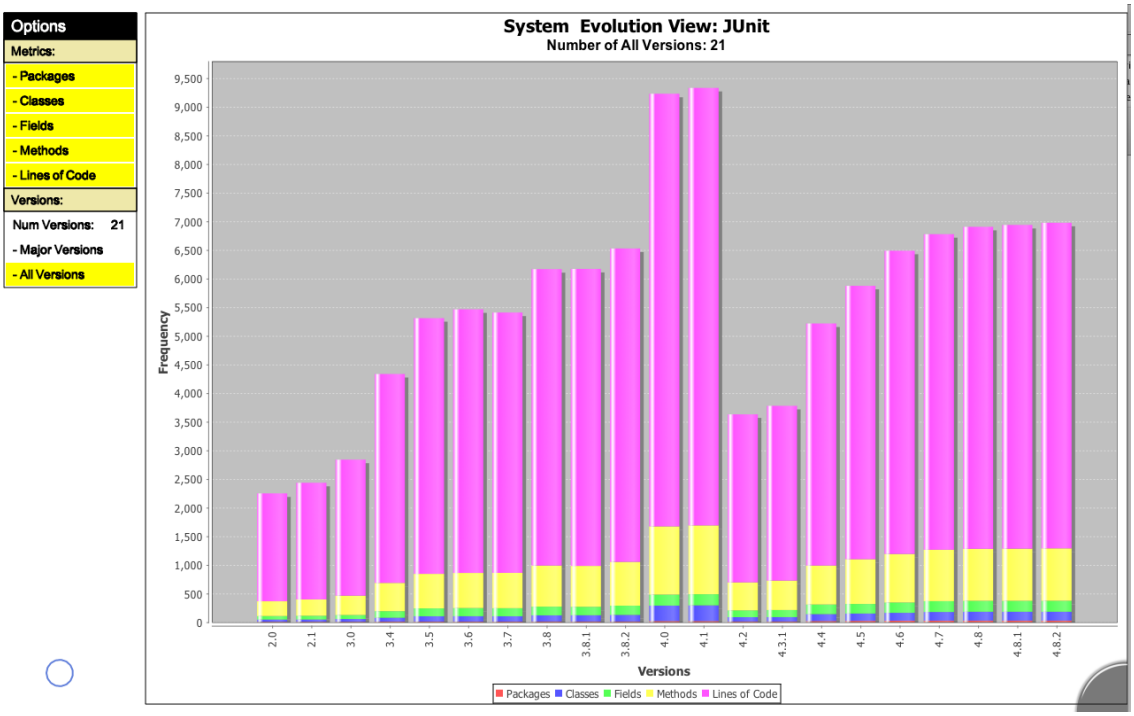


Figure 4.13: System Evolution View.

4.4.2 System Package Evolution View

The System Package Evolution View (see Figure 4.14) shows all the packages in a system as a Polymetric View encoding where width represents the total number

of methods (WMC), height: the total number of fields (NIV), and shading: the total number of lines of code (LOC). Each package is grouped inside each version of the system. Packages can be sorted across the whole visualization or within each version. Performing a tap and hold gesture on a package displays the pie menu. Selecting the versions option from the pie menu highlights all the versions a package appears in. A package is highlighted in yellow for each version it is in and linked between versions with green edges. Note, if a package has changed name SourceVis does not control for this aspect. Packages can be searched for using the keyboard which filters out the packages not matching the search criteria. All versions or just major versions of the system can be shown. In the Figure the `junit.framework` package is selected and is highlighted to show that it appears in 21 versions of the JUnit system.

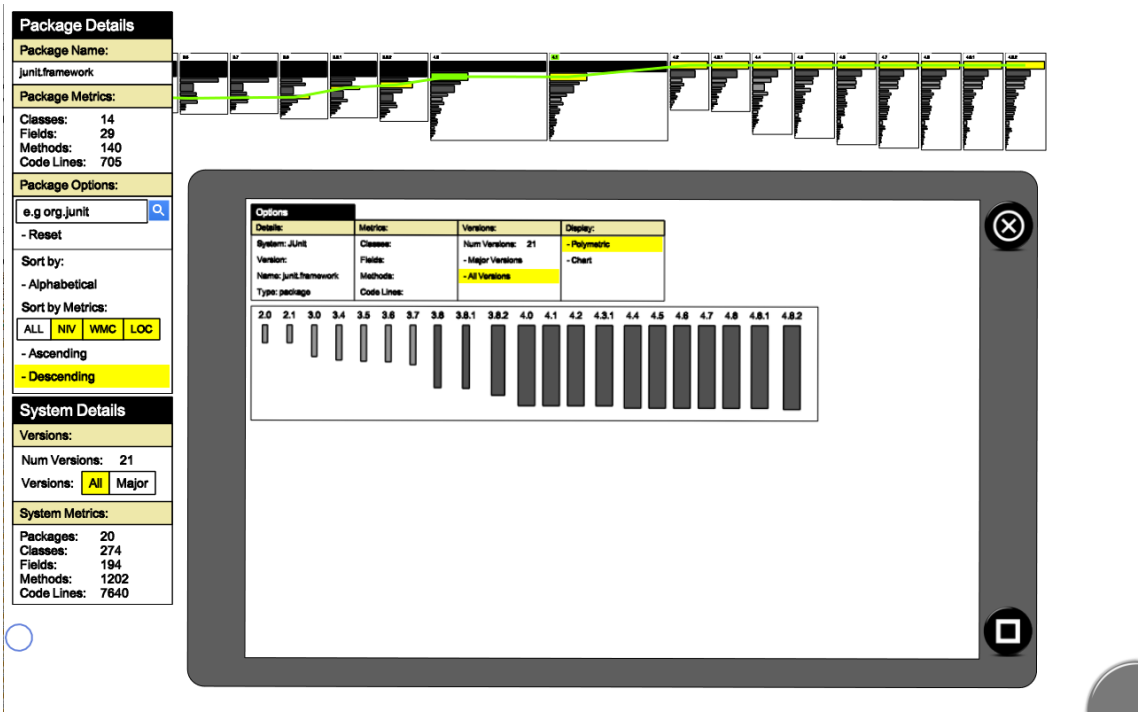


Figure 4.14: System Package Evolution View and Package Evolution View.

4.4.3 Package Evolution View

The Package Evolution View shows the evolution of one package over time. The visualization can be displayed in a separate window as in the case of Figure 4.14. The Package Evolution View can be displayed as a Polymetric View encoding or as a chart. All versions or just major versions can be displayed. This visualization can be launched from other visualizations that display packages (e.g. System Explorer, Metrics Explorer, Vocabulary, and System Hotspots View).

4.4.4 System Class Evolution View

The System Class Evolution View (see Figure 4.15) is similar to the System Package Evolution view but shows classes. The same kind of options can be applied for searching and sorting. Classes can also be filtered according to type (i.e. concrete class, abstract class, interface, annotation, enum type). In the Figure the `junit.awtui.TestRunner` class is selected and is highlighted to show that it appears appears in seven (3.4, 3.5, 3.6, 3.7, 3.8, 3.8.1, and 3.8.2) out of 21 versions. This class is the second largest class in each of the versions it appears in.

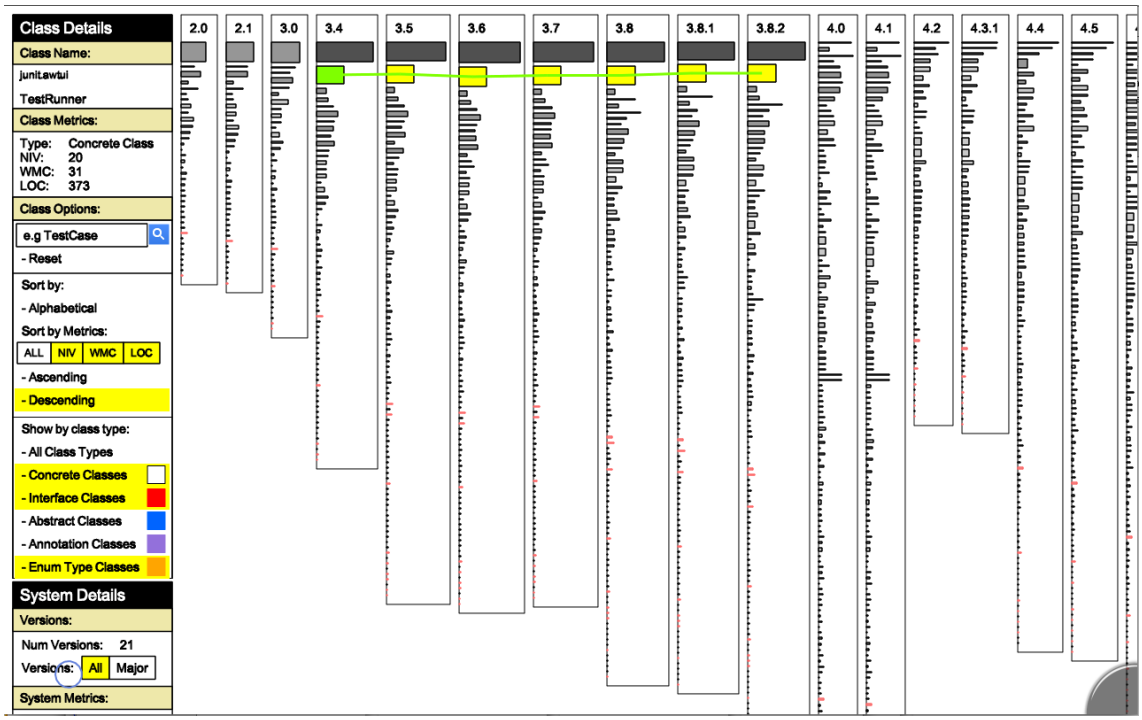


Figure 4.15: System Class Evolution View.

4.4.5 Class Evolution View

A Class Evolution View also exists similar to the individual Package Evolution View but displays classes. The Class Evolution View can be launched from other visualizations that display classes (e.g. System Explorer, Metrics Explorer, Vocabulary, System Hotspots View, System Dependency, and Class Dependency).

4.5 Discussion

Design. Developing upon MT4j required a significant amount of effort to build something substantial. Every object in in SourceVis that was displayed had to be rendered as a MT4j geometric shape, which meant all classes had to extend one of the existing built in shapes. There existed very few user interface controls and menus which we needed, hence it was time consuming to build our own widgets. The visualizations relied heavily on text so we tried to make the text as easy to read as possible, and allowed text objects to be scaled, rotated, and oriented in different positions. SourceVis currently only supports mouse, finger, and hand input but we would like to support others forms including: fiducial markers, digital pens, and tablets. Integrating SourceVis with Eclipse as a plugin and incorporating a static analysis tool would provide a more comprehensive application. The design of future interactive collaborative applications is heading towards supporting multiple kinds of devices [222]. SourceVis currently only supports multi-touch tables hence repurposing SourceVis to support more devices would be useful.

Extensibility. Extending SourceVis to support other visualization techniques is possible. As we have a representation of language constructs any visualization that utilizes these Java entities can make use of these classes. The generic options menu can easily be added to new visualizations, likewise the gestures and custom pie menu. If visualizations require charts they can use the ChartScene class. We have integrated two open source Java libraries (JFreeChart and OpenCloud), therefore integrating other libraries should also be possible.

Display. The table has a large display but the resolution is rather low at 1280x800 pixels. This makes it hard to visualize small details of information and requires zooming to see the details. Ideally we would like the table to have a much higher resolution (e.g. full high definition 1920x1080 pixels) when compared with contemporary desktops as this will allow greater precision for work place tasks. Using LCD screens instead of projectors would create a much higher resolution. Anything that is put onto the table generates a blob which affects the application be it fingers, paper, or some other physical object. Putting objects such as paper on the table minimizes the available display space for viewing and interacting.

Performance. SourceVis required a large amount of data to be rendered and the hardware we used did not perform fast enough for some visualizations. This was the case for the Toxicity Chart when modifying the slider. The System Dependency visualization could not render very large software systems at all. SourceVis worked very well for small to medium sized systems on all of the visualizations. We recommend using higher hardware specifications compared with specifications described earlier (§3.3) to enable a better user experience.

4.6 Summary

In this chapter we presented SourceVis, an interactive collaborative software visualization application designed for co-located software development teams for use on large multi-touch tables. We described the design, visualizations, interaction, architecture, and implementation features of SourceVis. We have illustrated how SourceVis meets our design goals by representing different aspects of software including metrics and dependency information, presenting multiple software visualization and information visualization techniques to visualize the structure and evolution of software, displaying the visualizations of SourceVis on large interactive multi-touch tables, and supporting multi-user interaction and multi-touch input. SourceVis has been demonstrated with small to very large open source Java systems from the Qualitas Corpus (Version 20101126) [325].

The next chapters present our qualitative user studies of SourceVis with computer science students (§5), and professional software developers (§6).

Part III

User Studies

Chapter 5

Preliminary User Studies

Contents

In this chapter we present three preliminary user studies we conducted with our multi-touch table (§3.3) and different versions of SourceVis (§4). The purpose of these user studies was to develop a protocol which we could use for a study with professional software developers (§6). The aim of the studies in this chapter was to collect qualitative data about how effective our software visualization techniques are, what coupling styles participants used, and participant's preference for individual vs. group work. We conducted three user studies to validate our design decisions following an iterative cycle using a grounded evaluation process [144]. Our approach included observational studies conducted as part of the design process, in situ interviews, and video recordings. The studies were conducted during different iterations of the implementation of SourceVis. Participants in our preliminary user studies were computer science students from within the School of Engineering and Computer Science at Victoria University of Wellington.

5.1 Preliminary User Study 1 - Early Feedback

The first user study was conducted near the beginning of the project with an early prototype of SourceVis and the Blue Multi-touch Table (§3.3). The study involved early versions of some of the visualization techniques from SourceVis. The visualizations included two versions of the Vocabulary visualization (Word Cloud and Wordle), Metrics Explorer, Class Blueprint, Systems Hotspots View, and a JavaDoc Web Browser (see Figure 5.1).

5.1.1 Participants

Table 5.1 lists the demographics of participants in our first user study who were a convenience sample of graduate computer science students. The column headings are abbreviated as follows: Participant ID (PID), gender (G), age range (Age), degree acquired (Deg), Java expertise (Exp), software visualization experience (SoftVis), regular use of smart mobile phones and touch tablets (M), regular use of touch screens and touch tables (T), and number of months the participant has known their colleague (Coll).

There were eight male and two female participants in this user study. The age of participants was in the range 20-34. All participants had a bachelors degree in computer science and three had a masters degree. Of the participants; one was currently an honours student (4th year undergraduate), three masters students, and six PhD students. All had experience in programming using the Java API. The average expertise of using the Java API on a self ranking between 1 (novice) and 10 (expert) was 6.3. Four participants had used some software visualization

Table 5.1: Preliminary User Study 1 - Participant Demographics. Participant ID (PID), gender (G), age range (Age), degree acquired (Deg), Java expertise (Exp), software visualization experience (SoftVis), regular use of smart mobile phones and touch tablets (M), regular use of touch screens and touch tables (T), and number of months the participant has known their colleague (Coll).

PID	G	Age	Deg	Exp	SoftVis	M	T	Coll
1	M	20-24	BSc	7	Y	Y	Y	18
2	F	25-29	BSc	7	Y	Y	Y	18
3	M	30-34	BSc	4	Y	Y	Y	6
4	M	25-29	MSc	6	N	Y	Y	6
5	M	20-24	BSc	7	N	Y	N	0
6	M	20-24	BSc	7	N	Y	Y	0
7	M	25-29	BSc	6	N	Y	N	12
8	M	25-29	MSc	4	N	N	N	12
9	F	25-29	BSc	7	N	Y	Y	2
10	M	25-29	MSc	8	Y	Y	Y	2

tools before but not on a frequent basis. One participant could not recall the actual software visualization tool they had used before. Nine of the participants had used or owned smart phones with touch interfaces or touch tablets. Seven of the participants had used touch screens or touch tables before. The participants conducted the user study in pairs. The pairs comprised of participant 1 and 2 (i.e. PID 1 and PID 2) as one pair, 3 and 4 as the second pair and so on. All but one pair chose their fellow participant before the user study. One group of participants had known each other for 18 months while the other pairs knew each other for 12 months, 6 months, and 2 months. One pair did not know each other previously.

5.1.2 Procedure

The procedure for the user study is documented in Table 5.2. Participants were welcomed and given an information sheet, consent form, and a pre-study questionnaire to complete. The questionnaire asked participants about their demographics and background experience. Following the pre-study questionnaire, pairs were given a warm up exercise by experimenting with example applications from MT4j for 10 minutes. The study involved the participants completing user tasks which involved answering 14 questions similar to the types of questions software developers ask within industry [101, 170, 333, 334, 302]. The questions asked participants to identify, count, and find information using early versions of some of the visualization techniques such as Word Cloud, Wordle (based on the Wordle layout [341]), Metrics Explorer, Class Blueprint, Systems Hotspots View, and

Table 5.2: Preliminary User Study 1 - Procedure.

Step	**Time** (max allocation minutes)
1. Introduction	5
2. Pre-study questionnaire	5
3. Example Applications	10
4. User Tasks	30
5. Post-study questionnaire	10
Total Time:	**60**

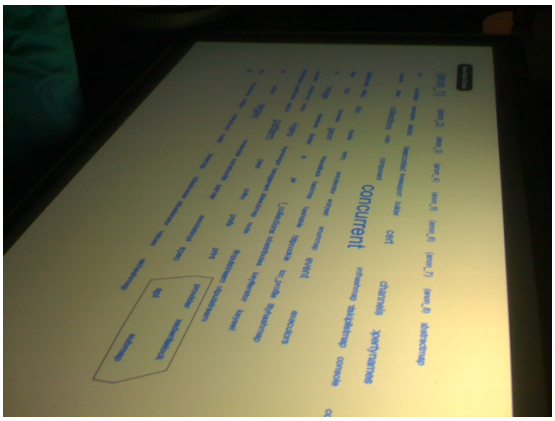
JavaDoc Web Browser (see Figure 5.1). These early visualizations did not have any options menu, system version menu, pie menus, or linking between visualizations. The rest of the techniques described earlier (§4) were not implemented at this time. The Java API version 1.6 and JHotDraw system were used in the visualization as they represented a programming language and a system with which we had experience building visualizations [10, 283]. Each pair of participants completed all the tasks using the same subset of visualizations. The participants recorded their answers to the questions on a sheet provided attached to a clip board. We asked participants to think aloud so we could understand why they were doing an action. We recorded the time it took participants to complete the user tasks including thinking aloud. Participants completed a post-study questionnaire which asked for their opinion on the effectiveness, strengths, and weaknesses of the interaction capabilities and the visualizations.

5.1.3 User Tasks

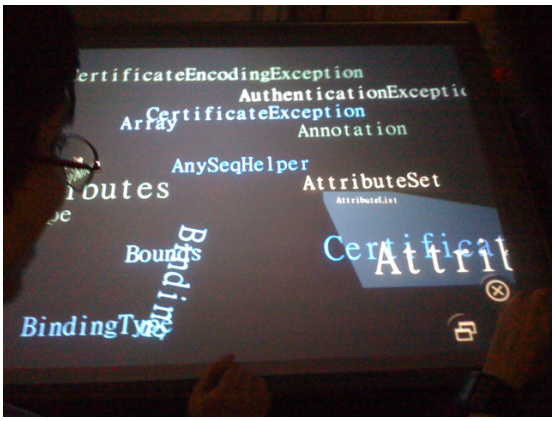
Appendix E lists the 14 questions participants were asked during the user study about the visualizations in SourceVis. Each question is listed with answers in parentheses following the question. The questions involved six visualizations Word Cloud, Wordle, Metrics Explorer, Class Blueprint, System Hotspots View, and JavaDoc Web Browser. Some of these visualizations became visualizations in SourceVis (§4).

5.1.4 Findings

We report the findings from the user tasks and the participants feedback on the effectiveness, strengths and weaknesses of the visualization techniques.

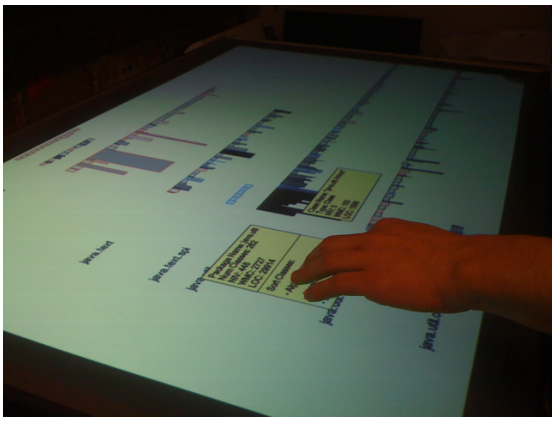


(a) Word Cloud.

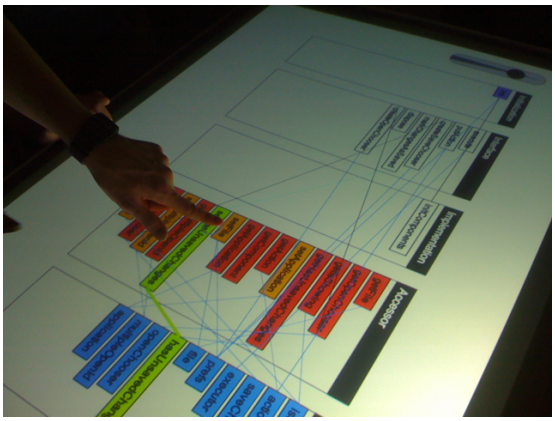


(b) Wordle.

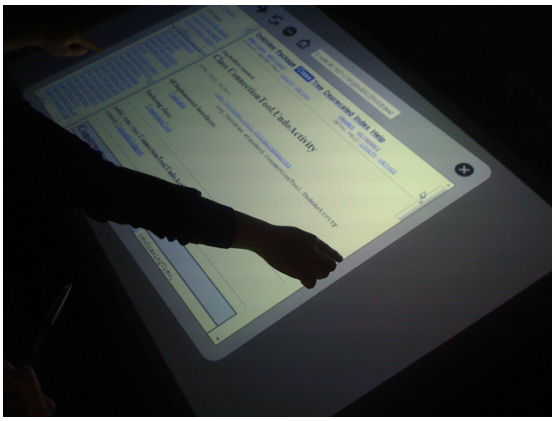
(c) Metrics Explorer.



(d) System Hotspots View.



(e) Class Blueprint.



(f) Java Doc Web Browser.

Figure 5.1: Preliminary User Study 1 - early software visualization prototypes.

Time and Errors

The first pair took 20 minutes to complete the user tasks, second pair 28 minutes, third pair 22 minutes, fourth pair 24 minutes, and fifth pair 21 minutes, for an average of 23 minutes. Pairs one, two, and five answered all the questions correctly, for a total of 36 (100%). Pair three received 34 (94%), and pair four 33 (92%). For

the Word Cloud, Wordle, Metrics Explorer, and Javadoc Web Browser all pairs answered all of these questions correctly. In the Class Blueprint questions pair three got the number of accessor methods wrong, they stated 13 rather than 1 method which is setHasUnsavedChanges. Pair four got the question using the Class Blueprint that asks which method refers to the "file" and "prefs" attribute wrong. Instead they counted the number of methods that refer to each of these attributes. In the System Hotspots View questions pair four did not get the right amount for the number of methods java.awt.Component contains, nor for java.awt.Window. Pair three incorrectly gave the number of fields java.awt.Component contains instead of number of methods.

Perceived Effectiveness of the Visualization Techniques

Figure 5.2 shows the perceived effectiveness of the individual techniques each participant stated in the post-survey. The Word Cloud ranks as the most effective technique followed closely by the Metrics Explorer, Javadoc Web Browser, and Wordle. The two domain specific software visualization techniques System Hotspot Views and Class Blueprint rank the same and a bit below these afore mentioned techniques. The Class Blueprint visualization has the largest range of values. Displaying multiple visualizations at once ranks the least effective as most participants were not aware that they displayed multiple visualizations, hence they ranked this aspect low.

Visualization Techniques

Word Cloud Visualization. All participants found the Word Cloud the most effective visualization. The word font size made it easy to understand and a white background made it clearer to read the words. Some found that the word length and letter size made it slightly confusing as there were some really long class names. Some stated that it was hard to compare words if they were not next to each other as the absolute size is not easy to determine. One participant wanted more colours in this visualization. One participant selected words using the lasso gesture and commented that separating clustered words can be difficult at times.

Wordle Visualization. All participants found the Wordle visualization easy to understand, like the Word Cloud. Adding colours to the different words helped to distinguish between them. Again like the Word Cloud the absolute size of a word is not easy to see and it is not clear what metric was being used. Words were allowed to overlap which helped when comparing the size of two words. Some complained that there was too much overlapping. A slider was added to this

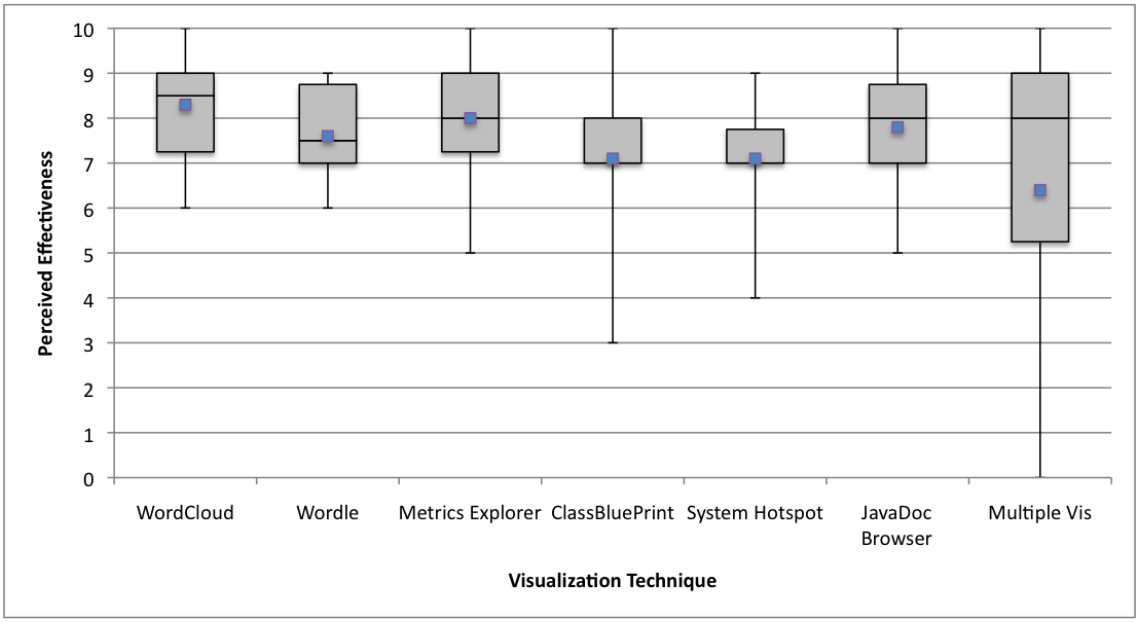


Figure 5.2: Preliminary User Study 1 - perceived effectiveness of techniques by individuals. Boxplot shows a box and whiskers plot for each visualization. The plot shows the range of values with upper and lower quartiles represented as the top and bottom line of the box. The whiskers above and below the box represent the highest and lowest values in the range. The line in the middle of the box represents the median value in the range. The blue dot represents the average value in the range.

visualization to filter out smaller words but the slider confused some participants who claimed it was not intuitive.

Metrics Explorer. This visualization provided an excellent overview and gave participants a clear summary of the metrics about a system. The white background made it clearer to read the names of packages and classes, and colour to highlight the entities made it obvious which classes were selected. One participant would have liked to have seen inheritance information about classes. A couple of participants were not sure what to expect when they tapped on the name of an entity such as where information was going to be displayed in the visualization. Some systems contained many packages and classes which required lots of scrolling to find an entity.

Class Blueprint. Participants liked how this visualization showed what methods and attributes were connected to each other. All participants commented that the highlighting of edges made answering questions more effective than not highlighting. They also liked how multiple users could highlight more than one edge at a time. The slider that adjusted the weight of the edges was a welcomed

addition, but some participants were not aware of it. Since edges crossed each other and crosses method names this made it confusing for some participants to be able to read the names of methods. One participant suggested that if different methods were selected then only display the intersection if one exists.

System Hotspots View. This visualization made it easy for participants to compare the different entities in a system as packages were laid out alphabetically, with classes grouped in packages, and the ability to move entities around the visualization. Once the participants remembered the information cues (metrics and colour encoding) it was easy to identify certain aspects of a system such as the types of classes and large classes. The properties windows made it easy to determine precise information about a package or a class. The sorting options provided a quick way to answer some of the identify and count questions. If a system was large, participants found that it was hard to get an overview of all the information because when the visualization started it did not show all the data at once. This meant lots of scrolling to find what they were looking for. Some classes in the visualization were small and required zooming in to identify their colour.

Javadoc Web Browser. The web browser provided an interface that participants were familiar with. They liked the Javadoc since they could see detailed information and the web browser was familiar to navigate. Even though the Javadoc was a familiar interface participants found it hard to find things quickly as there was no summary or overview information like in the other visualizations. Some participants did not realize that you could scroll the page with two finger panning and instead used the traditional scroll bars on the side of the browser. Half of the participants scrolled through the small left hand panes of the Javadoc to find the information they were looking for. One participant found the scrolling with two finger panning confusing because the page moves in the opposite direction to the gesture. This action mimics the behaviour of the Apple iPad web browser.

Multiple Visualizations. Allowing people to interact with several visualizations at the same time can help with productivity. Most of the participants did not actively display multiple visualization (i.e. showing more than one visualization at the same time to answer the questions). They really only displayed multiple visualizations when they were in the Metrics Explorer or System Hotspots View since these visualizations provided a way to drill down into the Class Blueprint. Some of the participants who did use the multiple visualizations commented that having different ways of looking at a system is definitely useful.

Collaborative Visualization. When asked if completing the user tasks would be easier as a single user or as a group of users, seven of the participants selected group of users. They stated that groups are better because you can each look at different parts of the system at once and discuss the solution with your colleague.

5.1.5 Discussion

We now discuss how the visualization techniques could be improved based on participant's feedback and our observations.

Word Cloud Visualization. The actual words could be resized using scale gestures which was a mistake we made in the implementation and occasionally some of the participants changed the size. It did not affect the participants' answers to questions and can be rectified in the code by making the word not have a scale gesture. Being able to select multiple words using the lasso gesture and then move the selected words around is useful. Only one participant used this gesture and found it hard to unselect words from the group. We would like to explore options for unselecting grouped items. This could be achieved by having a close button for the lasso shaped blue polygon which is used to indicate grouped items. When tapping on the close button, the items that are grouped are no longer grouped. In order to increase the colour in this visualization we could add a colour palette so that users can tap on the palette to change the colour of the words to their preference.

Wordle Visualization. Like the Word Cloud Visualization the words in the Wordle Visualization could change size using scale gestures. To help the participants who were confused with the slider control we could add labels to give more context such as "more words" or "fewer words." In terms of layout we could have a button that re-calculates the layout of the words similar to the original Wordle technique [341]. To avoid overlapping words we could apply physics so that words that are dragged around the screen make other words move out the way when there is a collision of words. Subsequently the Word Cloud and Wordle visualization were integrated into the Vocabulary Visualization (§4.2.3), words could not be scaled, a set number of words could be displayed, and there was an options menu.

Metrics Explorer. This was a new visualization technique we created from scratch. Participants liked this technique since it was simple, clear, and allowed them to drill down to details. This technique ranked second highest in terms of effectiveness. We could add method and field information about a class to this visualization and likely be displayed on the right hand side of the class information. To help with scrolling of package and class names we would add each list into a scrollable pane as opposed to doing a two finger gesture to scroll to the entity they were looking for. We would also like to add keyboard functionality so that users can search for entities in the system. Subsequently the scrolling panes, keyboard search, system version menu, and options menus (and hiding menus) were integrated into the Metrics Explorer Visualization (§4.2.2).

Class Blueprint. Edge crossing caused an issue for understanding this visualization which is why we added a slider to adjust the edge weights. Some participants were not aware of the slider. We would like to explore other graph layout algorithms to see if minimizing edge crossings has any affect on understanding this visualization. Like in the Metrics Explorer, adding keyboard search functionality may help to improve the ability to locate information especially for large classes. Being able to move layers around the screen, hide layers, and hide method and attribute labels might help with understanding too. Subsequently additional help information was added to use this visualization such as explaining the different layers, how to use the visualization, and an encoding layer (§4.3.4).

System Hotspots View. To help with navigation some form of scroll bar could be added. To maintain context within the visualization, a radar view similar to strategy game maps could be provided. The radar view will help users keep track of what part they are currently viewing and provide options for moving a lens in the radar view to go to more interesting parts of the visualization. The package and class properties features are enabled with a double tap gesture on the class or package label. All participants struggled with the double tap, especially when classes were small, partly due to the accuracy and responsiveness of the system. One solution would be to increase the threshold of time between taps. An alternative is to use a tap and hold gesture similar for displaying a Class Blueprint. Subsequently an options menu including metrics, search and sort features were added, packages and classes were put in pane windows, and metrics properties had close options in the corner of the box to make them disappear (§4.3.1).

Javadoc Web Browser. Most users were excited to be able to use a large multi-touch web browser. Since it was a familiar interface they expected it to work similarly to a desktop web browser and some participants failed to take advantage of the touch affordances of this interface, such as two finger panning to scroll up and down the page. Some participants were confused by hyperlinks, and tried double tapping when only a single tap was required. Subsequently this visualization was perceived too similar to a the existing JavaDoc web browser, hence it was not evaluated in our professional user study (§6).

Multiple Visualizations. The user study was not directly aimed at multiple visualizations. Participants were allowed to display multiple visualizations at once but most chose not to, or did not realize that they could do so until after the study. We were expecting to see issues relating to occlusion of visualizations (when one visualization appears in front of another) especially when pairs were working individually. The way we ordered the questions may have had an affect on which visualizations participants chose to display. We would like to explore how we could reorient visualizations more effectively based on the position of a

user interacting with a table, as this may help with usability.

Collaborative Visualization. At the start of the user study we stated explicitly that participants could work individually or together to answer the questions. All participants chose to work together. Before the study began during the warm up exercise most participants changed their position when interacting with the table. During the study only a couple of the pairs actually changed positions. All participants stood at the front of the table since that was the way most of the text was oriented. Pair 3 (the participants who did not know each other before the study) communicated the least while completing the tasks. Likewise, pair 4 share an office at university but do not know each other that well, also did not communicate much during the study. Both participants from pair 4 stated that they would have preferred to complete the user tasks individually.

5.2 Preliminary User Study 2 - Effectiveness and Coupling Style

Once SourceVis was more mature and we had completed all the visualizations (§4), we conducted another user study. In the second study we wanted to see if how SourceVis had met our design consideration and test the usability of the visualizations before we conducted a study with professional software developers. Besides just focusing on the effectiveness of the techniques, as in the previous study, we also wanted to know how pairs of developers would interact with the visualizations by observing their group coupling style [320].

5.2.1 Participants

Table 5.3 lists the demographics of participants in our second user study. As before these were a convenience sample of graduate computer science students. None of these participants participated in the previous study. The column headings are abbreviated as follows: Participant ID (PID), gender (G), age range, height, degree acquired (Deg), software development expertise (Exp), software visualization experience (SoftVis), smart mobile phones and touch tablets (M), touch screens and touch tables (T), number of months the participant has known their colleague (Coll), how often they program with others at the same time (Collab), and how often they conduct code reviews with other developers (Review).

There were six participants in this user study. All were male and between 18–34 years of age. Five of the participants were between 177–183cm tall except for one participant who was a lot shorter at 162cm. All participants had a degree

Table 5.3: Preliminary User Study 2 - Participant Demographics. Participant ID (PID), gender, age range, height, degree acquired (Deg), software development expertise (Exp), software visualization experience (SoftVis), smart mobile phones and touch tablets (M), touch screens and touch tables (T), number of months the participant has known their colleague (Coll), how often they program with others at the same time (Collab), and how often they conduct code reviews with other developers (Review)

PID	G	Age	Height	Deg	Exp	SoftVis	M	T	Colleague	Collab	Review
11	M	18-24	183cm	Hons	<2	N	Y	Y	4	Daily	Hourly/Daily
12	M	18-24	162cm	Hons	<2	N	Y	Y	4	Daily	Daily
13	M	25-34	177cm	Hons	3-5	N	Y	N	10	Never	Never
14	M	18-24	183cm	Diploma	3-5	N	Y	N	10	Monthly	Weekly
15	M	25-34	180cm	MSc	3-5	N	N	N	2	Never	Before Merging
16	M	35-44	180cm	PhD	3-5	N	N	N	2	Weekly	Weekly

in computer science: one a diploma, three with an honours degree, one masters degree, and one a PhD. The first four participants were currently studying towards a post-graduate degree, while the last pair were working as a research assistant and a research fellow. All had professional development experience, mostly gained through summer internships. None of the participants had used software visualization tools before, but three participants had used some tools that give outlines and overviews of software systems. Four of the participants had touch mobile phones, while one of these also had an iPad. Two of the participants had regularly used touch screens for work purposes before. The participants conducted the user study in pairs. The pairs comprised of participant 11 and 12 as one pair, 13 and 14 as the second pair, and 15 and 16 as the third pair. All the pairs chose their fellow participant before the user study. The first pair had known each other for four years, second pair 10 years, and third pair two years.

When working on previous development projects four of the participants regularly programmed with other developers daily, weekly, or monthly. All participants except one claimed that they did regular code reviews by developers either daily or weekly. The participants used a range of tools for code reviews including: source control revision tools, unit tests, project management tools, bug tracking tools, and IDEs. The other two participants who did not program regularly with others mainly worked on solo projects, hence didn't do code reviews with others or only did a review when merging with distributed revision tools.

Table 5.4: Preliminary User Study 2 - Procedure.

Step	**Time** (max allocation minutes)
1. Introduction	5
2. Pre-study questionnaire	5
3. Example Applications	10
4. SourceVis Demo and Training	10
5. User Tasks	20
6. Post-study questionnaire	10
Total Time:	**60**

5.2.2 Procedure

The procedure for the user study is documented in Table 5.4. Participants were welcomed and given an information sheet, consent form, and a pre-study questionnaire to complete. The questionnaire asked participants about their demographics and background experience. Following the pre-study questionnaire, participants were given a warm up exercise by experimenting with the example applications from MT4j for 10 minutes. Participants were then given a demonstration of SourceVis with a sample system by the session instructor. The participants were also given time to explore SourceVis using the training data. The study involved the participants completing user tasks which involved answering 16 questions similar to the types of questions software developers ask within industry [101, 170, 333, 334, 302]. The questions asked participants to identify, count, and find information using 11 of the visualization techniques. The systems used in the study were JUnit (including 21 versions) and JHotDraw (6 versions). Each pair of participants completed all the tasks using the same set of visualizations. The participants recorded their answers to the questions on a sheet provided. We recorded the time it took participants to complete the user tasks. With each participant's consent we video recorded their actions and asked them to think aloud. Participants completed a post-study questionnaire which asked for their opinion on the effectiveness, strengths, and weaknesses of the interaction capabilities and the visualizations.

5.2.3 User Tasks

Appendix F lists the 16 questions participants were asked during the user study about the visualizations in SourceVis. The questions involved 11 of the visualizations from SourceVis: System Explorer, Metrics Explorer, Vocabulary, Toxicity

Chart, System Hotspots View, Class Blueprint, Individual Class Evolution, System Dependency, Class Dependency, System Evolution, and System Class Evolution. The visualizations that were not used were System Package Evolution and Individual Package Evolution. Since the two package visualizations were similar enough in design to the System Class Evolution and Individual Class Evolution visualizations, doing more questions on these extra Package visualizations would not have been useful.

5.2.4 Qualitative Findings

We report the findings from the participants' feedback on the strengths and weaknesses of SourceVis, the visualization techniques, and the multi-touch table. We also report how the multi-touch table helped with team collaboration.

Strengths

SourceVis allowed multiple users to interact at the same time which encouraged users to collaborate, learn from each other, and work as a team.

> "Working with someone cooperatively helped me to better understand how to manipulate the information (which settings to toggle for instance). I was constantly communicating with my partner and we were always assisting each other. We were able to easily take turns manipulating the interface." Participant ID (PID) 12.

> "Large screen, people have to work together." PID 16.

SourceVis supported multiple visualizations and displaying multiple visualizations at once. Users can launch new visualizations from the start screen or using a pie menu from a currently displayed visualization to show more detailed information about an entity (e.g. Class Blueprint, Class Evolution).

> "It is easy to follow a particular class around the different visualizations, and are often linked directly via the tap and hold pie menu." PID 11.

SourceVis was designed so that many elements in the visualizations could be manipulated, in a visually consistent manner. The manipulation features were sorting, filtering, searching, and moving elements.

> "I liked how I could manipulate everything. I appreciated being able to zoom and rotate individual items away from my partner so I could get a better look." PID 12.

The visualizations provided an overview of a system and the ability to drill down to get more details on demand about specific entities including packages and classes.

> "There were lots of information on one screen which allowed me to quickly understand the data on a high-level before manipulating the components to get individual details." PID 12.

> "Having an overview of information fairly readily available and in easily-to-understand visual forms is really helpful and quite useful." PID 13.

The visualizations allowed users to easily discover information about software metrics and trends about the software than using existing software development tools.

> "The visualizations did a great job of showing me metrics, trends, and dependencies more easily than I have experienced with existing IDEs and version control tools." PID 12.

> "The visualization made several metrics easily discoverable that are hard to get at otherwise, especially concerning evolution and dependencies." PID15.

The visualizations were designed to help identify entities that are outliers, such as large and small classes.

> "Being able to quickly see problem classes is a great feature." PID 13.

The visualizations were designed to help show relationships between entities.

> "The visualizations helped when trying to look at the relationship between different methods, classes, and packages. They also helped to give a relative view of each object and to previous versions of an object." PID 14.

Participants liked the large size of the table as it easily allowed them to see and share lots of information about the visualizations at the same time.

> "The size makes it very easy for people to be looking at different things at the same time." PID 11.

Weaknesses

Given the participants were novice users of SourceVis, some were confused as to what visualization they were currently looking at, and got lost through the linking of some of the visualizations especially if they opened many visualizations. Adding breadcrumb navigation and labels would help users to remember the context of the visualizations. Some participants got lost in terms of navigating within a visualization.

> "Some visualizations were too big to use well, such as the System Dependency view and the lists for the hotspot view." PID 11.
>
> "I found that there were too many visualizations which made it difficult to remember which one does what. I sometimes forgot which visualization I was looking at or where I came from. Some visualizations look similar and I like to enlarge the screens which masked the screens behind the current one. Add breadcrumbs to help with navigating to an earlier window." PID 12.
>
> "Create an overview of which charts are currently displayed." PID 16.

All of the visualizations supported two finger / hand zooming and panning gestures on the background canvas for navigation. Occasionally when multiple users tried to zoom at the same time the application was confused. Navigating in the visualizations using zooming and panning gestures was really only effective when one user was in control. Some entities could be moved within a visualization and participants would have liked the ability to restore the initial location of the entity. Some of the visualizations could be launched in a few different ways and some participants did not like the way to navigate to these visualizations.

> "The various visualization components could also use a preset location which they can be returned to if they are lost (Through resizing or being moved off screen)." PID 11.
>
> "Some screens seem to load differently depending on how you accessed them. In one instance, when I attempted to access information via what felt like a natural route, the screen didn't load. I had to backtrack and do it a different way." PID 13.
>
> "With multiple users there were issues with two people zooming by accident." PID 14.

All of the visualizations were designed in a consistent manner so that users can move elements (e.g. drag) and perform navigation interaction gestures (e.g. two

finger zoom in and out, and pan for navigation). Many of the participants were confused as to what could and could not be interacted with in a visualization. Occasionally some elements were movable, and touching text elements had different effects on the visualization which was not what participants expected. Touching some text elements also behaved differently on some of the visualizations. Sometimes when participants were touching the same element in a visualization simultaneously occasionally the element would suddenly move or change size unexpectedly. Creating a consistent look and feel, and giving participants more time with SourceVis, would allow them to become more familiar with the user interface style.

> "Not always clear what is usable (such as the filter headers are collapsible). Icons on the header which can be interacted with would aid the users." PID 11.

> "Sometimes we both touched the same object at the same time, which lead to surprising changes in size and position of the object." PID 12.

> "Some interface issues got in the way of accessing information - clicking on Evolution in one screen didn't do the same thing as clicking on it in another." PID 13.

> "It was not always obvious what part of the interface could be touched and what it would do, and there were some inconsistencies as to how to close specific popups. Active elements should be made more discoverable, and a bit more consistency would be helpful (for example a tap always does this, tap-and-hold that, double tap something else)." PID 15.

> "The tools do not feature any highlighting where an interaction is possible." PID 16.

We positioned the main menus primarily on the left hand side of visualizations. This made it difficult for participants to interact with them if they were standing on the other side of the table. If a participant wanted to display a new visualization from the start screen and one visualization was being displayed at full screen it required closing or minimizing the currently displayed visualization. Creating display on demand features for the options menu and creating new visualizations (ones from the start screen) to open anywhere would give more flexibility for SourceVis.

> "It was a bit cumbersome if you wanted to for example click on something but the buttons were on the other side, then you had to ask your team mate to do it." PID 15.

Some of the questions involved finding out specific information such as counting the number of relationships an entity has with respect to other entities. Our visualizations required participants to count these relationships rather than providing summary information.

> "When we attempted to view specifics of a certain relationship such as counting the number of dependencies or checking which version contained the most classes." PID 14.

Some participants found it difficult to remember the visual encoding representation. Such as the what the dimensions of the classes are in the Polymetric Encoding, and the colour coding for classes.

> "I didn't understand some headings initially such as WMC and LOC. More description or more fuller headings would have been nicer." PID 12.

The pie menus caused some issues when displayed, especially if the canvas was zoomed in or out at the extreme ends. Either the pie menu was too small or too big and off screen. Most of the time the pie menu worked as expected and participants could read the sub menus.

> "Pie menu needs to be independent of zoom, as it becomes a little clunky and hard to read when half of it disappears off the bottom of the screen." PID 13.

The search feature was available in the Metrics Explorer, System Hotspots View, System Package Evolution, and System Class Evolution visualizations. When a search query was issued the entities that matched the query were displayed while other entities were hidden. Any subsequent search would search only entities that were currently visible and not the hidden entities. To make a subsequent search across all the original entities (i.e. currently visible and hidden entities) a user would have to first tap the reset option, which would then redraw the visualization in the original state. Another way to implement this would have been to highlight the entities found and make other entities less saturated in colour.

> "The search option should reset the list at the beginning of a new search." PID 11.

As the visualizations in SourceVis relied heavily on text, we tried to make sure that text was easy to read and when scaled also readable. Nonetheless occasionally some text was hard to read. Some participants suggested using a fish eye lens in the System Dependency Visualization in order to make reading class names easier.

> "Maybe use a fish eye view approach for the System Dependency view, diminish/hide elements which aren't related when reading the name of a class." PID 11.

> "Some text appeared too small to read so I had to enlarge the text or view manually." PID 12.

The touch table used a RearDI setup and CCV which makes it hard to get precise touch points and not detecting touch points slightly above the table. On our tables participants hovering their hands slightly above the table caused touches to be detected which had some unexpected behaviour with the visualizations such as object manipulation.

> "The touch precision wasn't the best, but it was still accurate enough to grab small objects." PID 12.

The resolution of the table was 1280x800 pixels and most of the visualizations relied heavily on text. The low resolution occasionally made some text hard to read especially when the current view was zoomed out a long way. Ideally we would like the table to have a much higher resolution, similar to contemporary desktop computers as this would allow greater precision for exploring finer detail in visualizations.

> "This particular table does suffer from low resolution, making reading small text hard." PID 11.

> "The resolution could be higher again, as one is able to read/recognize 10 pt text without problems, if rendered well." PID 16.

Some visualizations had issues with displaying lots of information, which meant users had to wait for the system to complete rendering the visualization. This was particularly the case for the Toxicity Chart when many classes were displayed and some were filtered out by the slider. The chart in the visualization had to be redrawn each time the slider was manipulated. The System Dependency visualization did not display at all for very large systems with many dependency relationships.

> "Slight performance issues apparent with the interaction stalling, not apparent if touch screen not picking up the action or if program is just thinking." PID 11.

> "Slow in response it needs time to get used to the system." PID 16.

In the previous study (§5.1) we allowed participants to use a clipboard to write down their answers. In this study we decided to remove the clipboard as

it interfered with participants' touch interactions. Instead we asked participants to enter the answers to the questions in an online form on a separate computer. We had large A3 sheets of the questions attached on the wall behind the table so that participants did not have to walk to the computer to refer to the questions while interacting. We also provided the questions on A4 sheets. We encouraged the participants to hold the pieces of paper or put them down on the table next to the multi-touch table. Quite often we observed that participants put the paper on the multi-touch table. This reduced the available display space for viewing and interacting. CCV detected paper on the table as a touch point which had an affect on the visualizations. We experimented having a separate window on the screen which contained the questions, but we felt that this would affect participants switching between visualizations and that the questions window would take up too much screen real estate.

> "It would be nice to have the questions on the table as well instead of pieces of paper." PID 16.

Team Collaboration

We asked participants how the multi-touch table helped with team collaboration. There were many aspects that were positive for team collaboration. These were: team work, multi-touch interaction, communication, and coordination between participants.

Participants liked the size of the table as it allowed them to work as a team to see lots of data at the same time, share visualizations, and learn from each other.

> "The table allowed for multiple users to work on the same thing at once and collaborate without having to try and explain as much." PID 14.

> "It was easy to show things to others." PID 15.

The multi-touch interaction and the large table removed the barriers from a single person being in control of the user interface such as in Pair Programming where there is only one keyboard and mouse for input which is controlled by the driver [22, 363]. This allowed for all team members to contribute to the task at hand.

> "Having a large screen that removes the need for a singular keyboard and mouse allows all team members to contribute to the task at hand. It felt quite good." PID 13.

The table encouraged participants to communicate with each other as they were working as a group.

> "I was constantly communicating with my partner and we were always assisting each other." PID 12.

Because of the inaccuracy at times of the detection of touch points some participants were cautious when they touched the table while their colleague was also touching the table. This led one pair of participants to coordinate taking turns when interacting with the table. When others needed to coordinate manipulating the interface for example some participants asked their colleague to select a control on the side of the table they were at as they could not reach the controls. Other researchers have considered reach when making tabletop interfaces [331].

> "I would have liked to manipulate the interface at the same time as my partner, but didn't due to accuracy issues with the table. It was easy, however, to take turns manipulating the interface." PID 12.

> "Can not not reach all the buttons." PID 16.

Summary of Qualitative Findings

Table 5.5 summarises the qualitative findings for the strengths, weaknesses, and how the multi-touch table helps team collaboration. The results from our professional user study (§6) builds upon these qualitative findings (§7).

Table 5.5: Preliminary User Study 2 - Summary of Qualitative Findings.

Strengths	**Weaknesses**	**Team Collaboration**
Multi-user Interaction	Visualization Context	Team Work
Multiple Visualizations	Navigation	Multi-touch Interaction
Data Manipulation	UI Consistency	Communication
Overviews and Details on Demand	Finding Specific Information	Coordination
Software Metrics and Data Tends	Remembering Visual Encoding	
Data Outliers	Pie Menu	
Data Relationships	Search	
Table Size	Reading Text	
	Touch Precision and Accuracy	
	Screen Resolution	
	Hardware Performance	
	Tangible Objects	

5.2.5 Quantitative Findings

We report the quantitative findings from the user tasks, perceived effectiveness of the visualization techniques, frequency of the group coupling styles, and how participants could potentially use multi-touch tables like this in the work place.

Time and Errors

We expected the participants in the user study to complete the user tasks somewhere between 20-30 minutes. The first pair took 22 minutes to complete the user tasks, second pair 24 minutes, and third pair 29 minutes, for an average of 25 minutes. All the pairs answered all of the questions correctly.

Perceived Effectiveness of Visualization Techniques

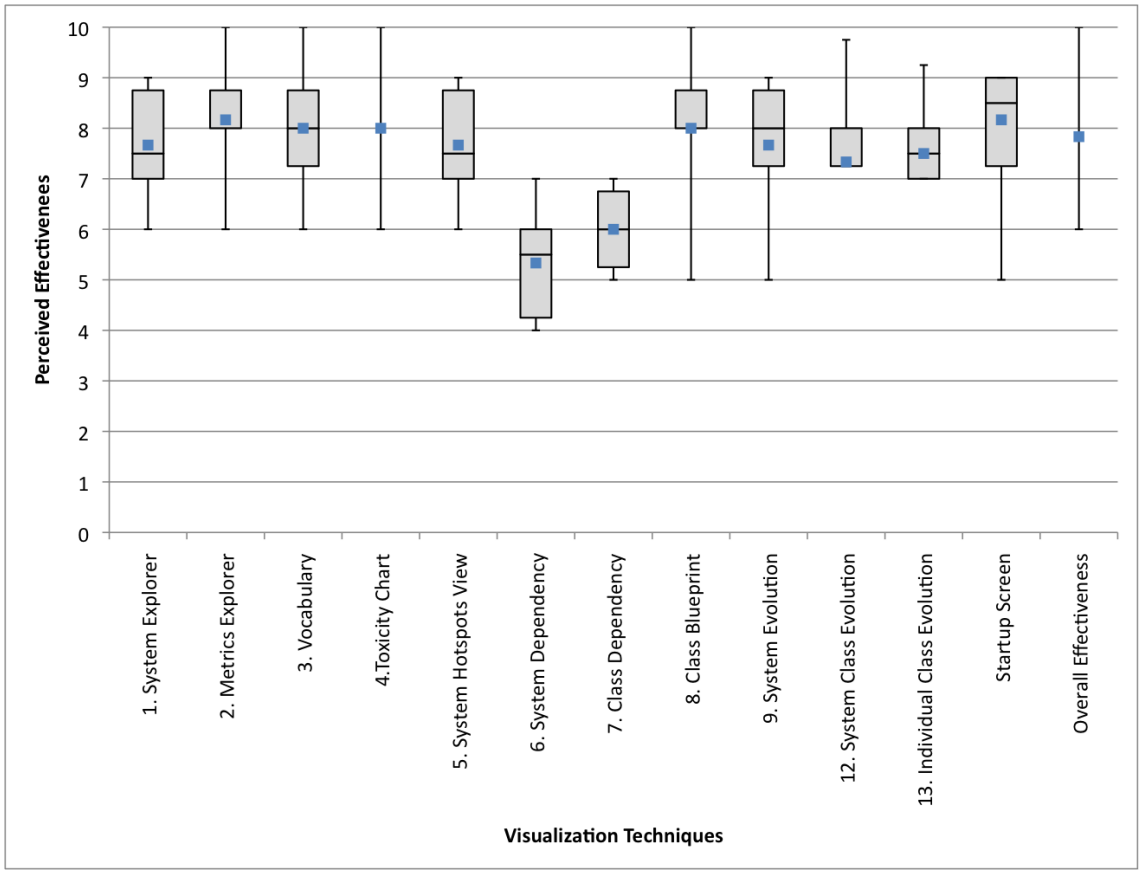


Figure 5.3: Preliminary User Study 2 - Perceived effectiveness of techniques by individuals. Does not include the System Package Evolution and Individual Package Evolution visualizations as they were not used in this study.

Figure 5.3 shows the perceived effectiveness of the individual techniques each participant stated in the post-survey, with a range of 0 being least effective and

10 being most effective. Note there is no rating for the System Package Evolution and Individual Package Evolution visualizations as participants did not use these visualizations when answering the questions. Most of the visualizations rank somewhere between 7 and just over 8 for the average. The overall perceived effectiveness of the visualizations is 7.8 for the average.

The Exploration visualizations rank the highest between 7.5 and just over 8 for the average. The chart based visualizations ranked between 7.3 and 8 for the average, with the Toxicity Chart being the most effective of these chart visualizations. The System Hotspots View and Individual Class Evolution visualizations both use Polymetric encodings and rank quite similar approximately 7.5 for the average. The least effective visualizations are the Class Dependency and System Dependency visualizations which both ranked between 5 and 6 for the average. The Class Blueprint View has the largest spread of values. We now examine each of the visualization techniques in more detail.

Visualization Techniques

System Explorer. This visualization made it easy to move between different packages and classes in a system, and to see different versions of a system. It was a good starting point when looking at a particular class to find out more information. When this visualization loaded on systems with many classes it was a bit slow as it listed all the classes. For systems that had more than 1000 classes the class list became very large and the performance of scrolling the list became slower. Most participants ranked this visualization between 7-9, except for one participant who ranked it as a 6 hence the average was 7.6.

> "Easy to move between different packages/classes. Good starting point when looking at a particular class." PID 11.

Metrics Explorer. This visualization provided lots of information about packages and classes. The visualization took up all the screen real estate when displayed at full screen. Participants liked how this visualization augmented the system explorer with extra metrics information including an options menu, and the ability to filter entities in a list. This visualization ranked as the most effective technique at 8.2.

> "Lots of information available with good filtering options." PID 11.

Vocabulary. This visualization was based on the common word cloud visualization. There were some issues for large words which participants wanted to down scale in size. The only way to scale the words is to do a zoom out gesture

which adjusts the camera. We designed it this way to prevent word changing size and losing their context against other words as experienced in our earlier user study (§5.1). Given the common word cloud representation, participants found this visualization easy to use and adding in filtering options increased the utility of the visualization. This visualization ranked quite high at 8.

> "Simple to use and filter." PID 11.

Toxicity Chart. This visualization was a chart and built upon the metrics explorer by providing a view that showed outlier classes that violated certain thresholds for different metrics. The visualization required participants to understand what the metrics meant to be able to make sense of the chart. Since the values were cumulative it was rather straight forward to get a good understanding of which classes were toxic and required refactoring. Participants liked how the chart changed due to the filtering by metrics. This visualization also ranked quite high at 8.

> "Good scaling of graph to screen making it easier to use, good filtering options." PID 11.

System Hotspots View. This visualization was the first visualization in which participants were exposed to the Polymetric encoding. This encoding made it possible to see the size of a class. This visualization was slightly more complicated for participants compared to the earlier visualizations they experienced as the Polymetric encoding was unfamiliar to them. Nonetheless, it was still ranked quite high at 7.6 almost the same as the less sophisticated System Explorer.

> "Easy to see class size visually." PID 11.

System Dependency View. This visualization displayed lots of information. For large systems the visualization did not scale as there were too many relationships that needed to be displayed. This also meant that entity labels were hard to read without zooming right in to see more detailed information. It was hard for some participants to determine how many dependencies an entity had. Some participants were confused when tapping an entity as it highlighted a relationship and then tapping turned it off. There were some inconsistencies with the information that was displayed depending on what was selected or tapped in this visualization. These were primarily programming errors which we could fix. This visualization ranked the least at 5.3.

> "The system dependency view looks nice, but I had to manually count occurrences of dependencies to get my information. Some tools to make the dependency view easier would be nice (for instance count the number of highlighted links and display that to me)." PID 11.

Class Dependency View. This visualization was less complicated than the System Dependency and instead showed a local view of the dependencies a class has instead of being displayed in a global view. We added thickness of an edge to represent the strength of a dependency and edge labels to provided extra information. Classes could also be moved around and filtered out. This visualization did not seem to rank as highly as we anticipated. Perhaps it was due to participants thinking that it should be more complicated. Navigating from one class to another required launching a new window which may have caused some confusion. This visualization was second to worst, ranked at just over 6.

> "Nice way to see more information about a class visually, same good filtering options." PID 11.

Class Blueprint. Participants like the information we added to this visualization such as an encoding legend, layer information, metrics information, and the ability to see other versions of the class. This visualization improved its ranking to 8.

System Evolution. This visualization was a chart similar to the Toxicity Chart with less complicated data. It ranked lower than the Toxicity Chart. The rendering performance of this chart was faster than the Toxicity Chart as it mainly had less data to display. This visualization ranked the same as the System Explorer, and System Hotspots View at 7.6.

System Class Evolution. This visualization also used Polymetric encodings for the classes. It was similar to the System Hotspots View but showed all classes from all versions. Participants liked how they could trace a class through the different versions, but when a class was small in size it was hard to see in the different versions. The visualization can display very large systems but takes longer to render them which decreased usability. The visualization ranked slightly lower than the System Hotspots View at 7.3.

> "Some colours in the class evolution visualization were too similar to allow me to follow a trend. I had to enable highlighting to assist me here." PID 12.

Individual Class Evolution. This visualization was simple and used two different displays either a chart or Polymetric encodings. Users could switch between the different displays. The visualization displayed an individual class over different versions. This visualization was ranked at 7.5 similar to the System Class Evolution.

Startup Screen. The startup screen ranked similar to the Metrics Explorer visualization as being the most effective technique at 8.2 on average. Participants found this screen useful as there were large icons for each visualization that were

grouped into categories and laid out in a uniform way. It was obvious for each participant to tap the icons to launch a visualization. The load menu stood out and it was clear for participants on how to load a system to be visualized. The help information when tapping and holding on an icon provided some useful documentation about the visualizations for the participants.

Group Coupling Style

We video recorded participants interacting with the table and observed what coupling style they used for the tasks. We used a subset of codes from the group collaborative coupling style for tabletop displays [137, 320]. Descriptions of the group coupling styles we used are listed in Table 5.6. We manually analysed the video recordings and coded what group coupling style the participants performed. Each video took up to four hours to examine. We only used a subset of the closely coupled styles as at all times participants were working as groups at the same time on the table.

Figure 5.4 shows the different coupling styles we observed participants performing. The frequency is measured by how many times we observed a pair using a style. We observed the pairs using very similar group coupling styles during the study. Most of the time pairs were either discussing the task and not interacting with the table (DISC), or one participant actively interacting with SourceVis while the other participant was viewing (VE). Sometimes participants would both interact with the table at the same time in the same location (SPSA). Less frequently participants would interact with the table at the same time in different areas (SPDA).

Table 5.6: Subset of Group Coupling Styles [137, 320].

Code	**Description**
DISC	Discussion: Conversation about the tool, task status, or work strategies.
VE	Viewing Engaged: One participant actively works with the table; the other is actively viewing, possibly commenting but not touching.
SPSA	Same Problem, Same Area: Both participants work on the same problem and the same area of the table.
SPDA	Same Problem Different Area: Both participants work on the same problem and different areas of the table.

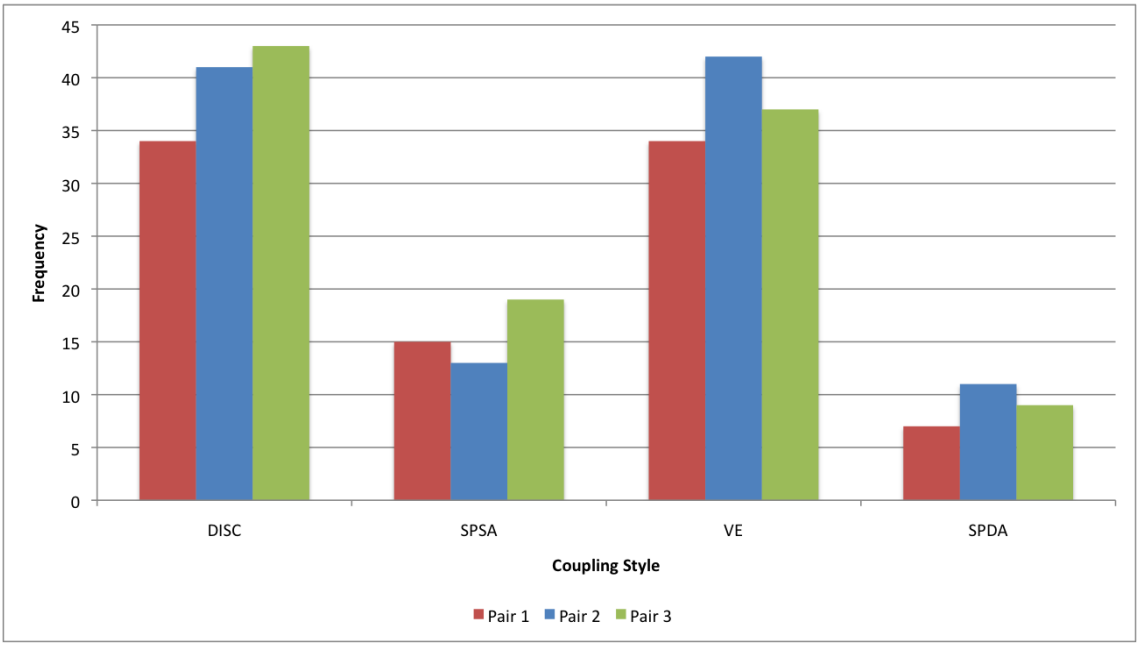


Figure 5.4: Preliminary User Study 2 - Observed Group Coupling Style.

Due to time constraints we only analysed the frequency of the coupling styles performed by the participants. We would like to analyse how much time pairs spent in the different coupling styles, and the temporal sequence of coupling styles. In this study we only looked at closely coupled styles. We look like to explore both closely and loosely coupled styles. The next chapter presents our user study with professional software developers that explores both closely and loosely coupled styles, time spent in the coupling styles, and the temporal sequence of styles (§6).

Work in Practice

Our final question in the post-survey was to ask how participants could potentially use technology like multi-touch tables in the work place. Figure 5.5 shows the frequency of activities participants thought a multi-touch table could potentially be used for collaborative software development in a co-located environment. The chart shows that participants favour the activities for designing software with architecture and modelling tasks, followed by planning team development tasks. Analysis and code reviews was also favourable. Implementing and testing software were the least favourite activities. None of the participants suggested using a multi-touch table for face to face or video conference meetings.

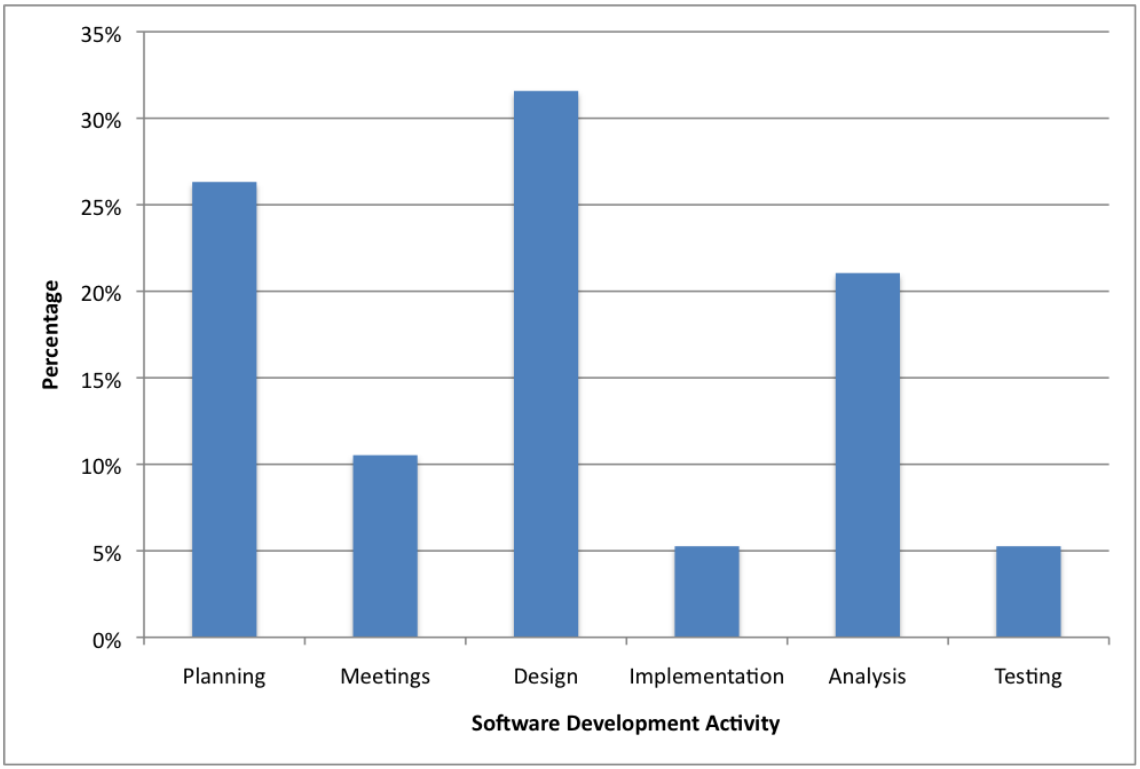


Figure 5.5: Preliminary User Study 2 - Multi-touch table work in practice.

5.3 Preliminary User Study 3 - Group vs. Individual Work

Finally, we conducted another study to see how participants would interact with SourceVis using different combinations of the user tasks as the study condition. The combinations were working as a *group* on the same task at the same time or as *individuals* working on different tasks at the same time. Being able to work as individuals on multi-touch tables at the same time is one of the claimed benefits of a multi-user multi-touch table setup [291]. We wanted to see what combinations participants favoured.

5.3.1 Participants

Table 5.7 lists the demographics of participants in our third user study. As before these were a convenience sample of graduate computer science students. None of these participants participated in the previous studies. The column headings are abbreviated as follows: Participant ID (PID), gender (G), age range, height, degree acquired (Deg), software development expertise (Exp), software visualization experience (SoftVis), smart mobile phones and touch tablets (M), touch screens

and touch tables (T), number of months the participant has known their colleague (Coll), how often they program with others at the same time (Collab), and how often they conduct code reviews with other developers (Review).

There were only two participants in this user study who formed one pair. Both participants were male and between 18–24 years old, were tall (between 183–185cm), working towards a bachelors degree in computer science, had some professional development experience gained through summer internships, and neither had used software visualizations tools before. Both participants have touch mobile phones but no touch tablets. They were friends, currently working on an undergraduate group project together, known each other for one year, don't program with others, and review their code daily.

Table 5.7: Preliminary User Study 3 - Participant Demographics. Participant ID (PID), gender (G), age range, height, degree acquired (Deg), software development expertise (Exp), software visualization experience (SoftVis), smart mobile phones and touch tablets (M), touch screens and touch tables (T), number of months the participant has known their colleague (Coll), how often they program with others at the same time (Collab), and how often they conduct code reviews with other developers (Review).

PID	G	Age	Height	Deg	Exp	SoftVis	M	T	Coll	Collab	Review
17	M	18-24	185cm	BSc	<2	N	Y	N	1	Never	Daily
18	M	18-24	183cm	BSc	<2	N	Y	N	1	Never	Daily

5.3.2 Procedure

The procedure for the user study is documented in Table 5.8. The procedure was the same as the previous study, except for the user tasks (Step 5 in the procedure). We wanted to explore participants answering the questions as groups and as individuals. Pairs got to experience working in both conditions. We allowed more time to complete these tasks as there were more questions and an individual component added to the procedure which we expected to take longer.

5.3.3 User Tasks

The tasks involved answering questions either as a group or as individuals. When answering the questions as individuals participants were instructed to use the table at the same time. Appendix H lists the questions participants were asked during the user study (these questions were subsequently used in the professional user study (§6)).

Table 5.8: Preliminary User Study 3 - Procedure. Step 5 involved answering questions as a group or as individuals.

Step	**Time (minutes)**
1. Introduction	5
2. Pre-study questionnaire	5
3. Example Applications	10
4. SourceVis Demo and Training	10
5. User Tasks (three sections as Groups or Individuals)	*25*
6. Post-study questionnaire	5
Total Time:	**60**

Table 5.9: Preliminary User Study 3 - Study condition combinations.

Section	**GIG**	**GII**	**IGG**	**IGI**
1	Group 1-5	Group 1-5	Individual 1-5	Individual 1-5
2	Individual 6-10	Individual 6-10	Group 6-10	Group 6-10
3	Group 6-10	Individual 11-15	Group 11-15	Individual 6-10

The first section of five questions participants answered either as a group or as individuals. The second section of five questions the participants did the opposite. For example if the participants did group first, they then answered the second five questions as individuals. The third section of five questions the participants got a *choice* to either do the questions as a group or as individuals. Table 5.9 outlines the possible combinations of the questions the participants answered. For example GIG means the participants answered the questions as follows: first set of five questions as a group, second set of five questions as individuals, and third set of five questions as a group.

5.3.4 Findings

We report the findings from the study condition combinations and the user tasks. Due to the small number of participants in this study and time constraints we did not ask the participants about the perceived effectiveness, strengths and weaknesses of the visualization techniques, how the table helps with collaboration, and how participants could potentially use a table like this in the work place.

Study Condition Combinations

The participants in this study started off doing the first five set of questions as a group and second set of questions as individuals. Upon completing the third set of five questions both participants selected to do group combination. This led to the combination of GIG as listed in Table 5.9. Future studies would like to explore what user task combinations pairs prefer. We would like to know if there is a difference in user performance if they start with either the group or individual condition first. We would like to know what group coupling style was performed in the different study conditions. These specific issues are addressed in more detail in our professional user study (§6).

Time and Errors

We expected the participants in the user study to complete the user tasks somewhere between 25-30 minutes based on the previous studies. The pair took 27 minutes to complete the user tasks and answered all of the questions correctly.

5.4 Limitations

There were some limitations to our preliminary user studies which we now outline. They were used to improve the process for our professional user study (§6).

Participants. We conducted our user study with a small number of participants who were a convenience sample of computer science students and not a representative sample of the target users of SourceVis. We were more interested in obtaining feedback from participants on what worked effectively, what didn't, and what other features participants were interested in.

Previous Use. None of the participants had used either the Blue Multi-touch Table or SourceVis previously. In the first study the warm up exercise was some example applications from MT4j and not SourceVis, so participants had limited exposure to SourceVis before they got to use it in the actual study. We decided not to use any previous participants across these studies as we did not want them to have any misconceptions on SourceVis or be able to have any advantage over other participants.

User Tasks. We did not vary the order of the questions (except the third study) for the different pairs of participants which may have introduced a learning bias for the visualization techniques. To reduce any learning bias we would vary the order of questions like in the third study.

Apparatus. The first user study was the first time we had conducted a study

with our Blue Multi-touch Table. Our table was custom built, so it had not been exposed to rigorous user testing or industrial performance benchmarks. For the subsequent two studies we configured the table settings to perform better.

Measurement. As this was a qualitative user study we were less concerned with how long it took participants to answer the questions, instead we wanted to have a general understanding of how long it would take to conduct the study so we could better prepare for our professional user study. Time measurements were taken using a manual stop watch. As we asked participants to think aloud to provide us feedback, we included that time when measuring how long it took them to complete the questions. Some pairs may have taken more or less time to complete the questions depending on how much time was spent thinking aloud.

Recording. In the second and third study we video recorded the participants but we only showed their hands interacting on the table. When conducting our video analysis we could not see the facial expressions of participants which made it harder to determine the group coupling style. We would have liked a multi-camera setup, with one camera recording the participants' hands and another camera recording the group interaction from a wider angle.

5.5 Summary

In this chapter we presented three qualitative preliminary user studies of participants interacting with SourceVis. We had a total number of 18 participants across the studies. The participants were a convenience sample of computer science students from our department. The first study explored the effectiveness of our visualization techniques at an early stage of development. The second study explored the effectiveness of the techniques near the end of implementation, and users' coupling style when interacting with SourceVis. Finally, the third study explored if participants preferred working as a group or as individuals with SourceVis.

In the next chapter (§6) we present a study we conducted with professional software developers using SourceVis, based on the protocols developed in this chapter.

Chapter 6

Professional User Study

Contents

In this chapter we present the design of a user study conducted to evaluate our hardware and software designs. This is based on the protocol developed from our previous user studies (§5), and uses our multi-touch table (§3) and SourceVis (§4). In this chapter we describe the design of the user study, demographics about the participants, the procedure for the study, and limitations. The following chapters present the qualitative (§7) and quantitative (§8) findings from this study.

6.1 User Study Design

Based on the protocol developed from our preliminary user studies (§5) we wanted to explore participants working on the multi-touch table completing user tasks as a group and as individuals at the same time, what condition type combination pairs selected, what coupling styles were used, what arrangement styles were used, and how effective our multi-touch visualization techniques are, and what the strengths and weaknesses were. To address these design issues we chose a within-subjects test [224].

6.1.1 Study Condition Combinations

As part of this user study we wanted to explore participants working on the multi-touch table completing the user tasks in two condition types: as a group or as individuals. We wanted to find out what condition type was most effective and what condition participants preferred. All participants got the opportunity to explore both conditions. For our study we used a within-subjects design [224] with one factor, condition type:

Group: pairs working jointly together on the same task at the same time.

Individual: pairs working separately in parallel on different tasks at the same time.

Pairs completed three sections of the user tasks (§6.3.3) in one of the two different condition types (e.g. Group or Individual). To vary the order of condition type in which pairs completed the sections, we had two groups: odd numbered pairs and even numbered pairs. Odd numbered pairs completed Section 1 in the Group condition, and Section 2 in the Individual condition. Even numbered pairs completed Section 1 in the Individual condition, and Section 2 in the Group condition. For Section 3 each pair had a *choice* of Group or Individual condition. Table 6.1 lists the study condition type combinations. The four possible combinations were Group Individual Group (GIG), Group Individual Individual (GII), Individual Group Group (IGG), and Individual Group Individual (IGI).

Table 6.1: Study Condition Type - Combinations.

Section	GIG (odd pair)	GII (odd pair)	IGG (even pair)	IGI (even pair)
1	Group	Group	Individual	Individual
2	Individual	Individual	Group	Group
3 (choice)	Group	Individual	Group	Individual

6.1.2 Collaborative Coupling Style

When participants interacted with SourceVis and the table we wanted to observe what collaborative coupling styles they utilized. The coupling styles are listed in Table 6.2 are based on Tang et. al [320, 137]. The coupling styles are divided into two categories: closely coupled (C) and loosely coupled (L). Closely coupled groups work together while loosely coupled groups work separately.

The coupling styles include discussion (DISC) amongst participants. Viewing engaged (VE) where one participant is actively working on the table and the other observing. Viewing disengaged (VD) where one participant is actively working on the table and the other is disengaged and may not even be close to the table. Both participants working on the same problem at the same time and the same area of the table (SPSA). Both participants working on the same problem but different areas of the table (SPDA). We subdivided the different problem (DP) code by adding in two new sub categories. Participants working on different problems and the same area of the table (DPSA) and participants working on different problems and different areas of the table (DPDA) (highlighted in Table 6.2).

6.1.3 Physical Arrangement Style

As part of our user study we were interested in observing where participants stood in relation to the collaborative coupling style and in the different study conditions. Only three sides of our table were available for use as one side of our table had equipment set up for video recording purposes. Figure 6.1 shows five of the physical arrangement styles similar to Tang et al. [320] that we were interested in. The five position arrangements around the table based on relative positions are: (A) Together, (B) Kitty Corner, (C) Side by Side, (D) End Side, and (E) Opposite Ends. Tang et al. [320] had two other positions as their table was accessible from all sides. These other positions were Straight Across and Angle Across where participants were on opposite sides but using both the wide sides of the table. We also came up with another position, (F) Apart, for when one or both participants were standing away from the table.

Table 6.2: Collaborative Coupling Styles [320], with our modifications highlighted.

Category	Code	Description
C	DISC	**Discussion:** Conversation about the tool, task status, or work strategies.
C	VE	**Viewing Engaged:** One participant actively works with the table; the other is actively viewing, possibly commenting but not touching.
L	VD	**Viewing Disengaged:** One participant actively works with the table; the other is passively viewing or disengaged.
C	SPSA	**Same Problem, Same Area:** Both participants work on the same problem and the same area of the table.
C	SPDA	**Same Problem Different Area:** Both participants work on the same problem and different areas of the table.
L	DPSA	**Different Problem Same Area:** Both participants work on different problems and same area of the table.
L	DPDA	**Different Problem Different Area:** Both participants work on different problems and different areas of the table.

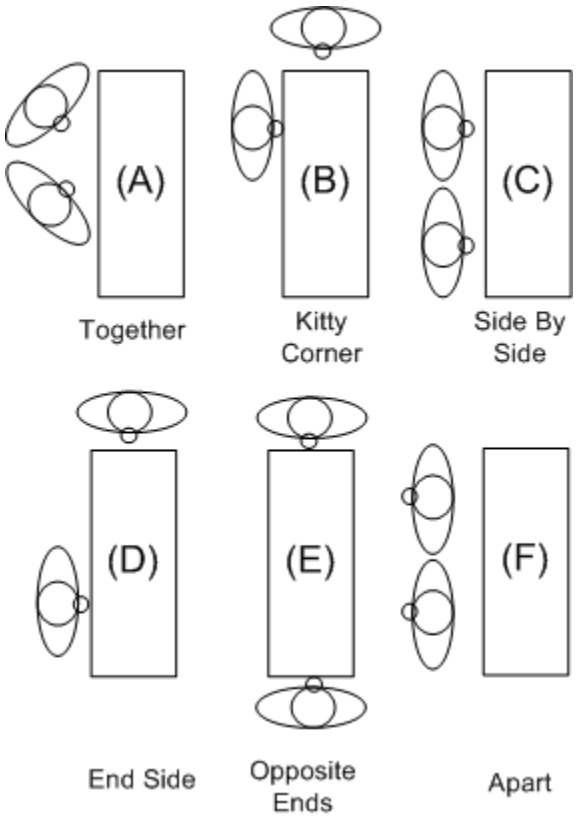


Figure 6.1: Physical Arrangement Styles around the table [320], with our modification style – Apart.

6.1.4 Participant Qualitative Feedback

As part of the study we wanted to find out what visualizations participants perceived to be the most effective. What the strengths, weaknesses and improvements are for SourceVis and the multi-touch table. We also wanted to know if participants had access to a multi-touch table for work purposes what software development activities they would use it for.

6.1.5 Research Questions

Based on the previous sections we wanted to address following research questions. The following chapters address the qualitative (§7) and quantitative (§8) findings to these research questions in more detail.

Q1 What are the strengths of the visualizations and the multi-touch table (§7.1)?

Q2 What are the weaknesses of the visualizations and the multi-touch table (§7.2)?

Q3 What improvements could be made for the visualizations and the multi-touch table (§7.3)?

Q4 Does the multi-touch table help with team collaboration, and if so how (§7.4)?

Q5 Which study condition (e.g. Group or Individual) did the pairs favour for the Section 3 of the user tasks (§8.1)?

Q6 Which coupling categories did the participants use (§8.2)?

Q7 Which coupling styles did the participants use (§8.3)?

Q8 Which physical arrangement styles did the participants use? (§8.4)?

Q9 Which visualization techniques did the participants perceive to be the most effective (§8.5)?

6.2 Participants

We now describe our recruitment process, the demographics of the participants such as characteristics, education, experience and skills, team details, programming and code review experience, touch devices, software visualization tools experience, and human ethics approval.

6.2.1 Recruitment

Conducting research about the human side of software engineering necessitates the participation of real software developers in studies, but getting high levels of participation is a challenge for software engineering researchers. Smith et al. suggest a number of ways to improve developer participation rates in surveys [305].

We exclusively recruited participants from local software development companies as they were our target users. Table 6.3 shows the list of participants in our professional user study. There were 44 participants who were grouped into 22 pairs. The participants came from 18 different companies. Two large software development companies provided six participants each which made three pairs from these companies.

We were strict on the criteria for who we recruited for the user study. Our criterion was that each pair of participants had worked for the same software development organisation and within the same team. Some pairs were currently working apart but had previously worked together for a number of years. The participants were recruited from local mailing lists, alumni of our University, personal contacts we had at local software development companies from Wellington, and emails addressed to the CEO or HR contact of an organisation. We obtained the email addresses of CEO and HR contacts through our University's Careers Service. We sent email messages to seven technical user group mailing lists, 50 companies, and 112 personal contacts. The email we sent to contacts at companies we did not know personally is located in Appendix C. A modified version of the email was sent to personal contacts. We offered participants a small gift (movie voucher) as a token of appreciation of their effort.

6.2.2 Demographics

We asked participants to complete a pre-study questionnaire where we obtained demographic information about each participant. Table 6.3 and the Figures in Appendix G chart some of the demographics data which we now elaborate upon.

Table 6.3: Participants in the Professional User Study. Pair ID, Participant ID, Gender (male, female, other), Age (years), Height (cm), Development Experience (years), Known Colleague (years), Access to Mobile Touch Phone (yes/no), Access to Touch Tablet (yes/no), Location of Development Team (co-located, distributed), Combination of Condition Type (group, individual). Grey shaded rows = GIG combination, and unshaded = IGG combination.

Pair	PID	Gender	Age	Height	Experience	Colleague	Mobile	Tablet	Location	Combination
1	1	M	25-34	183	5-10	1	Y	N	C	GIG
	2	M	25-34	175	3-5	1	Y	N	C	GIG
2	3	M	25-34	172	5-10	2	Y	N	C	IGG
	4	M	25-34	170	10-20	2	Y	N	C	IGG
3	5	M	25-34	183	3-5	8	Y	N	D	GIG
	6	M	25-34	185	5-10	8	Y	N	D	GIG
4	7	M	25-34	183	3-5	5	N	N	C	IGG
	8	M	25-34	175	5-10	5	N	N	C	IGG
5	9	M	25-34	180	3-5	1	Y	N	C	GIG
	10	F	45+	158	10-20	1	Y	Y	C	GIG
6	11	M	25-34	175	3-5	3	Y	N	C	IGG
	12	M	25-34	187	3-5	3	Y	N	C	IGG
7	13	M	45+	187	20+	1	N	N	C	GIG
	14	M	35-44	195	10-20	1	N	N	C	GIG
8	15	M	35-44	187	10-20	6	Y	Y	D	IGG
	16	M	35-44	183	5-10	6	Y	N	D	IGG
9	17	M	35-44	180	5-10	6	N	N	C	GIG
	18	M	35-44	185	< 2	6	N	N	C	GIG
10	19	M	25-34	183	5-10	1	Y	N	C	IGG
	20	M	35-44	183	20+	1	Y	Y	C	IGG
11	21	M	18-24	185	< 2	4	Y	N	C	GIG
	22	M	18-24	177	< 2	4	Y	N	C	GIG
12	23	M	35-44	172	10-20	1	Y	N	C	IGG
	24	M	35-44	185	3-5	1	Y	N	C	IGG
13	25	M	45+	170	20+	15	Y	Y	D	GIG
	26	M	45+	180	10-20	15	Y	N	D	GIG
14	27	M	25-34	187	< 2	5	Y	N	C	IGG
	28	O	25-34	183	10-20	5	N	N	C	IGG
15	29	M	25-34	177	5-10	1	Y	Y	C	GIG
	30	M	18-24	183	3-5	1	Y	N	C	GIG
16	31	F	18-24	172	< 2	2	Y	Y	C	IGG
	32	M	25-34	177	5-10	2	Y	N	C	IGG
17	33	M	35-44	172	10-20	2	Y	Y	D	GIG
	34	M	35-44	177	10-20	2	Y	Y	D	GIG
18	35	M	35-44	183	10-20	1	Y	N	C	IGG
	36	M	25-34	183	3-5	1	Y	N	C	IGG
19	37	M	45+	177	20+	8	Y	Y	C	GIG
	38	M	25-34	187	5-10	8	Y	Y	C	GIG
20	39	M	18-24	177	3-5	3	Y	Y	D	IGG
	40	M	18-25	180	3-5	3	Y	N	D	IGG
21	41	M	35-44	177	10-20	3	N	N	C	GIG
	42	M	35-44	180	10-20	3	Y	N	C	GIG
22	43	M	25-34	175	5-10	1	Y	N	C	IGG
	44	M	35-44	177	10-20	1	N	N	C	IGG

Characteristics

The "Pair" column of Table 6.3 are unique IDs for the pairs. The "PID" column of Table 6.3 are unique IDs for each participant. 41 of the participants were male, 2 were female, and 1 transgender. Figure G.1(a) shows the wide range of ages of the participants. 43% of the participants were in the range 25–34, 32% in the range 35-44 range, 14% in the range 18–24, and 11% over 45+.

Figure G.1(b) shows the height ranges. The average height was 180cm. 52% of participants are of medium height (e.g. 177–183cm, 5ft 8in–6ft), 25% are short (e.g. 150–176cm, 5ft–5ft 7in), and 23% are tall (e.g. 184cm+, 6ft 1in+). Our multi-touch table was designed to be at a height of a standard kitchen bench with his 820mm in height. As 75% of participants were medium to tall height we expected them to have no problem reaching elements on the table.

Education

Figure G.2(a) shows the highest qualification level the participants have obtained. 32% of participants have only a bachelors degree, 30% an honours degree, 29% either masters or PhD degree, and 7% a diploma or lower. In total 93% of participants had a tertiary qualification.

Figure G.2(b) shows what the subjects of these qualifications. 63% of the participants had qualifications in computer science, 7% software engineering, 21% either computer engineering, electrical engineering, maths, or physics, and 9% business with an IT focus, arts or applied computing. 91% of the participants had a science or engineering qualification.

Experience and Skills

The "Experience" column of Table 6.3 records the participants' professional software development experience. The classification were as follows: less than 2 years a graduate developer, 3-5 years a junior developer, 6-10 years an intermediate developer, 11-20 years a senior developer, and 20+ years an expert developer. Figure G.3(a) shows a wide range of software development experience due to our recruitment strategy. 30% were classified as senior, 25% junior, 23% intermediate, 14% graduate, and 9% expert. 62% of participants had 6 or more years and 86% had more than 2 years of software development experience. Therefore the participants we recruited were appropriate target users for our study.

Figure G.3(b) shows the job descriptions of participants. 48% classify their job as being a developer, 18% software engineer, 9% programmer, 7% architect, 5% consultant, web developer, and analyst programmer. The remaining 4% were

either business analysts or technical support analysts. 96% of participants were working at the code or design level of building and maintaining software systems. Therefore the participants we recruited were appropriate target users for our study.

Figure G.3(c) shows the programming languages that participants are currently developing in. Some of the participants are using multiple programming languages. 24% are currently using JavaScript, 12% Java and C/C++, 11% Python, 10% PHP, and less than 10% for other languages. Of the use of these programming languages Figure G.3(d) shows that of these programming languages, 58% are dynamically typed (non object-oriented) and 42% are statically typed (object-oriented). The example systems used in SourceVis are all Java based systems which is a statically typed language, however, the majority of the participants are currently using dynamically typed languages and may not be aware or have forgotten statically typed language features. We did not ask participants to rank their expertise on these programming languages and what languages they had used previously. Therefore it was hard to determine the expertise of participants using non object-oriented versus object-oriented programming languages.

Team Details

The "Colleague" column of Table 6.3 records how long each participant had known their fellow participant. As many of our developers had worked with their colleague on different projects, it was hard to determine exactly the number of months or years they had worked with each other. We asked them to give a best effort answer. Figure G.4(a) shows how long colleagues had known their fellow participant. 41% of participants had known each other for between 2–5 years, 36% one year or less, 18% between 6–10 years, and 5% greater than 10 years. 64% of participants had known each other for more than a year.

The "Location" column of Table 6.3 records where the pairs are currently developing software in their team, either co-located or distributed. Co-located was defined as within the same team in the same location. Distributed was defined as within the same team or project but in a different location, building, city, or country. Participants 5 and 6 had worked on open source projects together in the past. Of the distributed pairs, participants 15 and 16 were currently working on the same team but with different projects for different customers, but they had worked quite closely on the same project in the past. This was also the case for participants 25 and 26 who had worked with each other on many different projects and had known each other for 15 years. Participants 33 and 34 worked in different cities within New Zealand. Participants 39 and 40 worked in different countries, New Zealand and Australia. Figure G.4(b) shows the break down of co-located

versus distributed, of which 77% were co-located and 23% distributed.

Figure G.4(c) shows when participants have their team meetings. Participants were only allowed to select one of the possible entries. 50% of participants meet daily, 30% meet weekly, 7% meet hourly, and 2% meet when needed. We did not ask how long the meetings were, or what took place in the meetings.

Team Programming and Code Review Experience

Figure G.5(a) shows how often participants reported that they work together to program software or design an aspect of the system. 11% work together daily, 41% weekly, 16% monthly, 7% rarely, and 25% never at all. Just over 50% of participants work with another team mate to program software or design an aspect during the course of a week.

Figure G.5(b) shows how often participants reported that they conduct code reviews with other developers or perform code reviews of other developers code. Code reviews were conducted 5% hourly, 23% daily, 43% weekly, 7% monthly, 9% on demand (no time frame was mentioned), and 14% never. 66% of code is reviewed by developers during a week, while 14% never review code.

Figure G.5(c) shows how often participants reported that management review code. 75% of participant's management never review code. 5% of participant's management review code daily and 14% weekly.

Touch Devices

The "Mobile" column of Table 6.3 records participants' access to mobile smart phones that have a touch interface, whether it be for personal or work purposes. Figure G.6(a) shows what mobile smart phones participants have access to. 25% have a Samsung phone, 15% an Apple iPhone or HTC, 10% Sony 10%, less than 10% use other brands, and 19% have no mobile smart phone at all. Approximately 80% have access to a mobile smart phone.

The "Tablet" column of Table 6.3 records participants' access to touch tablets. Figure G.6(b) shows what tablets participants use, whether it be for personal or work purposes. 69% have no tablet and 31% have a tablet. 15% of participants have access to an Apple iPad and 16% have other tablet models including: Acer, Amazon, Asus, Microsoft, and Samsung. More than two thirds of participants reported they do not use touch based tablets for personal or work purposes. Hence using a large multi-touch table is a novel computing experience for the participants. We also asked participants if they use touch screen interfaces at work but did not specify for what task. 6% of participants use touch screen interfaces for work purposes.

Software Visualization Tools Experience

Figure G.7 shows the software visualizations tools participants have experienced. 30% have not used any tools before, 23% use some form of UML tools; 20% use whiteboards, pens & paper, or Visio for sketching diagrams; 11% use development tools like version control viewers for navigation; 8% use features within IDEs; 7% listed specific software visualization tools such as CodeCity [360], Gource [59], and Structure 101[1]; and 2% have used information visualization toolkits like D3 [44] and GraphViz[2]. Less than 10% use specifically designed visualization tools and toolkits.

Study Condition Combination

The "Combination" column of Table 6.3 shows what combination each participant and pair performed. We had 22 pairs and an even split of 11 pairs each for only two combinations of the condition type, GIG for odd numbered pairs (light grey shaded rows in Table 6.3) and IGG for even numbered pairs (white rows in Table 6.3). Neither GII or IGI combinations were selected by the participants.

6.2.3 Human Ethics Approval

As this user study involved human participants we required human ethics approval before we could conduct the study. We obtained Human Ethics approval from the Victoria University of Wellington Human Ethics Committee for testing software visualization prototypes with users for the duration of the PhD research. Appendix A lists our application and approval documents. The study was performed onsite in a controlled room at Victoria University of Wellington.

6.3 Procedure

The procedure for the user study is documented in Table 6.4. Participants were welcomed and introduced. We offered them a glass of water, had a brief discussion about their background to make sure they were suitable for the study based on our criterion (§6.2.1), and gave a short explanation of the study. Participants were allowed up to 120 minutes to complete the entire study. If participants reached the allocated time we asked them to move to the next activity. We now describe the pre-study activities, user study setup, user tasks, post-study activities, and data collection, coding and analysis.

[1]http://www.structure101.com/
[2]http://www.graphviz.org

Table 6.4: Procedure of the Professional User Study and maximum allocation time.

Activity	**Time** (max allocation, minutes)
1. Introduction	5
2. Pre-survey Questionnaire	5
3. Example Applications	15
4. Training Data	15
5. User Tasks	60
6. Post-survey Questionnaire	10
7. Post-interview	10
Total	**120**

6.3.1 Pre-Study

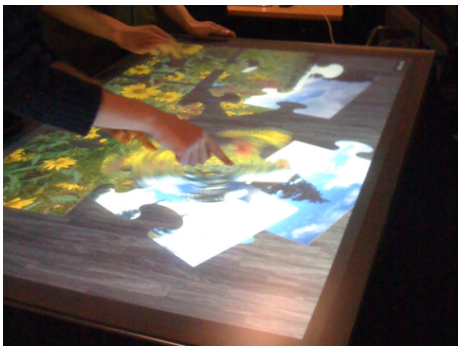
Participants in the user study were given an information sheet that provided a brief introduction of the project, how their participation will aid in our research, and what was involved in the study. Participants signed a consent form which asked their permission for us to observe and video record their interactions during the study (see Appendix B). Participants could express their right to receive access to future publications relating to the research, and were given a month within which they could withdraw their data from the study.

Before the commencement of the study each participant completed a pre-survey questionnaire (see Appendix D). The pre-survey questionnaire gathered the data reported in Section 6.2.1. We allowed up to 10 minutes for the introduction and pre-survey questionnaire.

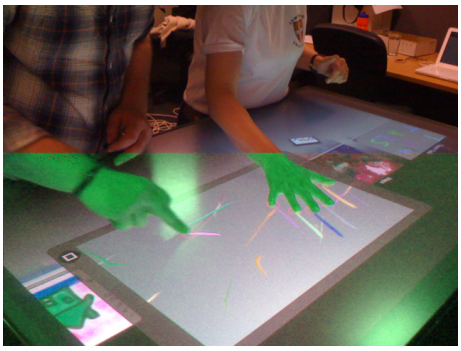
Once the pre-study was completed, the participants were given an explanation of the multi-touch table, and how the touch technology worked. To get an understanding, appreciation, and some practice on the multi-touch table participants were given some warm up exercises by exploring sample multi-touch applications for up to 15 minutes (see Figure 6.2). The participants were then given a demonstration of SourceVis by the session instructor to understand the basics of the application. To become familiar with SourceVis participants were given an additional 15 minutes to explore visualizing some example software systems (different to the ones used in the user tasks) on their own. The example systems were of different sizes ranging from very large to small including: Weka, ArgoUML, GanttProject, and SquirrelSQL (see Table 4.1).

(a) Paint.

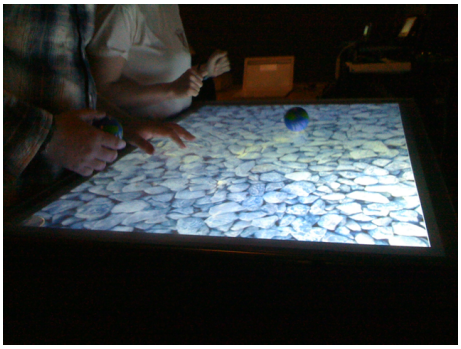
(b) Maps.



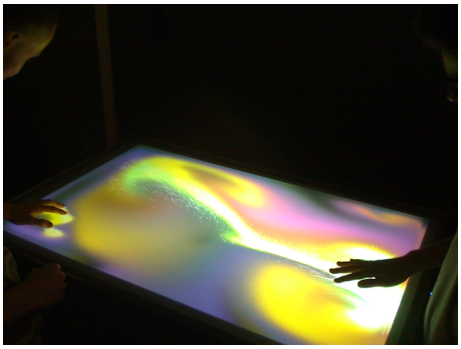
(c) Puzzle.



(d) Touch Tails.

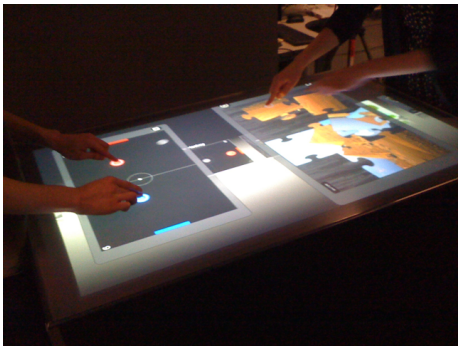


(e) Water Ripples.



(f) MSA Fluids.

(g) Air Hockey.



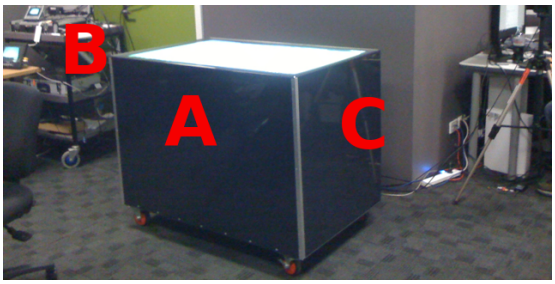
(h) Multiple Applications: Puzzle and Air Hockey.

Figure 6.2: MT4j Example Applications [188].

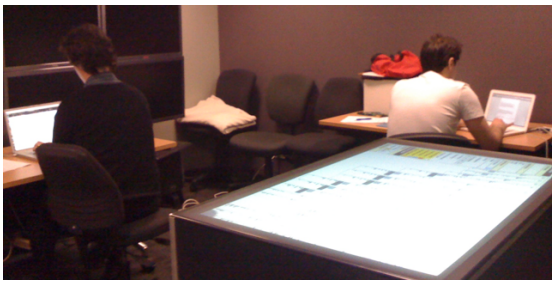
6.3.2 User Study Setup

We now describe the room of the user study as displayed in Figure 6.3. Figure 6.3(a) shows the the multi-touch table used for the study. There were three sides of the table that could be used, denoted as (A) bottom side, (B) left end side, and (C) right end side. The side closest to the wall could not be used as that encased the projector and other cables from the table. There was ample room for participants to walk around and use the other three sides of the table.

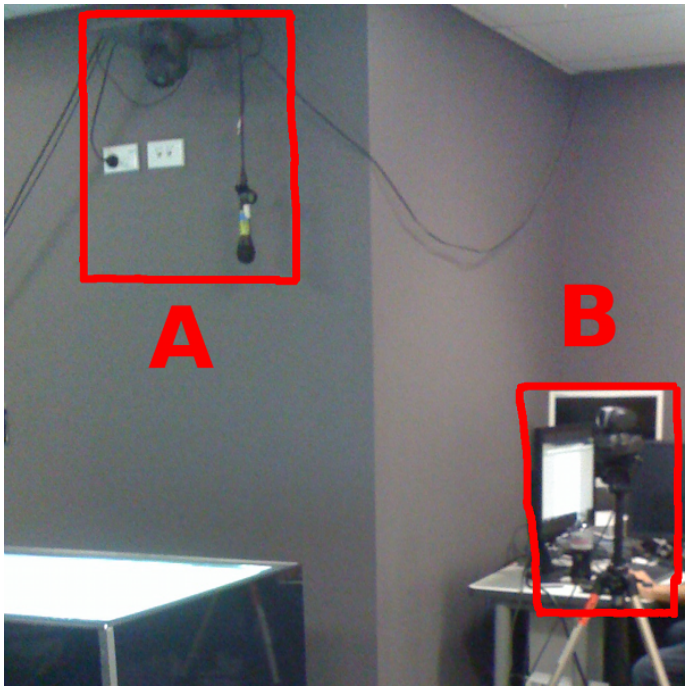
Figure 6.3(b) shows two other tables behind the multi-touch table that had two laptops for participants to complete the pre and post-survey questionnaires and for information and question sheets. This is similar to an an arrangement we would envisage for a real team where a multi-touch table is in close proximity.



(a) Multi-touch table setup: (A) bottom side, (B) left end side, and (C) right end side.



(b) Laptops setup on secondary tables.



(c) Video camera and microphone setup.

Figure 6.3: Professional user study room setup.

We asked participants to use a think aloud protocol [224] during the study, which encourages users to talk about their actions, perceptions, and expectations regarding the application's interface and functionality. Getting the users to talk about their actions and thoughts should enable us to gain insight into how each user views the computer system, identify their misconceptions and determine which parts of the interface cause the most problems for them.

We video recorded the actions of the participants from two angles during the user study. Figure 6.3(c) shows the two cameras and a microphone denoted as (A) and (B). Camera (A) was mounted above the table and pointed in a downwards angle to record the hands of the participants interacting on the screen. The microphone also hung from this camera mount. Camera (B) recorded how the participants worked around the multi-touch table and was angled horizontally face on. The output of the two video streams included a large capture of camera (A) and then a smaller picture of camera (B). We also attached large copies of the questions on paper on the wall beneath camera (A).

6.3.3 User Tasks

The user tasks in the study involved the participants answering questions in the different study conditions either as a group or as individuals in parallel using different visualizations from SourceVis. This part of the study we expected to take up to 60 minutes. Appendix H lists the Group (§H.1) and Individual (§H.2 and §H.3) questions participants answered. To simulate a real world example (but in a controlled lab study) the questions are similar to some of the types of questions software developers ask within industry such as "when, how; by whom; and why was this code changed or inserted?" [101, 170, 333, 334, 302].

Pairs answered questions in three sections where each section was five (Group condition) or ten questions (Individual condition) depending on the condition they were in. We alternated the condition each pair began in (either Group or Individual), as denoted in Table 6.3. Table 6.5 builds upon the previous table (Table 6.1) and lists the specific questions for each section.

For *Section 1* pairs either started in the Group or the Individual condition. If the pairs were in the Group condition they worked together to answer questions labelled 1–5. If the pairs were in the Individual condition then one participant answered Individual A questions labelled 1–5 and the other participant answered Individual B questions labelled 1–5 at the same time.

For *Second 2* a pair that started in the Group condition then answered the Individual A and B questions 6–10 at the same time. A pair that started in the Individual condition then answered the Group questions labelled 6–10.

For *Section 3* pairs got a *choice* as to what condition they wanted to do, either Group or Individual. If they chose the Group condition then they answered questions 6-10 if they had previously done Group (section 1) and then Individual (section 2) (i.e. GIG) or questions 11–15 if they had previously done Individual (section 1) and then Group (section 2) (i.e. IGG). If they chose Individual condition then they answered questions 11–15 if they had previously done Group (section 1) and then Individual (section 2) (i.e. GII) or questions 6–10 if they had previously done Individual (section 1) and then Group (section 2) (i.e. IGI).

Table 6.6 shows the visualizations in SourceVis required to answer each question as we wanted pairs to use a selection of visualization techniques and in different order to remove any learning bias. In Section 1 participants used visualizations from the Exploration Category (§4.2). In Sections 2 and 3 participants used visualizations from the Structure (§4.3) and Evolution (§4.4) categories. For Section 3 all participants chose the Group condition.

As denoted in Table 6.3 the only two combinations the pairs performed were GIG (odd pairs and light grey shaded rows) and IGG (even pairs and white rows). In Table 6.6 the visualizations and questions used for the GIG combination are shaded dark grey. For GIG pairs in Section 1 Group answered questions 1-5, Section 2 Individual (A and B) 6–10, and Section 3i Group 6–10. The visualizations and questions used for the IGG combination are shaded light grey. For IGG participants in Section 1 answered questions Individual (A and B) 1–5, Section 2 Group 6–10, and Section 3ii Group 11–15. The Individual (A and B) questions for 3i and 3ii are not highlighted as they were not used as neither the combinations of GII nor IGI eventuated. Note the questions in Section 2 and Section 3i are the same. For example if an odd pair had completed the Individual condition in Section 2 they then had the choice of selecting Group (6–10) from Section 3i or Individual (11–15) from Section 3ii. In summary, all pairs completed 20 questions (10 group and 10 individual – five per participant) and using different visualizations depending on what combination they were in.

Table 6.5: Study Condition - Combinations and Questions. Pairs only performed the GIG and IGG combinations.

Section	**GIG (odd pair)**	**GII (odd pair)**	**IGG (even pair)**	**IGI (even pair)**
1	Group, q 1–5	Group, q 1–5	Individual, q 1–5	Individual, q 1–5
2	Individual, q 6–10	Individual, q 6–10	Group, q 6–10	Group, q 6–10
3 (choice)	Group, q 6–10	Individual, q 11–15	Group, q 11–15	Individual, q 6–10

Table 6.6: User Tasks - Visualizations. GIG combination questions and visualizations are shaded dark grey and IGG shaded light grey. Questions and visualizations in white (e.g. questions in 3i and 3ii) were not used.

Section	Question	Group	Individual A	Individual B
1	1	System Explorer	System Explorer	Metrics Explorer
	2	Metrics Explorer	Vocabulary	Metrics Explorer
	3	Metrics Explorer	Vocabulary	Toxicity Chart
	4	Vocabulary	Metrics Explorer	Toxicity Chart
	5	Toxicity Chart	Toxicity Chart	Vocabulary
2	6	System Hotspots View	System Hotspots View	System Dependency
	7	Class Blueprint	Class Blueprint	Class Dependency
	8	Class Dependency	System Dependency	System Evolution
	9	System Evolution	System Evolution	System Package Evolution
	10	System Class Evolution	System Package Evolution	System Class Evolution
3i	6	System Hotspots View	System Hotspots View	System Dependency
	7	Class Blueprint	Class Blueprint	Class Dependency
	8	Class Dependency	System Dependency	System Evolution
	9	System Evolution	System Evolution	System Package Evolution
	10	System Class Evolution	System Package Evolution	System Class Evolution
3ii	11	System Dependency	System Dependency	System Hotspots View
	12	Class Dependency	Class Dependency	Class Blueprint
	13	Class Evolution	System Evolution	System Dependency
	14	System Package Evolution	System Package Evolution	System Evolution
	15	System Package Evolution	System Class Evolution	System Package Evolution

6.3.4 Post Study

On completion of the user study, each participant completed a separate post-survey questionnaire on one of the computers behind the multi-touch table (see Appendix D). We asked the participants' perceptions are on the effectiveness of the visualization techniques, the strengths and weaknesses of each of the techniques, what could be improved for the techniques and multi-touch table, and how the multi-touch table helped with team collaboration. We left SourceVis running so that participants could remind themselves and re-examine the visualizations in order to complete the questionnaire.

Once participants completed the questionnaire we conducted an open ended interview with some questions relating to how the participants interacted during the user tasks. It was also an opportunity for participants to provide any other feedback we had not collected during the study. We continued video recording right until the end of the interview. We allowed up to 20 minutes to complete this part of the study, which was the final activity in the study.

6.3.5 Data Collection, Coding, and Analysis

We performed no instrumentation in the SourceVis code of the participant's touch points, as that would have slowed the performance of the hardware. We also had no way to differentiate between the different users touching the surface as CCV could only tell us a touch id and location. We made notes of behaviours, coupling styles, and arrangements of participants during the study.

We collected video for all 22 pairs. Upon completion of each user study we generated a final video output which merged the two video streams, and saved them to the file system. Each video was approximately 2GB in size.

We analysed each of the 22 videos and coded the actions we observed the participants performing during the user study. The codes we used were from the study condition types (§6.1.1), collaborative coupling styles (§6.1.2), and physical arrangement styles (§6.1.3).

We examined three software tools to perform the analysis: NVivo[3], ANVIL[4], and ELAN[5]. We chose ELAN as it worked on our MacBook Pro and had an open source licence. Figure 6.4 shows the ELAN tool in action. The window shows the video output from camera (A), with the large image being the video angled down to capture the hands of the participants interacting. The smaller inset image shows camera (B) which was used to capture the pairs' coupling and arrangement styles. The right hand part shows the individual codes represented as a tier, where one tier matches one set of codes (e.g. coupling style). The separate codes are the questions each participant or pair was completing, collaborative coupling style, and physical arrangement. The bottom part shows the different codes represented as swimming lanes, one lane for each tier. Each video took up to eight hours to code as it required very precise analysis for each action. Each video had up to 250 codes totaling over 5000 codes across all 22 videos.

The data from survey questionnaires was collected into spreadsheets. We generated charts and analysis from these spreadsheets for the qualitative findings. The output files of codes from the ELAN tool were inputted into Excel to generate charts and R[252] to perform tests for the quantitative findings.

The next chapters present our qualitative findings (§7) and quantitative findings (§8) from this user study.

[3]http://www.qsrinternational.com/products_nvivo.aspx
[4]http://www.anvil-software.org/
[5]http://tla.mpi.nl/tools/tla-tools/elan/

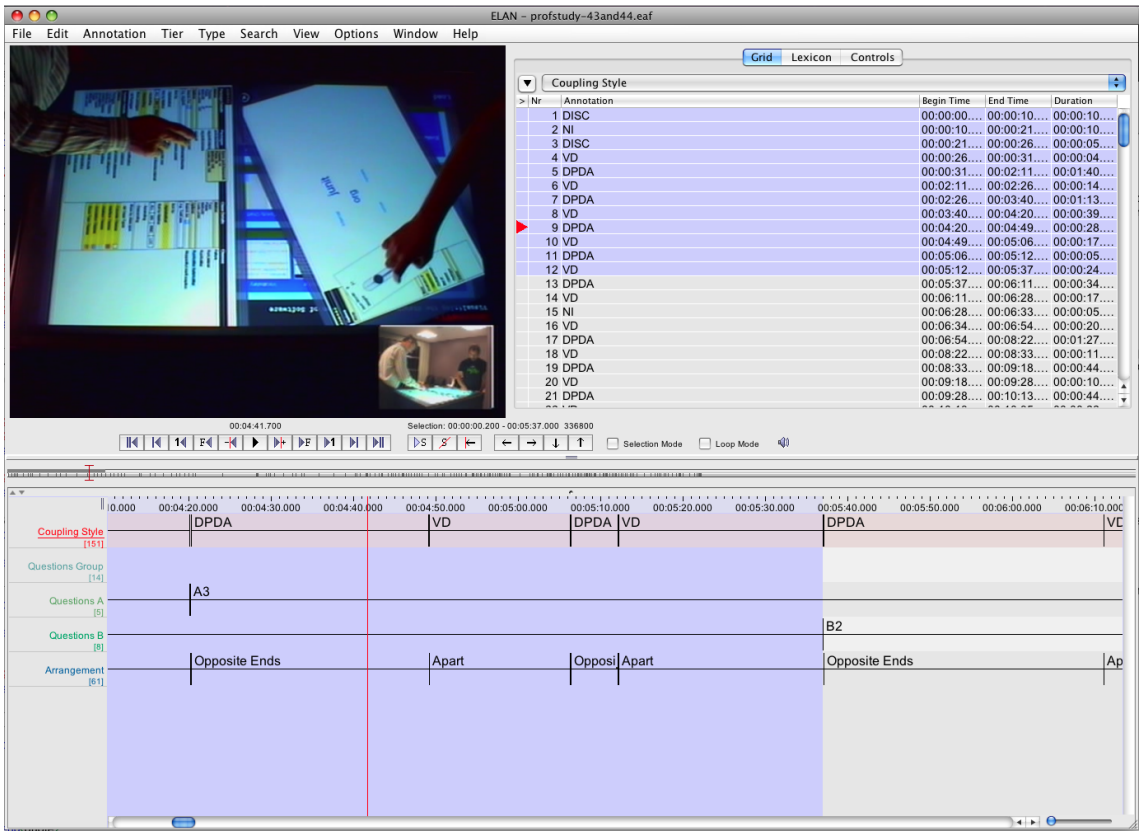


Figure 6.4: ELAN Video Analysis Tool. Top left window image is the video output. Top right individual tiers for the coupling style, questions, and arrangement codes. Bottom codes in tiers displayed in swimming lanes.

Part IV

Research Findings

Chapter 7

Professional User Study — Qualitative Findings

Contents

In this chapter we present the *qualitative* findings of the professional user study which was described earlier (§6). The findings are based on participants' feedback in the post-study questionnaire (§6.3.4), video recordings of the participants, spoken aloud thoughts by the participants, and observations and notes made by us during the study. The questionnaire asked participants to inform us about the strengths, weaknesses, and improvements of SourceVis and the multi-touch table, and how the multi-touch table helped with team collaboration.

7.1 Strengths

In this section we discuss the strengths of SourceVis and the multi-touch table as identified by participants and through our observations (§6.1.5 Q1).

7.1.1 Multi-touch Table

Group Work

We wanted to find out if participants favoured group or individual work. Participants experienced working in both conditions. All participants preferred to work as a group.

> "Ability to work with team members collaboratively made a nice change from working independently." PID 6.

> "The table seems to be much more effective and easier working together." PID 9.

> "It was so much easier working in a team. We both bounced ideas off each other. Both remembered different parts of the interface. We could also both select different parts." PID 10.

> "It was easy to work as a group." PID 13.

Some participants also liked that the multi-touch table allowed them to work as individuals at the same time.

> "The table being so big meant that you could have like you're own personal zone to do things with, so when something needed to happen on my side of the screen I did it, and when something needed to happen on his side of the screen he did it. Which I think worked quite well." PID 31.

> "The table allowed users to work independently in a small space. Multi-touch allowed for me to perform tasks while the other participant was working on a separate task. It also allowed us to work on different aspects of the same task at other times." PID 39.

Participants claimed that they were faster at completing the user tasks when working as a group compared with individuals. Some participants suggested using two machines for the individual condition.

> "Collaboration on the table makes finding solutions more efficient in a group." PID 3.

> "I found tasks were completed a lot faster when only one task was going on. This was due to the small text, window resizes and repositions, and the windows obscuring the initial screen where new applications can be loaded." PID 22.

> "When working on separate questions / problems I think that two separate PCs would have been easier." PID 24.

> "Team collaboration meant questions could be solved quicker." PID 27.

Multi-touch

An important aspect of SourceVis was the multi-touch interaction which allowed participants to manipulate objects with their fingers. For example participants could touch an object and then drag it to move it around on the table. Most participants had never used a touch screen of this size before so the novelty factor was high, and contributed to a generally positive user experience.

> "Being able to move items helped." PID 5.
>
> "Overall I really like the idea of manipulating visualizations by touch, with multiple people. It's useful to manipulate objects by touch" PID 17.
>
> "You could move stuff on the table with your hand – that was cool, as my colleague said." PID 42.
>
> "The multi-touch capabilities when working with a team mate wasn't painful." PID 43.

Participants felt the touch interaction allowed them to find answers to the questions within a few touches.

> "The ability to visualize the size of a given class, the relationships it maintains, its history all through a few touches is a lot of insights with little effort involved." PID 39.
>
> "The interactivity of the visualizations meant I could quickly play around and test out new visualizations. This was much faster than using existing visualization frameworks where they had to be defined using text files/drop down boxes/etc." PID 27.

Multiple Users

Participants liked that the multi-touch table allowed multiple users to stand around the table, use different sides, work together, and learn from each other. We observed many participants showing each other how to select options, close visualization windows at full screen, and display new visualizations through the pie menu.

> "Had a way of using different sides of the table and we just knew where to click and what it would do." PID 1.
>
> "The table helps because it is large enough to have several people round it." PID 8.
>
> "The size of the table is great for working with others." PID 26.
>
> "Good for sharing with groups, interactiveness is quite engaging." PID 30.

Table Size

Participants liked that the size of the table allowed them to interact, collaborate with each other, and make full use of the screen real estate.

> "Was good to be able to have a large full screen system, and one person read and point." PID 1.
>
> "Good to have a large interactive display that both people can touch and use at the same time as opposed to smaller devices that only one user can interact with at one time." PID 21.
>
> "Mostly good as a big table that we could both interact with." PID 24.
>
> "The large table makes it easier for multiple users to interact in things like code reviews and investigating class architectural structure." PID 32.
>
> "The large table made it quite easy to collaborate over shared input." PID 42.

Participants liked that the size of the table allowed multiple users to share viewing large amounts of data and to utilize the same workspace.

> "The device let us view large amounts of information well. Even when we were using different visualizations the device worked well." PID 16.
>
> "Because it's a big screen it's easy to look at data and reason about it with more than one person which is very useful." PID 17.
>
> "The large screen provides a shared view of the system, which can be collaboratively interacted with." PID 18.
>
> "The table was big enough for two people to share the workspace nicely." PID 30.
>
> "The table was great to be able to share the space and have multiple windows open at once." PID 40.

7.1.2 Visualizations

Multiple Visualizations

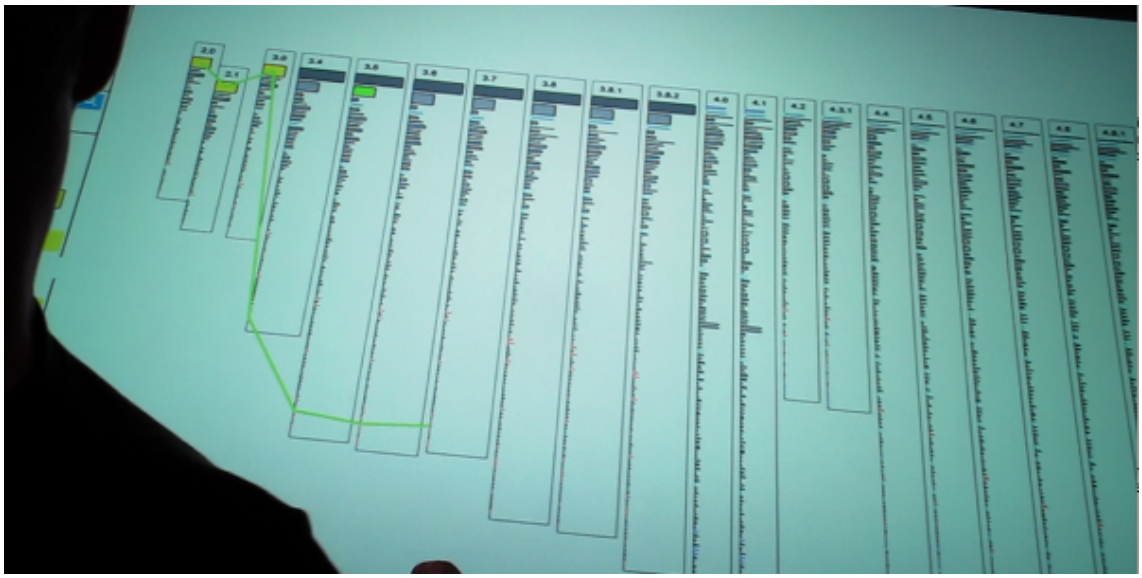
SourceVis supports 13 types of visualizations. Participants liked that there were many types of visualizations, and that multiple visualizations could be opened simultaneously to look at different parts of a system at the same time.

> "Being able to open several visualizations at once was helpful." PID 21.

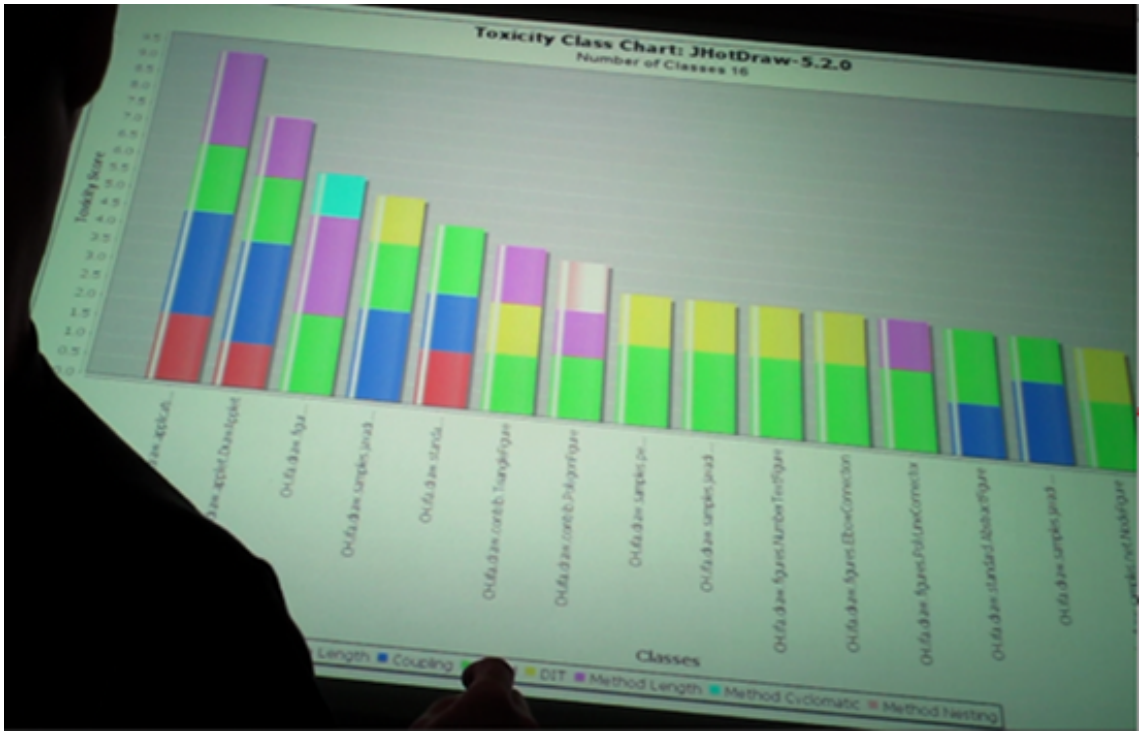
The visualizations supported different aspects of software systems and are grouped according to different categories such as exploration, structure, and

evolution. Participants liked how the visualizations covered a range of software development aspects.

> "The visualizations are grouped into useful categories and cover most of the common software development issues (e.g. coverage, dependencies)." PID 43.



(a) System Class Evolution Visualization and highlighted class versions.



(b) Toxicity Chart Visualization.

Figure 7.1: Evolution and Chart Visualizations.

Evolution Visualizations

The Evolution Visualizations showed trends and patterns about how software entities had evolved over many versions. The participants really liked how they could see a global view of the data (e.g. System Package Evolution and System Class Evolution) and then individual visualizations focusing on just packages or classes (e.g. Package Evolution and Class Evolution). Participants found highlighting a package or class in the either the System Package Evolution or System Class Evolution visualizations by using the version option in the pie menu helped put an entity into perspective against other entities. Figure 7.1(a) shows the System Class Evolution Visualization with one class being highlighted over seven versions.

> "Ability to view information across versions." PID 6.

> "Showing hidden structures (e.g. class evolution between versions)." PID 23.

> "I thought the Evolution Views were the most interesting in visualizing useful changes such as tracking classes and packages through versions. It gives an interesting way to view the history of a project." PID 38.

Chart Visualizations

Using charts to display information as in spreadsheet applications is a common approach to aid understanding data. Some participants liked how the Toxicity Chart, System Evolution, Package Evolution, and Class Evolution visualizations used charts to display information. Figure 7.1(b) shows a Toxicity Chart with different metrics and the toxicity score adjusted.

> "The evolution ones, for instance, which generated charts, were nice." PID 4.

> "Definite strengths are in seeing the Toxicity Chart, which I would definitely use day to day as a regular QA check – hopefully automated with the view sent directly to the developers." PID 26.

> "The ones with charts, were easy to see which entities were the biggest or smallest." PID 31.

7.1.3 Interaction

Navigation

Navigation across all visualizations was done through two finger gestures for zooming, panning, and rotating (see Figure 7.2). All participants found the zoom features helpful. Zooming allowed participants to see more detailed information about an entity and reduce the amount of information visible at once. Figure 7.2(a) shows a participant performing a zoom gesture to find details about a component in the System Dependency Visualization. Figure 7.2(b) shows a participant performing a pan gesture to navigate within the System Package Evolution for a large system.

> "Reasonably easy to navigate once the interface is understood. Clearer than a wall of text and statistics." PID 2.

> "The ability to zoom in on details was also handy." PID 4.

> "Zooming in on the particular class/package/version and then select the the thing you needed. Much easier than trying to use grep/wc -l." PID 10.

> "Ability to zoom in helps to mitigate the masses of information which can be thrown at you from time to time." PID 32.

The windows which displayed visualizations could be rotated in any direction. The System Dependency Visualization allowed users to rotate the circle in any direction with fingers or hand gestures so that text labels could be read from any side (see Figures 7.2(c) and 7.2(d)).

> "Being able to zoom into large diagrams, or rotate them (in the case of the dependency wheel diagram)." PID 8.

Visualization Switching

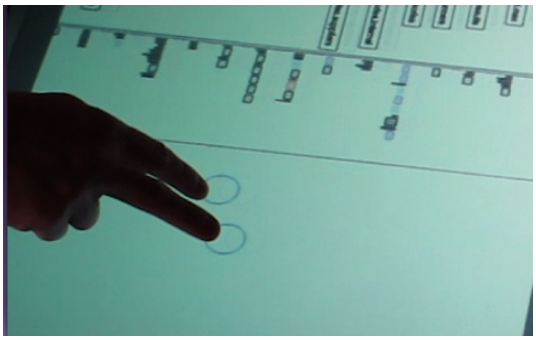
Participants often needed to switch between visualizations to answer the questions in the study. Participants liked how they could switch to another visualization by launching a new visualization from the startup screen or through the pie menu (see Figure 7.3).

> "Having a pie menu with links to dependency/evolution visualizations for a selected object." PID 8.

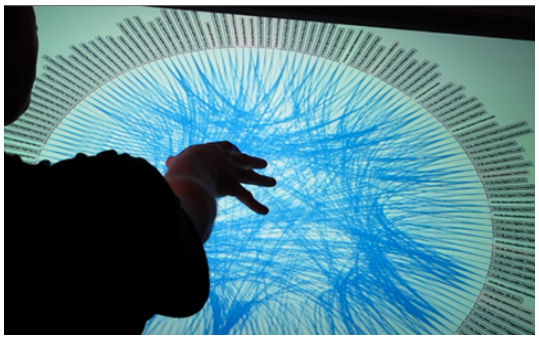
> "I really like the pie chart graphic menu for drilling down." PID 26.

> "The ability to open visualizations on the data you are viewing (e.g. opening the dependency visualization for a class)." PID 28.

(a) Zooming gesture with two fingers.



(b) Panning gesture with two fingers.



(c) Rotating gesture with hand at full screen.

(d) Rotating gesture with hand in separate window.

Figure 7.2: Navigation Gestures.

The pie menu was not fixed to any part of the display. Participants could display the pie menu after tapping and holding on a package or class entity, or on a chart. Many participants found the pie menu easy to use.

> "Easy switching between views by the popup menu to bring up more detail." PID 13.

> "The pop up pie menu was very effective and intuitive." PID 24.

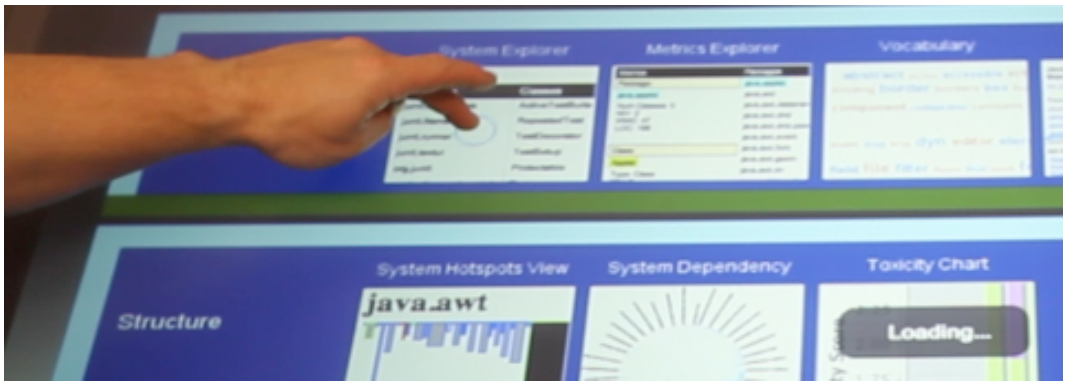
7.1.4 Data

Data Overview

Each visualization begins by displaying an overview of the data. Participants liked how they could get a quick understanding of a system. The visualizations helped them to highlight points in a system that were worth exploring further.

> "Good selection of system overviews." PID 5.

> "I could quickly tell by the pictures how a class depended on something." PID 7.



(a) New System Explorer visualization being launched from the startup screen.

(b) New Class Blueprint visualization being launched from the pie menu.

Figure 7.3: Visualization Switching.

> "Being able to have large scalable visualizations is quite a relaxed way to understand a lot of information quickly." PID 12.
>
> "Easy to read, quick assessment. Highlights areas that need a closer look." PID 15.
>
> "Useful for getting a global view and then focussing on specific aspects." PID 18.
>
> "See things at a glance once you get used to the software." PID 20.
>
> "It was an incredible overview and a quick way to get down to the ground on a project." PID 29.

Some participants found that the overviews visualizations of information about a software system may be easier to grasp than having to trawl through many source code files to find what one is looking for.

> "Can get an overview of the code and where the weaknesses are in a system much quicker than looking at the code directly." PID 35.

> "The visualizations display huge code bases, that can not practicably be viewed without a great deal of time and effort in existing tools." PID 44.

Details on Demand

Participants liked that the visualizations could provide details about entities on demand. For example the Metrics Explorer shows all the packages from a system and a user can select a package to view all the classes contained within the package. Entity properties such as metric details about individual classes and package can be displayed (see Figure 7.4(a)).

> "Being able to drill down into the details of everything viewed in a sensible fashion meant that with minimal instruction the user can work out what to do to show the information wanted." PID 11.
>
> "The ability to drill down to other levels of detail." PID 13.
>
> "You can pick specific objects within a larger visualizations, and learn various details. Most answers were discoverable in this way." PID 18.
>
> "Good to be able to drill down to the relevant level of detail." PID 24.
>
> "Drilling down through packages to classes worked well." PID 25.
>
> "The fact that almost everything was touchable and gave some context options for drilling in further, which is what you want when you're trying to identify specific things to refactor." PID 32.
>
> "The ability to deep-dive into selected items proved helpful when investigating particular packages and classes." PID 39.

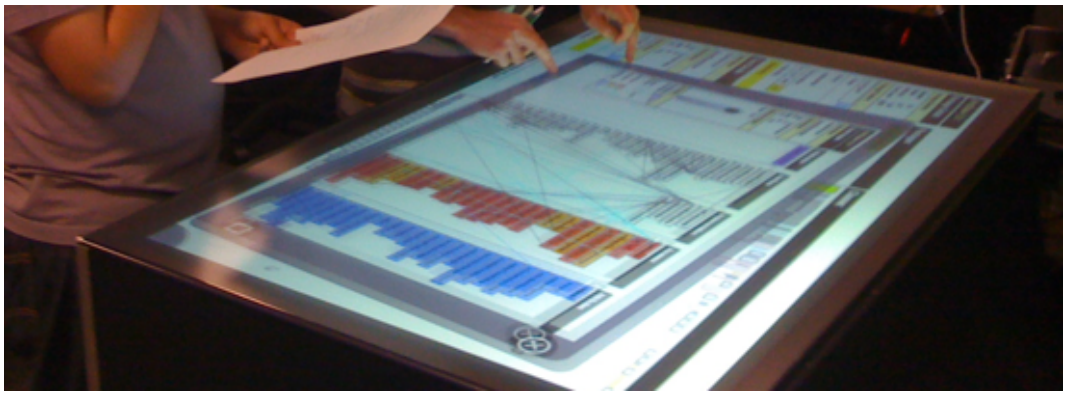
All of the visualizations allowed participants to show "Details on Demand." New visualizations could be displayed from existing visualizations that gave specific information about entities (see Figure 7.4(b)).

> "Many of the visualizations allowed you to open up additional contextual layers based on selected classes." PID 27.

Data Manipulation

Participants liked the data manipulation features for highlighting entities, sorting entities, filtering by class type, filtering by number of displayed entities using a slider, and searching for entities via a keyboard. Being able to move entities around on the screen, in particular packages and class entities, was also welcome.

(a) Displaying class properties from the System Hotspots View.



(b) Displaying a Class Blueprint from the System Hotspots View.

Figure 7.4: Details on Demand.

Highlighting in the Evolution Visualizations allowed users to easily trace packages and classes across different versions. Figure 7.5(a) shows the System Class Evolution Visualization where classes have been sorted and the largest class is highlighted and shows what other versions the class is located in.

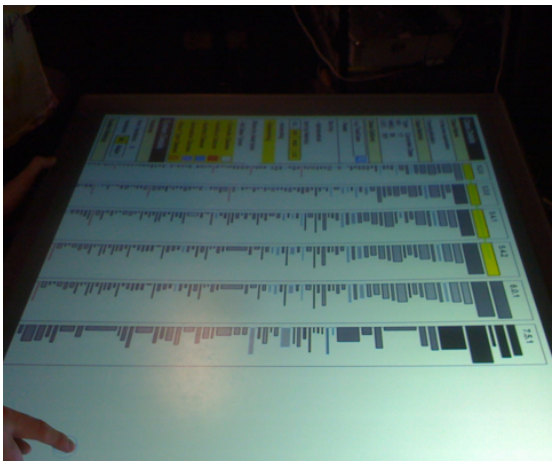
> "Easy to find and highlight the information that was being sought." PID 21.

> "Highlighting and filtering were the most help when answering." PID 22.

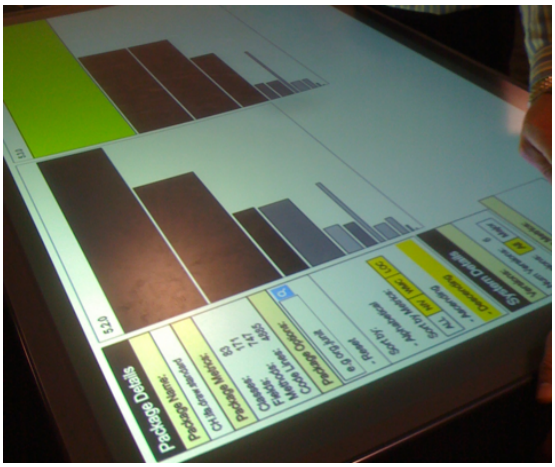
There were options for sorting entities alphabetically, and in ascending or descending order (by different metrics) which participants liked. Figure 7.5(b) shows the System Package Evolution Visualization where packages have been sorted in descending order. The packages on the left of the image are the largest packages and one package has been selected.

> "I liked the sorting." PID 13.

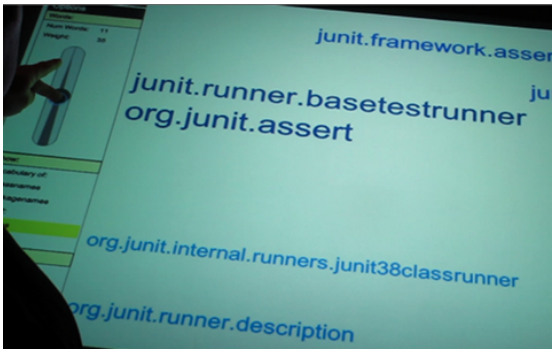
> "Sorting and filtering." PID 23.



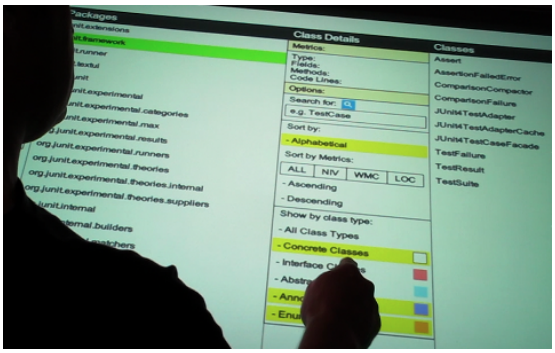
(a) Highlighting versions of classes in System Class Evolution Visualization.



(b) Sorting packages in descending order in System Package Evolution Visualization).



(c) Filtering classes by the slider in Vocabulary Visualization).



(d) Filtering classes by class type in Metrics Explorer Visualization.

Figure 7.5: Data Manipulation features, highlighting, sorting, and filtering.

> "Showing the sizes of things visually, and sorting them relative to one another." PID 33.

Participants particularly liked the filtering options as it allowed them to see only the entities they were most interested in. Figure 7.5(c) shows the Vocabulary Visualization where small classes have been filtered out by the slider. Figure 7.5(d) shows the Metrics Explorer where interfaces (red) and abstract (blue) classes have been filtered out, leaving only concrete classes on display.

> "You can remove details from the imagery using filters. Useful when questions are about specific kinds of things (e.g. concrete classes v. interfaces) and for de-cluttering the image to make it more understandable." PID 18.

> "Liked the ability to filter to narrow down the results." PID 24

> "Filters and sliders were good for drilling down." PID 25.

> "I liked the vocabulary one the best - it was the easiest to see quickly which were the classes I was looking for and filter out others." PID 31.

The searching options allowed users to find entities based on name. Entities that did not match a search query were hidden.

> "The visualizations where you can search were good." PID 31.

Participants liked to manipulate data by being able select entities by touch and then drag them around the canvas of the visualizations.

> "Being able to adapt the layout of items in a visualization to my needs." PID 6.

> "The drag and move options are quite handy." PID 30.

> "Moving and clustering of stuff." PID 41.

Many of the objects in a visualization could change size with gestures for scaling, including visualization windows. Resizing objects helped show more information within the visualization. For example, in the chart based visualizations, the charts could be scaled with a zoom gesture which would update the size of the chart (see Figure 7.1(b)). Increasing the size of the chart can also make the labels and legend more readable.

> "Resizable imagery." PID 18.

The data manipulation features and metrics data were useful for participants to be able to answer the questions.

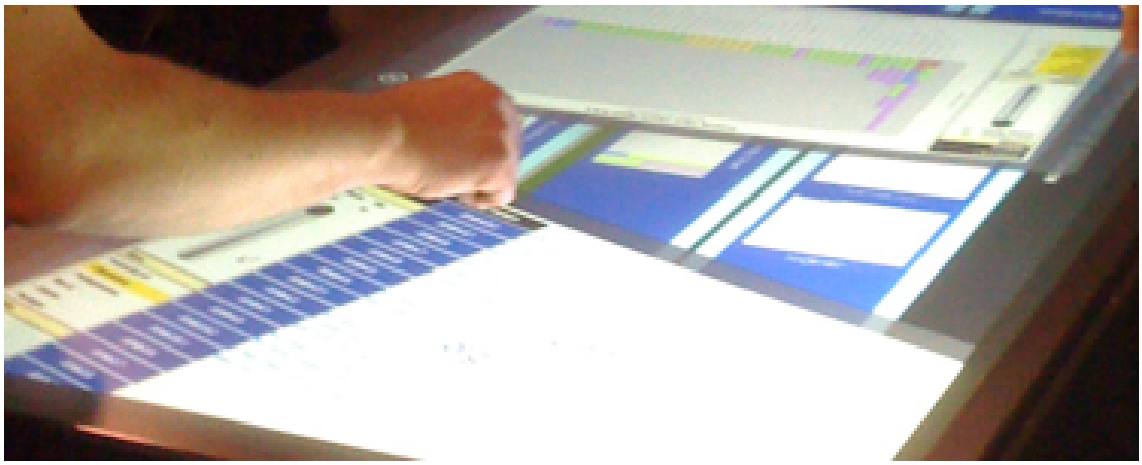
> "The summaries of the current displayed information (e.g. count, current filter levels) made answering some of the questions much easier, no focus was taken away from the task to keep track of other information (e.g. a count), the info needed was displayed once the correct filters were set." PID 11.

> "The visualizations were very helpful in answering the questions and I can see how they could be built into regular quality meetings (reviews/inspections). I have had little experience of using such metrics in real developments as I think developers tend to work on hunches (for good or bad)." PID 26

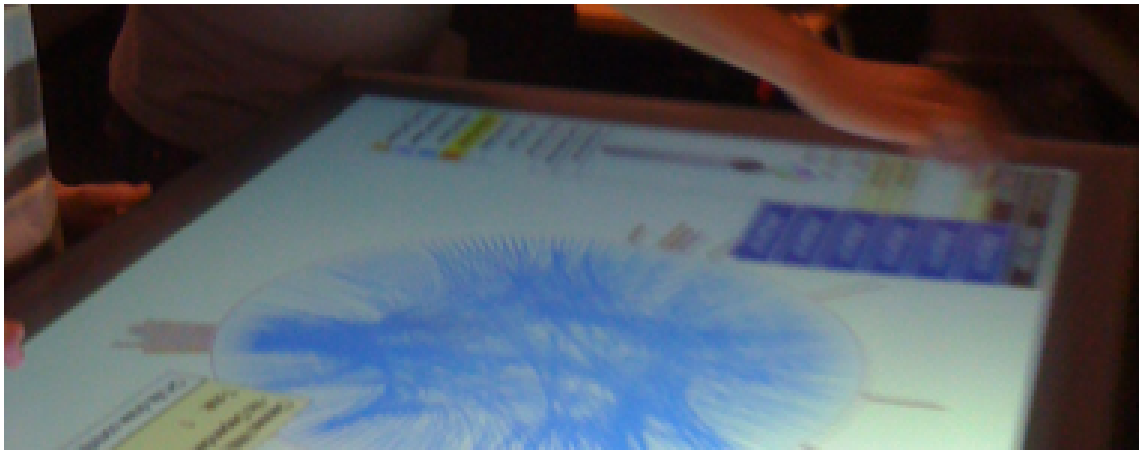
Data Switching

Participants liked how the visualizations could show different versions of a system and they could change the data directly within the visualization. To switch between one version and another we created a system version menu for the system and exploration visualizations. The Evolution Visualizations showed either all or only the major versions of a system. Selecting a system version name in the menu updates the data in the visualization (see Figure 7.6).

> "Helpful links between different systems and the context switch to view different data. This made it easy to navigate to the object you were looking for, then link to another explorer system to get the information/metrics needed." PID 1.



(a) Vocabulary Visualization.



(b) System Dependency Visualization.

Figure 7.6: Data switching to a different version using the system version drop down menu in dark blue.

Data Outliers and Trends

Participants found that the visualizations provided a good way to identify outliers such as large or small entities, problematic entities that could have potential design issues, entities that have no dependencies and no references, and entities that offer refactoring opportunities. This was particularly the case for visualizations that employed Polymetric View encodings such as the System Hotspots View, System Package Evolution, Package Evolution, System Class Evolution, Class Evolution, and Class Blueprint.

> "Instead of just numbers on the screen, the visualizations assist in identifying outliers and patterns more easily." PID 3.
>
> "Showed oddities such as interfaces and large classes stood out." PID 23.
>
> "Getting an overview of large code bases, highlighting outliers and instances with potential OO design issues." PID 34.
>
> "Quick access to key stats which help identify god classes and dead classes." PID 36.
>
> "Good to find weak points in the software." PID 43.

Participants found that the Vocabulary, Toxicity Chart, Structure, and Evolution Visualizations allowed them to identify information that is generally not easily obtainable in current IDEs.

> "Ability to see information such as hotspots that aren't traditionally available in development environments." PID 6.
>
> "Visual approach suggests things to look at that otherwise might go unnoticed in IDEs. Called out some interesting features of the code base." PID 37.

Participants found the Evolution Visualizations were good at showing data trends over time such as systems that have increased or decreased in size over time.

> "I thought the Evolution Visualizations were best at showing overall trends." PID 4.
>
> "Visualization is generally useful to find points of high complexity. The Evolution Visualization helps recognizing trends of code deteriorating, before it causes headache." PID 14.
>
> "Being able to track the life-cycle of a given class/package in the Evolution Visualizations is really powerful." PID 39.

7.1.5 User Interface

Intuitive and Engaging User Interface

Most participants found SourceVis intuitive and easy to learn.

> "The interface was very quick to pick up." PID 10.
>
> "Fairly intuitive use after minimal training." PID 21.

Participants found the interface clean, simple to use, and information was easily accessible.

> "Information is easy to access, there are no deep levels of dialogues or windows (relatively shallow)". PID 19.
>
> "Interface is simple, not cluttered." PID 39.
>
> "Good use of graphics, it is easy to read information when it's in a graphic format." PID 43.

The user interface was novel for participants and we hoped that it would be engaging for users to explore a system. Participants found the concept of exploring a software system using a multi-touch interface with multiple users engaging.

> "Engaging interface (i.e. move things round with hands and explore with minimal guidance) may be harder to use for precise expert tasks such as investigating source code." PID 8.
>
> "Awesome interactive discussion. Love the idea. Great way to explore a system." PID 15.
>
> "There's something very good about the physical interaction that helps with invigorating meetings (we really like to use post it notes and whiteboard markers for planning)." PID 24.
>
> "Multi-touch table allowed for multiple simultaneous exploration and encouraged the users to have some fun with their individual explorations and interactions." PID 27.
>
> "Two people exploring the data in different ways simultaneously and conversing about it is great. Having multiple windows that you can pull to the side makes this easier." PID 36.

Visual Encoding Representation

We used a few different representations for encoding data including: lists, graphs, charts, and Polymetric Views [185]. Many participants liked the Polymetric View

encoding for displaying packages and classes to get a perspective of how big or small an entity was.

> "I could quickly tell by the pictures how big something was." PID 7.
>
> "Size based graphical representation." PID 13.
>
> "Good visual representation of code structure." PID 16.
>
> "Being able to see graphical links between classes or packages, and see a physical difference between items of differing size or complexity was really powerful. Being able to see at a glance how big or complex something was rather than having to look at the numbers made it much easier to find an answer." PID 40.

Participants liked how we used graph like views for the dependency and Class Blueprint visualizations.

> "Graphical/area views were good. Lines connecting components between windows were good." PID 5.
>
> "Various graphs and metrics are definitely helpful." PID 7.
>
> "Graphs are still very useful. I would have assumed that they were too dated in comparison to the other styles of visualizations." PID 12.

UI Consistency

Many aspects of the visualizations were designed in a consistent manner including representation of elements (e.g. menus and encodings), manipulation of elements (e.g. drag, tap, and hold), and navigation interaction gestures (e.g. two finger zoom in and out, and pan). We observed that participants quickly understood the interface style after a short while.

> "Good application metaphors to achieve a consistent understanding across all applications (e.g. size and length of boxes). PID 19.
>
> "Interface was nice and consistent once I started to get the hang of it." PID 24.
>
> "Common look and feel. Once we got the hang of it, the questions were quicker to answer." PID 25.
>
> "Long press selections and selection clicks are consistent and helped quickly understand the system." PID 36.

7.2 Weaknesses

In this section we discuss the weaknesses of SourceVis and the multi-touch table as identified by participants and through our observations (§6.1.5 Q2).

7.2.1 Multi-touch Table

Individual Work

We observed that most participants struggled with the individual condition. This was primarily due to issues with the graphics not being easily readable in separate windows and the screen resolution being too low hence limiting the number of items that could be displayed at once. Some of the pairs completed the individual tasks sequentially as opposed to completing them at the same time. We observed pairs 3,17,19 and 20 completing the individual condition as a group.

> "Working as an individual didn't really offer any huge advantages." PID 9.

> "It was tricky for two people to work independently, especially with the full-screen button! Perhaps have a half screen button." PID 13.

> "When multiple people were working on the same table on different tasks, screen real-estate became dear, making some the application harder to use." PID 21.

> "Not so effective for individuals answering different questions!" PID 37.

> "Initially I thought it would be possible to achieve a lot of work with two primary windows open (i.e. completing separate tasks concurrently). Although it was possible to have two separate visualizations open at once, the text scaling meant that the controls were small and blurry and so it was harder to complete tasks than when we were sharing the screen together. This meant that instead of completing our tasks in order, we needed to work out which visualizations we both needed, open them in full screen, and complete our tasks together, rather than work completely independently. This wasn't a problem in this task, but if we had two completely independent tasks to perform it would have been a problem." PID 40.

Touch Experience

The touch experience was generally quite acceptable for most participants, however, there were issues with the responsiveness of elements and objects behaving differently in the visualizations which decreased the usability of the interface.

> "Usability issues with responsiveness and smoothness of the graphics." PID 17.
>
> "UI elements were sometime unresponsive. This was the most frustrating thing for me." PID 18.
>
> "Touch table was sometimes hard to operate." PID 25.
>
> "Often the visualization was hindered by the usability of the touch experience." PID 28.

The navigation gestures can work slightly different depending on which fingers participants were using, how much of a finger was touching the surface. Some participants felt the gestures were inconsistent across the visualizations.

> "Inconsistent touch experience, sometimes one finger to move, sometimes two, sometimes two fingers dragged out a selection box." PID 5.

Occasionally when participants decreased a window in size by manipulating the borders, the window would disappear momentarily until the they then increased the size of the window. If they released the touch points the window would sometimes completely disappear. When two people were navigating on the canvas such as zooming, or panning the application would get confused as to what gesture action to perform.

> "Too many fingers at once on the screen can cause the application to behave weird with the windows." PID 3.

Touch Precision and Accuracy

As described earlier (§3) we built our own multi-touch table using open source techniques. We calibrated the table prior to each user study. As the table used a RearDI setup, natural lighting and the size of the finger touching the surface sometimes affected the precision and accuracy of the touch points.

> "Fine touch control seems to be a bit of a problem." PID 20.
>
> "Accuracy of the touch points. Sometimes I had to press more than three times for the same point to be recognized." PID 43.

As many of our participants had experience with mobile phone and tablet touch screens that use capacitive technology (§6.2.2), they were accustomed to better touch precision and accuracy.

> "Some of the touch events were a little too easily triggered, although that might just be because I am used to capacitive touch screens that are far less sensitive." PID 39.

Screen Resolution

Participants found the resolution of the display screen at 1280x800 pixels was too small. This size is smaller than most contemporary computer desktop displays. We would have preferred to use a greater resolution but we were limited in the capabilities of the projector chosen for this project.

> "Resolution of most images would be best on a bigger screen." PID 19.
>
> "For some visualizations the screen is still too small and too low resolution." PID 33.
>
> "Resolution not great." PID 35.

This low resolution made it hard for some participants to read text that was being displayed.

> "Pixel density was way too low to read text at anything less than full screen." PID 5.
>
> "Too much data coupled with low resolution meant that the text was unreadable in most cases." PID 6.

The low resolution size also affected the usability for some tasks when interacting with visualizations.

> "Even with higher pixel density some of the items were too small to reasonably select." PID 5
>
> "Resolution and power of the device probably hindered some usability." PID 7.

Hardware Performance

Given SourceVis was a prototype application and the multi-touch table was custom made, we expected there to be system performance issues for both the software and hardware. Quite often the participants experienced the Toxicity Chart visualization crashing as the visualization re-rendered the data every time there was a slider

modification. Occasionally many touches by participants on the table at once caused issues which threw some run-time exceptions for the MT4j TUIO Java client and SourceVis crashed.

> "A little bit crashy." PID 9.

The system was sometimes slow when CCV had to track many fingers of the participants at the same time, and there was lots of data that required rendering in the visualizations.

> "Performance was problematic, lots of waiting for some visualizations." PID 5.
>
> "When more than one person was using the table, sometimes was slow to respond." PID 43.

Prolonged Use

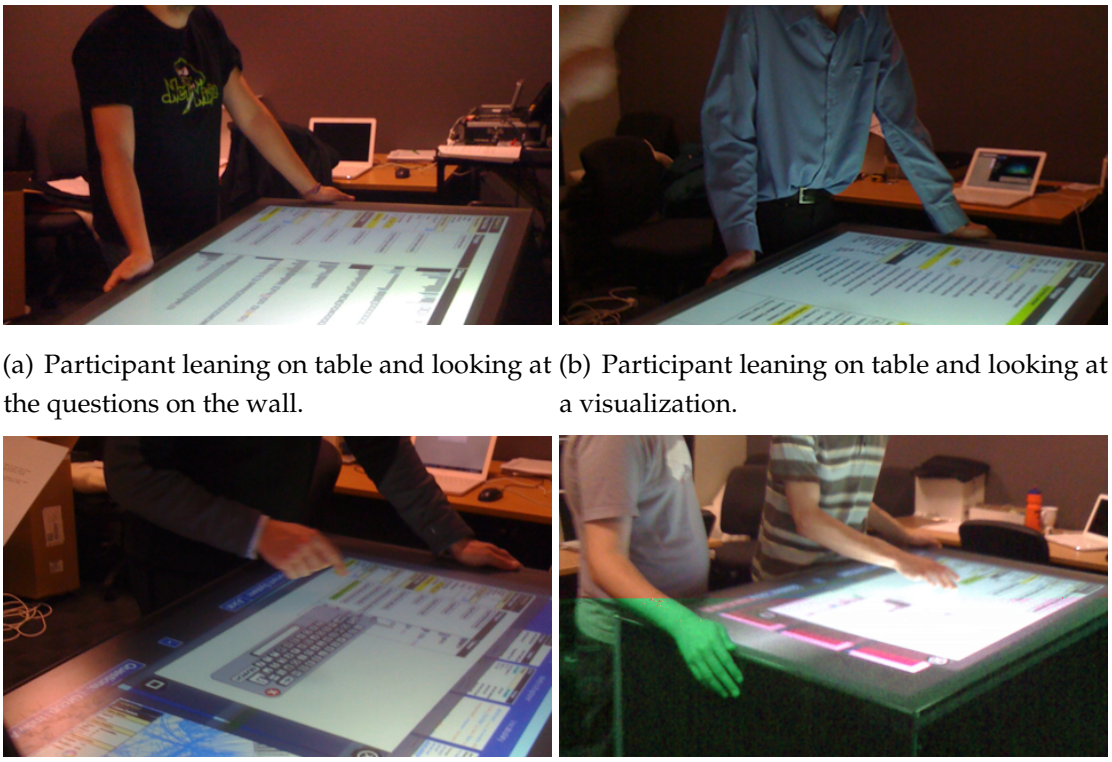
We anticipated that participants would use a touch table of this nature for up to 90 minutes as part of the study (including pre study exploration and user tasks) and not for long periods of time in a work place environment. Participants interacted with the table by standing up. We did not provide seating for the participants to use when interacting with the table. Subsequently some participants felt uncomfortable having to stand and lean over the table to use it. Quite often we saw participants leaning on the table with two hands during the user study (see Figures 7.7(a) and 7.7(b)), or leaning on the table with one hand while interacting with the other hand (see Figures 7.7(c) and 7.7(d)).

> "After 90 minutes of use, my fingers are starting to feel a little bit of tap-numbness, and my back is a little sore from craning over the table." PID 4.
>
> "Found it a bit uncomfortable leaning over the table for a long period without something to lean on." PID 24.

7.2.2 Visualizations

Visualization Context

Some participants felt the number of visualizations was quite overwhelming. For example some participants forgot what visualization they were viewing, as some of the visualizations were very similar (e.g. System Explorer / Metrics Explorer, and Package Evolution / Class Evolution). We did not add any labels to the



(a) Participant leaning on table and looking at the questions on the wall.

(b) Participant leaning on table and looking at a visualization.

(c) Participant leaning on table while interacting with a visualization.

(d) Both participants leaning on table while one is interacting with a visualization.

Figure 7.7: Participants leaning on the table during the user study.

windows nor to the visualizations once they were displayed at full screen. Adding labels for each visualization would have put a visualization into context.

> "Windows titles were not displayed so you didn't know which window you were using." PID 7.
>
> "Lack of titles on windows." PID 8.
>
> "Found it a bit confusing working out what level we were at (packages / classes) without referring back to the title of the window." PID 24.

With visualizations that utilized large amounts of screen space participants were confused as to where they were in a visualization and found it difficult to keep track of what they had seen previously.

> "Getting lost in the complexity of the tool and the vocabulary visualization (but experience would help)." PID 13
>
> "I sometimes got confused about where we were in the screens." PID 16.
>
> "Very complex, hard to keep track of functionality." PID 23.

System Dependency Visualization

We observed that most participants struggled with the System Dependency Visualization because of the amount of information on display and lack of some features (see Figure 7.8(a)). It was one visualization that did not scale well for very large systems. For large systems it was hard to read the text of class names and there were no search features.

> "System dependency visualization was too hard to read to be usable." PID 23.

> "In the case of the dependency visualization, with a large dataset it wasn't possible to see the actual labels of the packages or classes unless you zoomed in to only see a fraction of the data." PID 40.

The direction of the dependencies among classes was difficult to understand and could only be determined when selecting a class. There were no arrows pointing the direction of the dependency. Figure 7.8(b) shows a participant selecting a class and the subsequent dependency edges are highlighted in green.

> "Dependencies direction wasn't obvious when filtered down, had to click the class to know what direction it was going." PID 22.

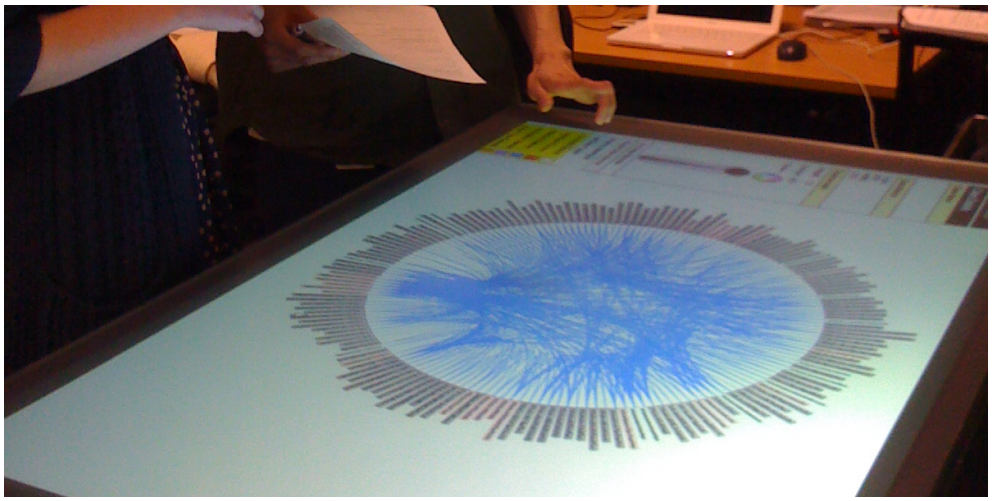
> "The class and system dependancy visualizations contained lots of lines which were confusing and hard to tell quite what was going on." PID 31.

When entities were filtered out by class type their dependencies were not filtered out and the outer circle did not shrink in size (see Figure 7.8(c)). Both of these issues contributed to participants having usability problems with this visualization.

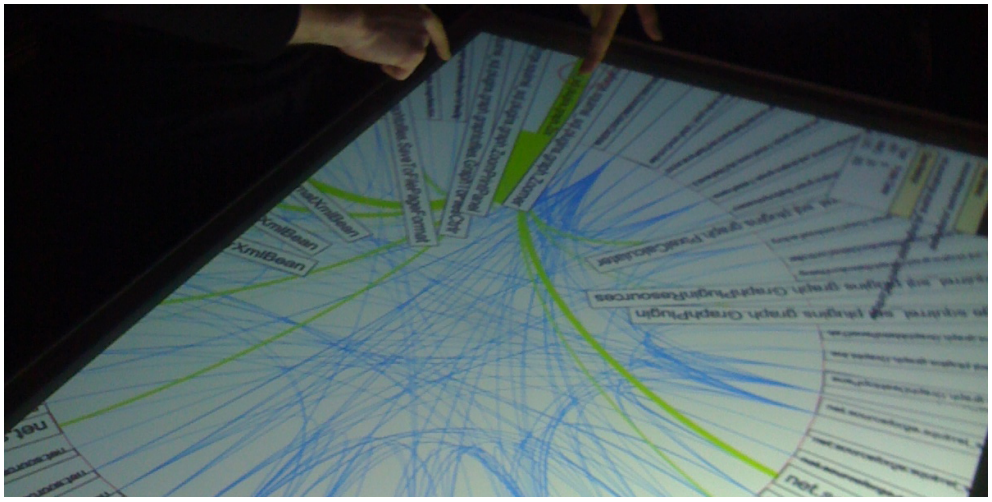
> "Not enough filtering for example the system dependancy does not get much simpler to see as you filter down using the scale because the names remain in the wheel, making it no easier to read." PID 12.

One participant questioned the layout as it why it was circular. Part of the reason for the visualization being circular is to promote reading the names of classes from different sides of the table by rotating the circle in the middle with gestures (see Figures 7.2(c) and 7.2(d)).

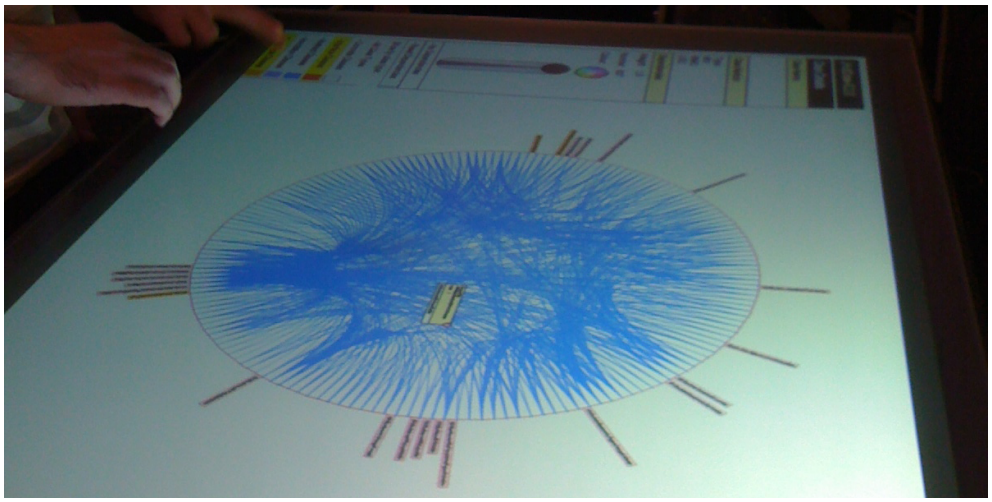
> "I still found the dependency visualization difficult to understand. Why was it circular?" PID 18.



(a) Dependency overload.



(b) Highlighting dependency edges.



(c) Filtering dependencies by class type.

Figure 7.8: System Dependency Visualization.

7.2.3 Interaction

Navigation

Many participants found that the navigation interaction gestures (panning and zooming) were not as effective or consistent as what they expected. We observed the participants struggling to perform navigation gestures on the canvas of visualizations at the same time when in the group condition.

> "Hard to zoom effectively, controls jumped around too much." PID 5.
>
> "Interaction in terms of zooming and moving components was inconsistent across visualizations." PID 6.
>
> "Scale when zooming was sometimes an issue." PID 40.

Lots of manual zooming was required in visualizations for large systems in particular the more space intensive such as the Package and Class Evolution Visualizations.

> "Often the visualizations require a lot of manual resizing." PID 33.

Some participants would have liked to have seen traditional desktop scroll bars for navigation integrated with the visualizations.

> " No traditional scroll features in the visualizations." PID 5.

Zoom Level

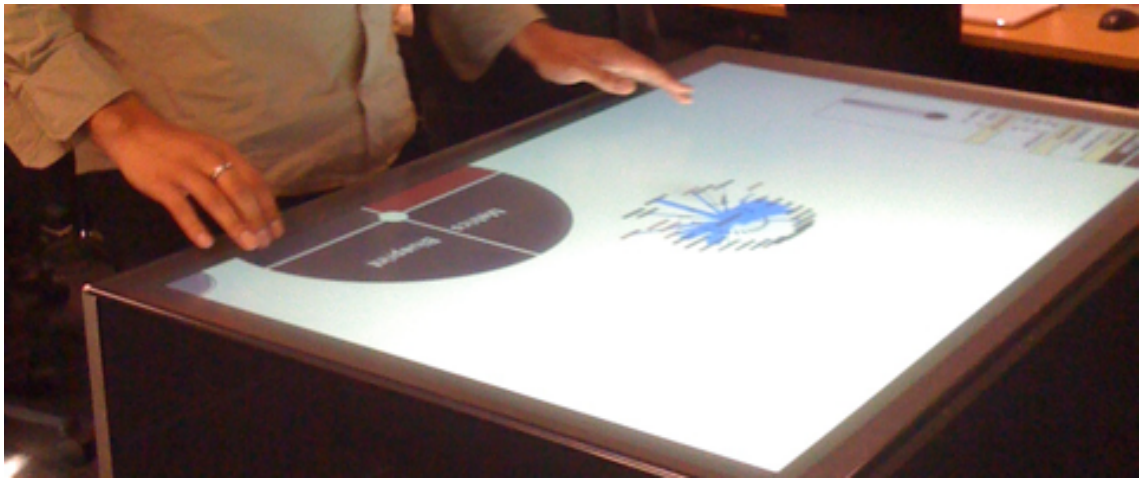
When participants zoomed in or out on the canvas and opened items such as a pie menu, keyboard, or properties about an entity, there were problems with the size of these items which affected their usability. Figure 7.9(a) shows a pie menu from the class dependency visualization where the participant has zoomed out quite considerably and the pie menu is too large and appears half off the screen.

> "A bit of difficulty zooming things correctly – sometimes the full area would be zoomed out and menus would hence be tiny." PID 8.
>
> "The way the zoom feature wasn't consistent and is a bit confusing. To enlarge the text on the left required making the entire screen bigger, but the graphs could be enlarged within the window." PID 30.
>
> "Dynamic scaling as you go deeper into packages/classes did mean that some of the keyboards and charts opened were quite small and had to zoomed, which could be fiddly sometimes." PID 39.
>
> "The pop-up pie chart menu was dependent on the size of the item in question, so selecting a very large display that was zoomed out to see

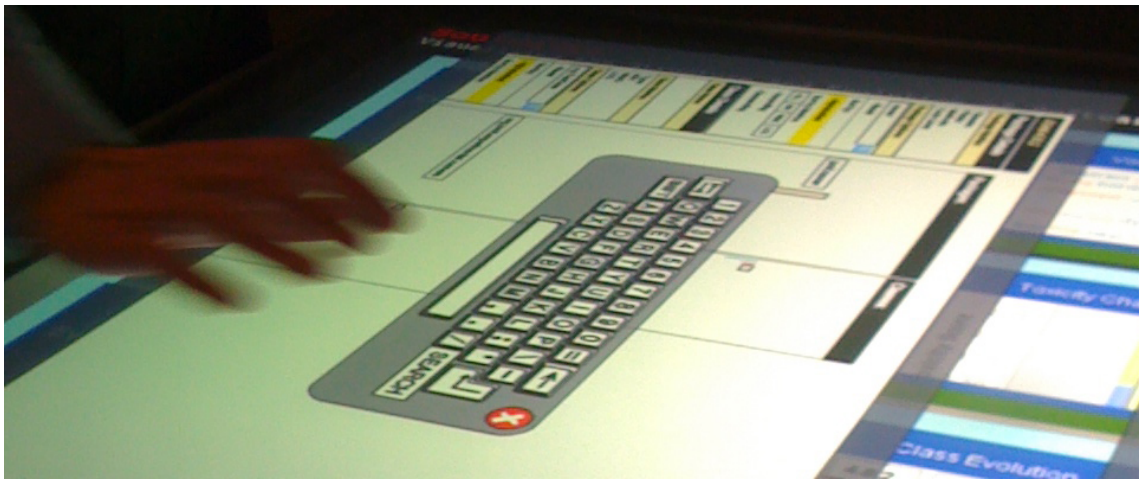
the whole image meant that the pie control that spawned was too small to see the labels." PID 40.

This zoom issue was quite problematic when participants were working in the individual condition as there was less screen real estate for each participant to display windows and visualizations. Figure 7.9(b) shows a small keyboard being displayed in a window and a participant needing to increase the size of the keyboard in order to use it properly.

"Zoom levels were an issue, mainly when doing individual." PID 29.



(a) A pie menu being much larger than the visualization and displayed half off the screen.



(b) A keyboard needs to be increased in size within a window.

Figure 7.9: Zoom level problems with menus and keyboard displaying inconsistently in size.

Manipulating Windows

We observed many participants having difficultly manipulating visualization windows. Increasing the size of the window required dragging two borders of the window at the same time (see Figure 7.10(a)). Moving a window around the screen can be performed by dragging the border of a window with one finger. In the individual condition we observed some participants dragging windows around the screen by using two fingers on different borders. For example they would stand at the bottom of the table open a visualization from the start screen and then walk with the window while dragging it to the side of the table they wanted to stand on (see Figures 7.10(b) and 7.10(c)). This appeared very cumbersome, and these participants were not aware you could drag with just one finger.

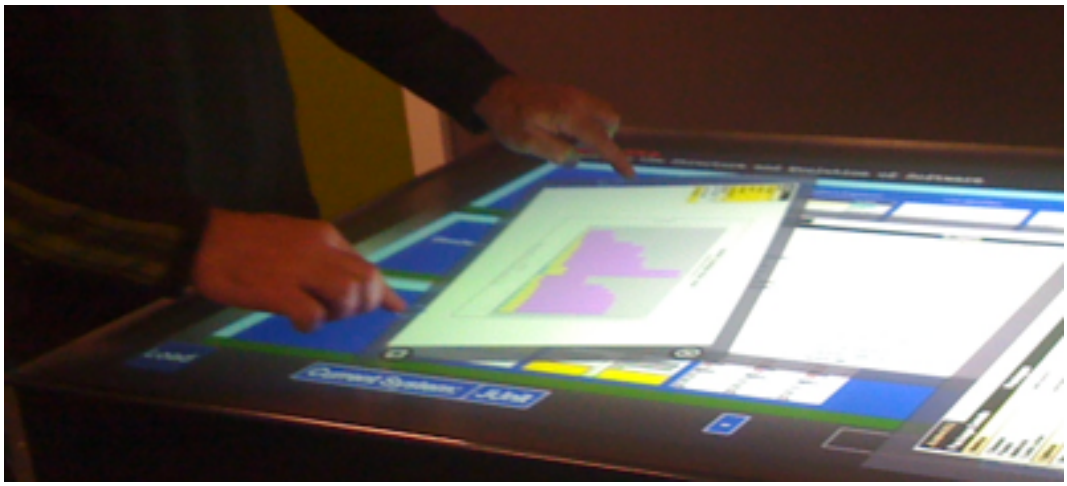
> "Mostly HCI related. Poor screen metaphors and stereotypes for managing windows and moving things around." PID 19.

We observed problems for participants in the individual condition when windows overlapped each other. If two visualizations were displayed and then one visualization was switched to full screen this prevented the other visualization from being viewed.

> "Inability to select that two windows were being used and so they should be immediately given half the screen real estate each to prevent overlapping." PID 6.
>
> "There was some overlapping of windows which slowed us down." PID 16.
>
> "The windows weren't adapting to the amount of screen space." PID 29.

When new visualizations were started they were displayed in the middle of the screen obscuring any visualizations already on display. Figure 7.11(a) shows one visualization being displayed by one participant standing on the right end side of the table and then another visualization is started by the other participant obscuring the view of the existing visualization. We observed participants moving windows which obscured another participants visualization window. Figure 7.11(b) shows one participant rotating a window which overlaps another participant's window.

> "New windows would pop up in the horizontal middle, obscuring anything that another group member was working on, plus I could completely obscure other people's windows." PID 30.

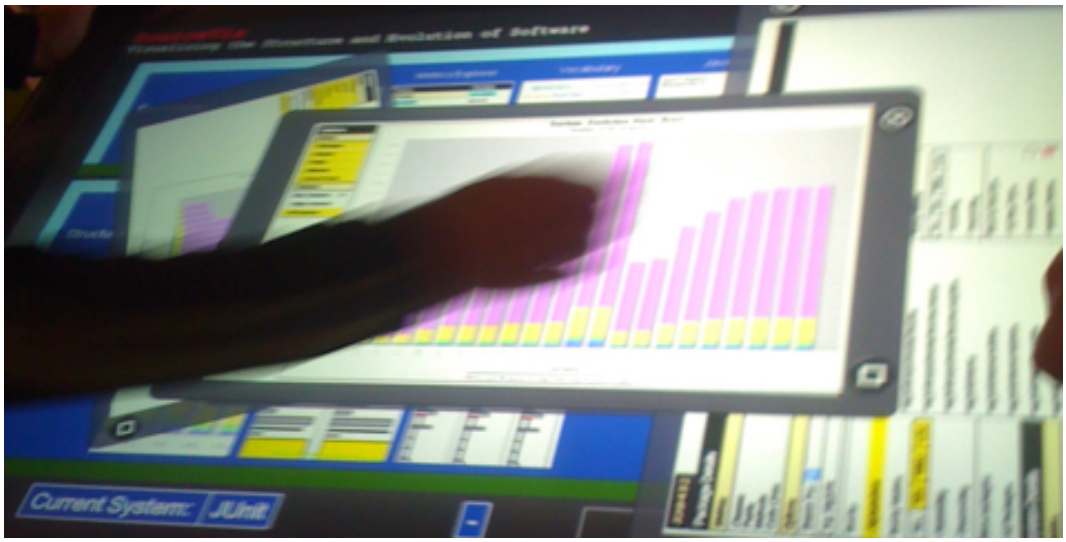


(a) Increasing the size of a window by dragging the borders.

(b) Moving a window to the side of the table by dragging two borders.

(c) Successfully moving a window to the side of the table.

Figure 7.10: Manipulating windows on the screen.



(a) A new visualization window displayed upon existing window.

(b) Rotating a window over another window.

Figure 7.11: Visualization windows obscuring each other.

We observed participants continuing working while visualization windows partially overlapped each other (see Figure 7.12(a)). Participants found managing windows within windows difficult, especially if the first window was not displayed at full screen. Figure 7.12(b) show two participants displaying windows within windows, which has decreased the amount of screen real estate as there are now four window borders being displayed. Some participants also found it challenging to move between windows.

"Sub-windowing was somewhat confusing." PID 13.

"Need to make it easier to move between windows." PID 19.

(a) Windows overlapping but participants continually working.

(b) Two participants managing windows within windows.

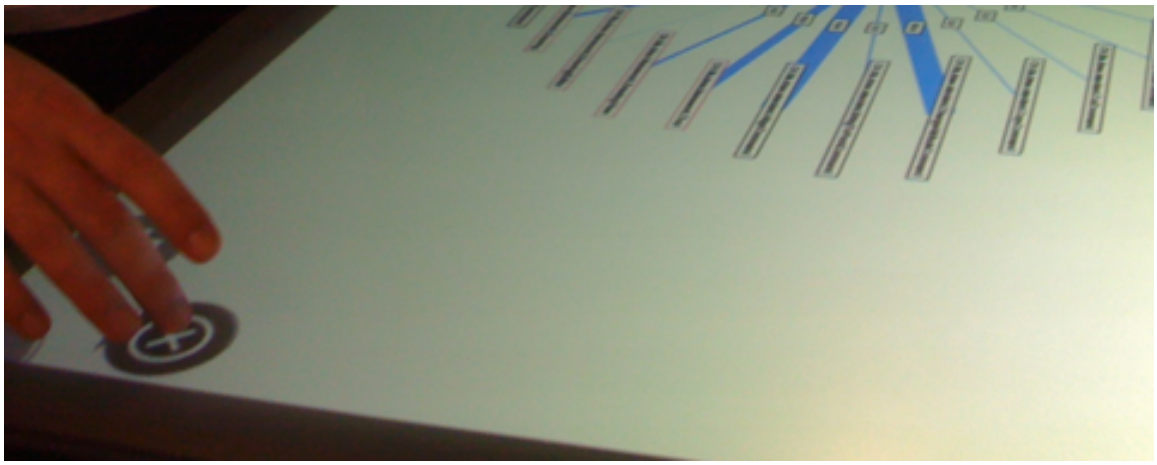
Figure 7.12: Participants working within visualization windows.

All participants had difficulty closing a visualization at full screen when using the close option at the bottom right of the screen (see Figure 7.13(a)). The difficulty was due to participants having to put a larger amount of their fingers on the corners of the table in order for CCV to detect their fingers. It was straight forward to close a visualization window and a property dialog box by using the close buttons in the top right of the borders (see Figures 7.13(b) and 7.13(c)).

"Close window button is hidden." PID 7.

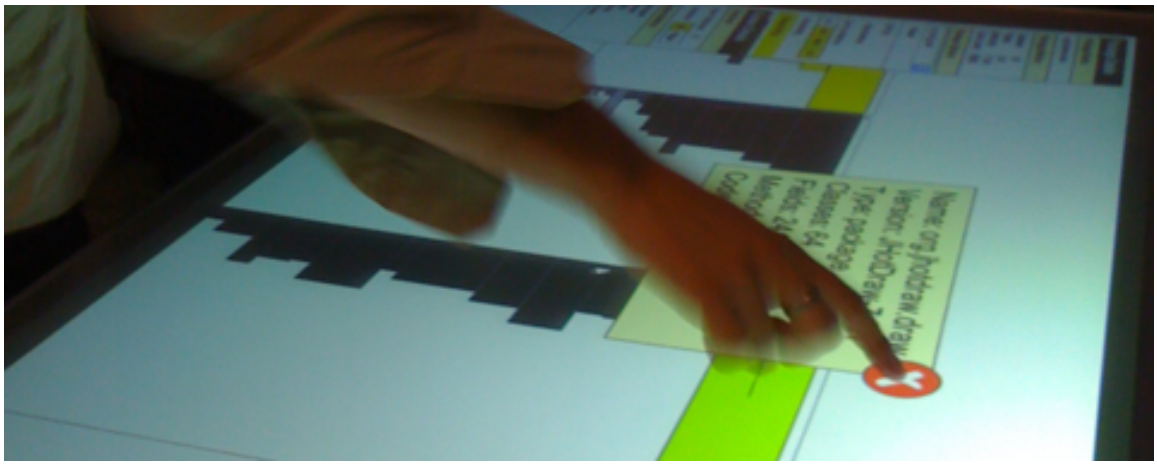
"The bottom right-hand button for closing." PID 30.

"Closing visualizations at full screen was a pain." PID 42.



(a) A participant having difficultly closing a visualization once displayed at full screen.

(b) A participant closing a visualization window with the close button located on the top right.



(c) A participant closing a package properties dialog box with the close button located on the top right.

Figure 7.13: Participants closing visualization windows and entity properties dialog boxes.

Entering Text and Data

The participants did not like entering text using the virtual keyboard. Entering text was easiest when the visualization was at full screen as the keys on the keyboard were bigger, making them easier to select (see Figure 7.14(a)). On the other hand when entering text inside a window the text was hard to read and the keyboard was constrained due to the size of the window (see Figure 7.14(b)).

> "On-screen keyboards are not nice to use." PID 2.

The layout of the keys on the keyboard used a QWERTY style. Given it was a virtual keyboard it would be possible to modify the layout of the keyboard with other layouts.

> "I normally type with the Dvorak keyboard layout, and the on-screen keyboard was Qwerty. I'm pretty used to that in all avenues of life, though." PID 4.

We tried to limit the number of ways to enter data into SourceVis, hence we used sliders and filters. Occasionally participants had usability issues with a few of these widgets. For example, we observed some participants on one question (§H.3 Q3) trying numerous times to modify the value of the slider to be a precise number in the Toxicity Chart. Part of the issue is that each time the slider was adjusted the chart would update and redraw itself, causing frustration for the participant. Some participants would have liked to be able to enter precise numbers with text input.

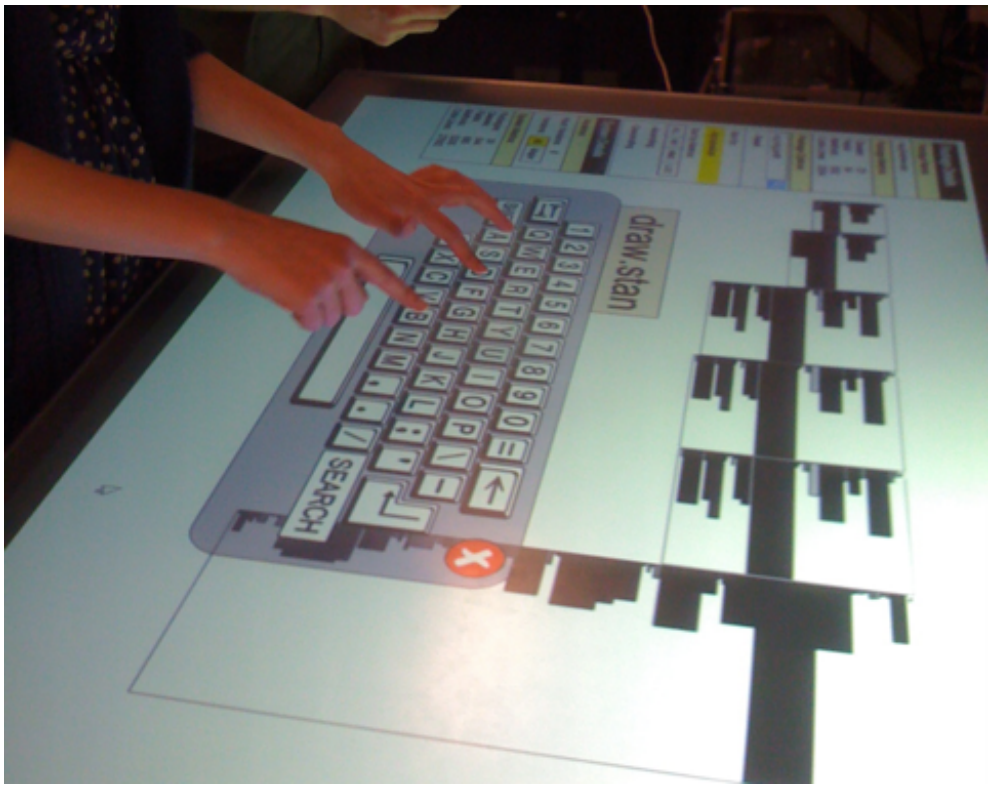
> "It was sometimes difficult to enter input as precisely as I would have liked. For instance, some controls were a slider, when I would have preferred to enter a precise number." PID 4.

User Interference

Although the multi-touch table allowed multiple users to interact at the same time this did not prevent user interference especially during the group condition. Occasionally we observed a participant take a visualization from another user, reach across another user, or dominate when interacting with the table and annoying the other user.

> "Easy to interfere with each other." PID 2.

> "Multiple users interfere with each other. Trying to do multiple things at once they just interfered with each other. Worked ok as long as users were working as a team to do one thing." PID 20.



(a) Entering text at full screen.

(b) Entering text inside a window.

Figure 7.14: Entering text using the virtual keyboard.

> "If both users were trying to do something, sometimes someone's interactions would get in the way of the others." PID 27.

We observed participants working as individuals and having separate windows open, which sometimes led to windows overlapping each other, and caused user interference and frustration (see Figure 7.11(a)).

> "If the tool is used with two people on different tasks, they often interfere with each other as the screen can't define separate workspaces that won't overlap when maximizing a window." PID 14.

> "People's windows would occasionally overlap each other." PID 30.

Sometimes when participants were working at the same time on different visualizations, one participant would want to display their visualization at full screen which meant the second participant had to stop working and wait for the first participant to complete their task.

> "Expanding the visualization to see better resulted in getting in the other team members way." PID 35.

When working as individuals, when participant had a window opened, it was hard for the other to launch a new visualization if the visualization icon on the start screen was beneath the opened window.

> "When working individually it can be hard to launch a visualization from the start screen when it is obscured by a window which is being used by someone else." PID 12.

7.2.4 Data

Information Overload

The visualizations provided lots of information for the participants. Sometimes there was too much information which could be overwhelming, cause confusion, clutter the screen, and make it hard to understand.

> "Overwhelming number of metrics for me, but that is likely because I am not too familiar with the metrics." PID 7.

> "Dealing with large number of items (e.g. dependencies circle with a huge number of items)." PID 13.

> "Some screens were complex to understand. Lots of information. Just need to play with it get a better feel." PID 15.

> "Clutter, it is to easy to be presented with too much information to process." PID 28.

Some participants felt that there was too much information that was displayed initially. This was especially the case for the System Hotspots View, System Dependency, System Class Evolution, and System Package Evolution Visualizations.

> "Too much initial information can cause confusion." PID 3.

> "The default views for some of the visualizations can be a bit overwhelming if for instance looking at a large package / class." PID 32.

Finding Information

Many of the visualizations required filtering to display the answer for the questions. Some questions required counting information or visually searching for colour coded items. Some participants did not like finding information by counting or visually searching and would have liked to have seen the answer displayed on the screen instead.

> "Sometimes the interface had me simply scanning for bolded or color-coded items, when I would have preferred to be able to filter or sort to spot the items I was interested in." PID 4.

> "Found myself counting on my fingers for numbers not calculated by the system." PID 20.

> "Sometimes it seemed we had to count boxes rather than being presented a number. Might have been my inexperience." PID 23.

7.2.5 User Interface

Novice Users

This was the first time participants had used SourceVis before and only had a limited time. We expected there to be a learning curve for participants to understand how to use SourceVis.

> "Some of the links were not very obvious from only using the system briefly. I think after more use some of that would disappear as common tasks come out of the woodwork." PID 1.

> "The learning curve of the visualization is at odds with the ease of use of a touchscreen" PID 2.

> "Still some refinement needed to make the system more intuitive." PID 21.

> "Without more familiarity I would struggle to know which tool to use for some of the questions." PID 37.

Selecting an entity in most of the visualizations displayed some information about the entity in a menu, which some participants were not always aware of.

> "When selecting the classes on the right (e.g. system class evolution) I didn't release that the menus on the left were changing. I realized it then forgot again." PID 10.

This was a new touch UI for all participants and many had experience with other touch UI devices (§6.2.2). Given participants' touch UI experience they had some pre-conceptions on how the multi-touch user interface should work with respect to gesture navigation, menus, and selecting objects.

> "Popup menus behave differently to traditional touch screen devices where you click and hold to open the menu." PID 6.

Remembering Visual Encoding Representation

Some participants found it hard to remember what the encodings meant. In particular the Polymetric View encodings and types of classes (e.g. red border = interface). Some participants would have liked to seen more legends in the visualizations to help them remember the encoding representation.

> "Widths and heights or even the type of a box – is it a class or a package? Lack of an obvious legend for meaning of colours." PID 8.

> "Found the width (methods?) vs height (fields?) confusing." PID 24.

> "Many of the visualizations did not have a legend." PID 27.

Menus and Search

There were problems with a few of the menus on some visualizations. For example, the options menu had features that were off screen and could only be displayed if other options were collapsed. Some menus could accidentally be moved making the options in the menu unusable if moved off screen. These are programming errors which can be easily rectified.

> "The System Hotspots View menu on the left hand side runs off the edge of the visualization, even if the options missing are not useful, I don't know that until I've read and discounted them." PID 11.

> "In some instances the visualizations put menus offscreen, making the visualizations difficult to use." PID 27.

Most of the visualizations supported search, unfortunately we did not have enough time to include a search feature on all visualizations but it is something we would like to do in the future. For example we observed on one question (§H.1 Q12) all participants spending a large amount of time trying to find a class in the System Dependency Visualization using navigation gestures.

> "Search was not always available." PID 7.

> "Not all visualizations had search interfaces or the ability to search, which made it hard to find a class sometimes (for instance on the System Dependency visualization)." PID 4.

Some participants found that the search feature was not obvious how to start. It required selecting the search box region to open up the keyboard to enter text.

> "Some controls were not obvious at first such as searching required selecting certain attributes, but there was not a search button (instantaneous results instead)." PID 7.

Reading Text

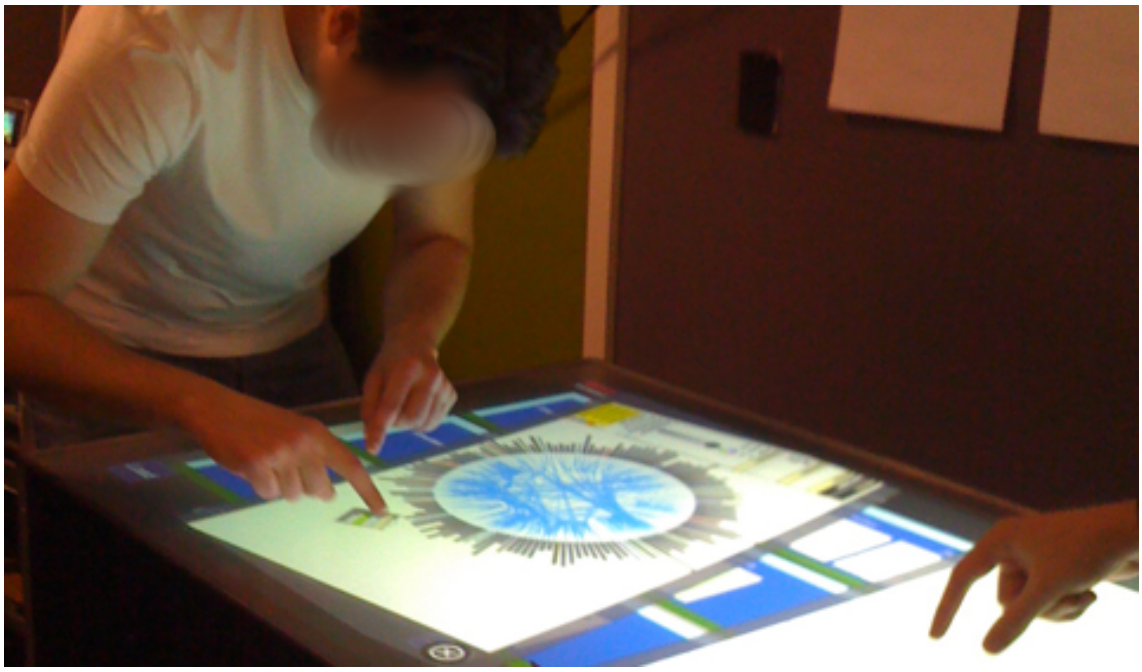
There were a number of issues with being able to read text on a screen. If text was displayed in windows, it was hard to read small text. Figure 7.15(a) shows a participant struggling to read text about entities with no dependencies and no references within a System Dependency Visualization inside a window. Figure 7.15(b) shows a participant doing a zoom gesture to read a class name in the Class Dependency Visualization also inside a window.

> "Most strings were too hard to read if a visualization not at full-screen." PID 23.

> "Some of the text was rendered too small by default." PID 27.

When a visualization was displayed at full screen most of the text was best read from the bottom side of the table (see Figure 6.3(a)). This sometimes made it difficult for participants when they were standing on the side ends of the table.

> "All the text is oriented to be read from one side which may make it trickier when there many people gather round all sides." PID 8.



(a) Struggling to read small text within the System Dependency Visualization.

(b) Struggling to read small text within the Class Dependency Visualization.

Figure 7.15: Reading text inside windows.

7.3 Improvements

In this section we discuss the suggested improvements of SourceVis and the multi-touch table as identified by participants, and through our observations (§6.1.5 Q3).

7.3.1 Multi-touch Table

Touch Experience

To make a better touch experience some participants suggested profiling the user touches to keep track of who is touching the table. We do not profile any touch points through instrumentation as it would have slowed the application down.

> "Profile the individual user touches." PID 3.

To increase the amount of people using the table and the size of the display some participants suggested spreading the application over multiple tables.

> "Maybe spread the application over multiple tables." PID 35.

Touch Precision and Accuracy

As most of the participants were familiar with mobile phone and tablet touch interfaces which are mainly capacitive designed they expected a similar accuracy with the touch table. Unfortunately the touch precision of our multi-touch table was not as accurate as these capacitive designed interfaces. We would like to explore SourceVis being displayed on a large commercial multi-touch capacitive surface to see if we can get better touch precision and accuracy.

> "I'd like more precise inputs." PID 4.
>
> "More consistent touch interactions." PID 5.
>
> "Could be more precise in touch detection." PID 41.

Screen Resolution

Most participants found that the low resolution impeded their progress and usability with SourceVis. Participants suggested increasing the resolution size, which would allow more information to be displayed, and allow for more windows to be displayed with less overlapping of information.

> "Nicer to operate at a higher screen resolution." PID 19.
>
> "Application really only work nicely when we worked on the same screen, mostly due to resolution." PID 23.
>
> "Too low resolution." PID 41.
>
> "Higher resolution would enable more simultaneous visualizations on screen." PID 42.

Hardware Performance

Many participants found that the hardware performed slowly. In order to get faster hardware performance for the touch interaction and visualization rendering it would be worthwhile increasing the hardware specifications. The computer we used in the multi-touch table ran both the touch detection and SourceVis. Separating the touch detection from the client application on separate machines may improve performance.

> "Stability and performance, was quite slow at times, especially during the individual section where control of the device sometimes did unexpected things." PID 1.

> "Only problem was the performance of the hardware was too slow." PID 19.

> "Sometimes it seemed a bit laggy." PID 29.

Participants wanted a faster frame rate from the camera to increase the UI responsiveness. The frame rate for capturing touch points was 60 FPS and the resolution input was 640x480 input. We could use a camera with greater resolution input and ability to capture faster FPS, which may yield better UI responsiveness.

> "Improve FPS and UI responsiveness." PID 3.

> "The table would need to be more responsive and stable." PID 12.

> "A more responsive system would be a huge benefit, and slightly smoother graphics would be good too I think." PID 17.

7.3.2 Visualizations

Visualization Context

A number of participants forgot which visualization they were viewing as many of the visualizations were similar in design. Adding titles, legends, and breadcrumb navigation to the visualizations could help participants to remember what visualization they were currently viewing. If a visualization had been opened previously on a version of a system then adding some kind of marker or annotation may help aware that they had previously seen that visualization and data before.

> "Add titles and a legend. A bit more structure might help, such as an obvious and consistent hierarchical key showing which part was being explored and to what level such as adding in breadcrumbs (e.g. codebase → version → package → class)." PID 8.

System Dependency Visualization

Participants struggled with the System Dependency Visualization, especially navigating and searching for a class. This visualization also did not render very large systems at all. The dependencies did not show edges direction of the dependency. Adding arrows at one of the edges when selecting a dependency might make the clarification of the direction of dependency clearer.

> "Dependencies could show direction of dependencies, and circular dependencies." PID 25.

> "Maybe put arrows on the ones with all the lines. That may make it clearer which is depending on which, because I found that confusing." PID 31.

Toxicity Chart

The Toxicity Chart allowed metrics to be turned on or off to update the chart. The threshold values associated for each metric were fixed and participants would have liked to be able to change these values within the visualization.

> "Ability to change thresholds on toxic chart." PID 25.

One of the individual questions in the user tasks (§H.3 Q4) required participants to look up different versions of a class to find out how many classes for that version were greater than a specific toxicity score value. All participants answered this question by using the System Version Menu except for one participant we observed who opened up a visualization for each version of the system and then inspected each visualization, discounting versions opened that did not meet the toxicity score value (see Figure 7.16). One improvement would be to iterate through the different versions rather than having to load a different version or multiple visualizations.

Additional Visualizations

We did not provide any visualizations that displayed source code or the algorithms that were employed. We purposely did not show any source code, as we were focused on providing high level architectural visualizations as opposed to low level artifacts. Displaying source code is something we would like to explore in the future. Many participants were interested in seeing code displayed within another visualization window, or being able to see code directly after seeing some form of representation of a class or a method. Perhaps a tap and hold gesture to display the source code might be a viable option.

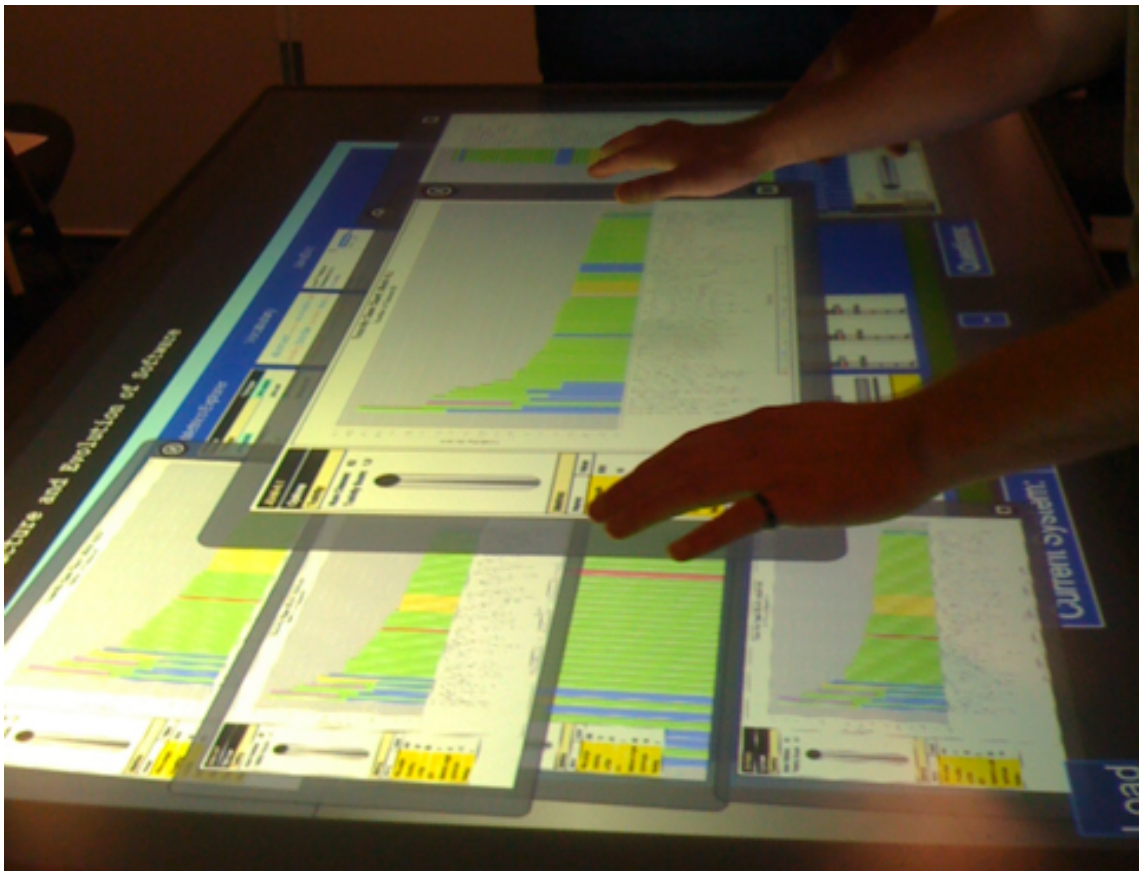


Figure 7.16: A participant opens multiple Toxicity Charts to answer a question.

> "In real life getting to see the code would be useful." PID 13.
>
> "It would be good to view code directly." PID 15.
>
> "The ability to zoom into the code or elements could be useful." PID 34.
>
> "It would be nice to see the source if it can be displayed." PID 38.
>
> "The visualizations don't help toward understanding how the code works, like how the cogs and flywheels of a machine spin. Or when the system is live which parts of the code is in use. I would like to be able to delve straight into the code, to visually inspect an item." PID 44.

Some participants would have liked to have seen inheritance information displayed, however, due to time constraints we did not implement these kind of visualizations. If we were to implement inheritance visualizations we would use a tree view or a modification of the System Complexity View which is part of the Polymetric Views [185].

> "A tree structure for usages of entities and showing an inheritance chart would be cool." PID 38.

During the post-study interview many participants suggested visualizing code contributions by developers from version control systems [59, 229]. We would like to explore this visualization technique in the future.

> "Ability to filter and group metrics by developer would be useful - could be used for identifying training needs." PID 25.

> "Focus on metrics as exploration tool, but more useful is usages and changeset navigation." PID 41.

Different Visualization Layouts and Linking

Some participants would like to have seen more graph representations of data in the visualizations. Only the System Dependency and Class Dependency visualizations support graph views.

> "More graph views that are persistent." PID 41.

Many participants found the circular layout in the System Class Dependency visualization not so good for large systems. We would like to experiment with different layouts for this visualization, such as tree views and tree maps.

> "A tree view for dependencies would be great." PID 36.

The pie menu linked overview visualizations and more detailed visualizations (see Figure 7.3(b)). Some participants would have liked more linking between overview visualizations.

> "More integration between visualizations. For example when in System Hotspots view, hold down on a hot spot and have an option to show a quick toxicity visualization for that item." PID 12.

7.3.3 Interaction

Navigation

Participants had problems when they both tried to perform navigation gestures on the canvas of visualizations at once. The navigation gestures could be improved by controlling who can zoom or a pan at any one time. Some participants suggested a full Zoomable User Interface (ZUI) [23, 24] might help to explore entities better.

> "Some sort of Zoomable User interface like Eagle Mode[1] might help to zoom into packages, classes, methods." PID 24.

[1] http://eaglemode.sourceforge.net/

Zoom Level

When some visualizations were started the camera view was at the same zoom level regardless of the size of the data. While other visualizations automatically adapted the camera view like the Dependency visualizations to a zoom level where all the data was visible but maybe unreadable (see Figure 7.15(a)). If the camera view was not adapted with respect to the size of the data it required the users to zoom to a level where they could view all the data. If the camera view was adapted then some text may have not been readable hence users would have to zoom to show the details they were interested in. Having navigation features that allow the user to adjust the zoom level without performing sometimes time consuming zoom gestures would offer more flexibility for users to control the zoom level. For example using a plus and minus sign, and a slider to indicate zoom level is common on many desktop map applications such as Google Maps may be an alternative.

> "In situations where there is lots of data, choosing a more pragmatic visualization zoom level as a default. This means that the visualization would be more responsive immediately, reducing user frustration." PID 6.

> "The tools I used today need some improvements in screen organisation: some elements appeared on screen in sizes not comfortable to use - too large or so tiny you couldn't control it. Some graph elements were grouped on screen in a way that you could not see all of them within the viewport, like a large line of squares, instead of line wrapping." PID 14.

Manipulating Windows

When working in the group condition we observed that once a visualization was initially displayed in a window most participants would instantly tap the full screen button on the border of a window to make the visualization go full screen (see Figure 7.17). The reason for doing so is that participants could gain the best view of the data. When opening a visualization it would be better if there was an option to display the visualization straight away at full screen rather than an intermediary step, or at least have another gesture such as tap and hold and select what size to display the visualization at.

> "Having to constantly maximize visualizations took time." PID 6.

> "Always wanted to maximize windows when working as a team, would be nice if this happened automatically." PID 24.

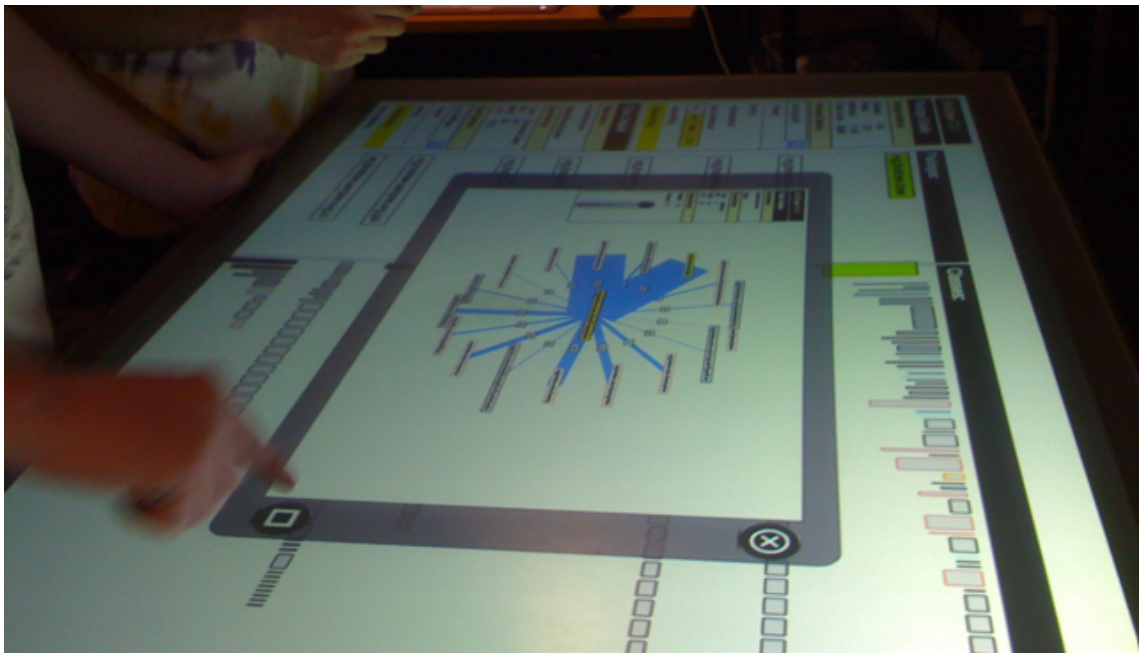


Figure 7.17: A participant immediately goes to display a new visualization at full screen by selecting the full screen button on the border of the window.

When a visualization was being displayed in a window the way to adjust the size of the window is to use two finger gestures on the border to increase or decrease the size of the window. Some participants found it hard to resize and position windows. We observed some participants moving windows with two hands instead of one finger drag on the window borders which was very cumbersome (see Figure 7.10). Adding more options to resize and move a window would give more flexibility to manipulate windows.

> "Maybe a more precise ability to position windows would be useful." PID 4.
>
> "Resizing windows is difficult at times." PID 15.
>
> "Management of views." PID 41.

When working as individuals, participants suggested it would be useful to have options to snap the windows to certain areas of the screen such as taking up half the screen and oriented to specific sides of the table. Alternatively the windows could be prevented from overlapping one another by providing collision detection techniques so that when one window is about to overlap another it would stop and bounce off each other. Nonetheless we need better support for the layout of multiple windows on the table at once. Other researchers have explored personal workspaces on tabletops [291].

"Snapping windows to particular sizes (e.g. half screen)." PID 5.

"Multi-visualization needs to be addressed so common situations like two visualizations being used is handed out-of-box (opening, dragging and zooming should be done for you)." PID 6.

"Have a window to snap to side of screen so that the optimum amount of screen can be used when two people are working on it." PID 21.

"There were some issues when using the table to do two different tasks, such as windows expanding, or overlaying each others, perhaps a working region could help reduce this." PID 44.

Participants found performing the windows close gesture when visualizations were at full screen problematic due to the inaccuracy of the touch detection in the corners of the table (see Figure 7.13(a)). Having an alternative close option or gesture could improve the usability.

"Found I wanted to drag and release when closing windows." PID 24.

7.3.4 Data

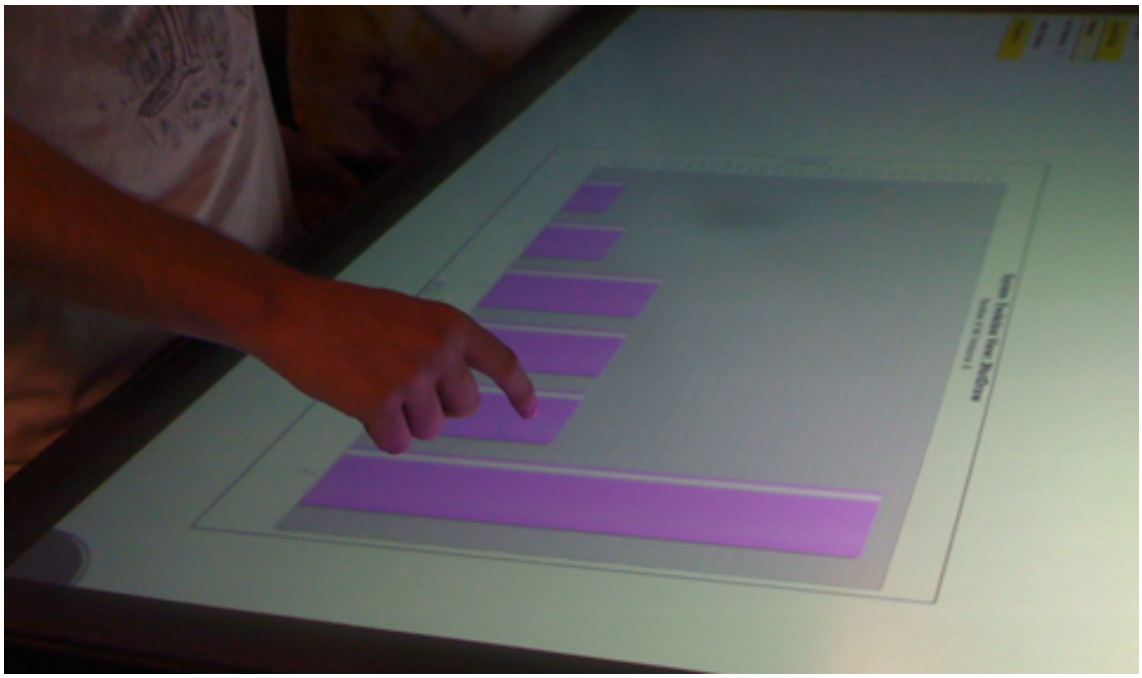
Data Manipulation

Participants liked the options to manipulate data through sliders, filters, sorting, and grouping. Some participants would have liked more data manipulation options, but they did not give very specific details.

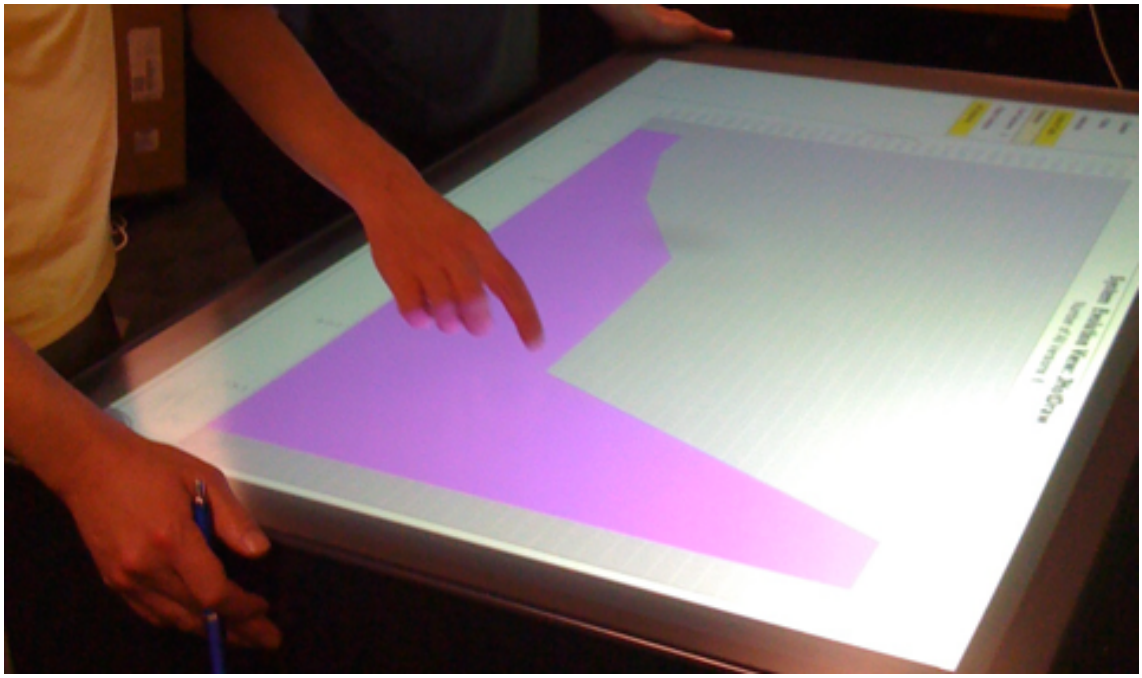
"More filtering." PID 12.

"More useful filters and group functionality tools." PID 28.

For example in the System Evolution Visualization one of the group questions (§H.1 Q9) asked pairs to find out how many lines of code one version had increased over another (see Figure 7.18). We observed most participants answering this question by deselecting all metrics except lines of code, which displayed the versions and one series of data. As there was a large difference between the two data values, participants tried to read the axis labels on the left hand side of the chart and then discussed with their colleague about what the answer was. The chart was hard to read as the axis labels were on the far left of the visualization and the values on the far right, which caused some confusion. We observed participants thinking aloud and all struggled to calculate an accurate number on the fly. Some participants used a bar chart (see Figure 7.18(a)) while others used an area chart (see Figure 7.18(b)). At the time, many participants suggested that SourceVis should support comparing multiple data directly in the chart and showing the differences visually.



(a) Bar chart.



(b) Area chart.

Figure 7.18: Comparing data using the System Evolution Visualization to find out which version has the most lines of code.

Metrics Data

We provided metrics information for systems, packages, and classes. Some visualizations like the Vocabulary Visualization showed how many entities were

displayed, however, participants would have liked more summary information.

> "The Metrics Explorer Visualization needs more summary info (e.g. count of what is currently displayed/selected). It was the only place I had to manually count items on the visualization to answer a question." PID 11.

The metrics data predominantly showed size and summary information about entities and were good for finding outliers and data trends. Some participants would have liked detailed information to be able to answer more statistical questions.

> "Max and average statistics for classes would be useful. Worked well for the questions we were asked. Other types of questions like finding average method length, or all methods over say 50 lines of code might be harder to answer." PID 25.

The visualizations were primarily based on software metrics. Some participants suggested that the metrics were not that useful and did not show anything about design decisions. In the future we would like to explore visualizations that show design decision information on top of metrics such as disharmony maps [187, 361].

> "Metric visualization tools don't know about design decisions and would alarm on things that have been designed that way intentionally. As an example a circular reference is seen as a bad thing in design, but when coding a relational data model in Java you can hardly avoid them." PID 14.

> "I'm not sure the selected metrics are always that useful. LOC is a sign of code smell I guess." PID 38.

Saving and Note Taking

SourceVis does not support annotating data within a visualization, making notes on a separate window, tagging important elements within a visualization, or saving any information. Some participants would have liked to be able to save the current view. All of these kind of features would help with the user experience and improve the utility of SourceVis especially if developers were to use the application in a work place environment. These features would certainly help sharing information with other members of the team at a later date once information was discovered.

> "Add manual notes. Allow manual tagging of important classes." PID 20.

Integration With Other Tools

We developed SourceVis as a standalone application and used systems from the Qualitas Corpus (Version 20101126) [325]. Many of the participants suggested integrating SourceVis with IDEs, version control systems (e.g. SVN, GitHub, Google Code), and other software development tools.

> "Produce plugins for popular systems like Hudson or Jenkins. Connect to a live code base." PID 19.

> "Linking into a single live system image." PID 42.

Some participants suggested performing automatic refactoring of the software within the visualizations. If SourceVis was integrated with IDEs then being able to perform drag and drop refactorings is an avenue to explore [190].

> "What is the highest priority question developers should ask of their code, and how do you actually decide which code to fix/review? Could this be done in the application by automatic refactoring?" PID 26.

Some participants use modelling tools for software development and maintenance. Some participants suggested integrating UML diagrams and tools with SourceVis might be useful.

> "I can also see how these visualizations could be blended with an IDE and UML (e.g. drawing tool)." PID 26.

Some participants would have liked to be able to share their discoveries using collaborative social computing systems and to generate reports. Others have explored social media artifacts in collaborative software development [335].

> "No collaboration with social networking systems or other applications like Google Talk or Microsoft Communicator. Provide the ability to connect data with other collaboration software like Google+, Twitter. What if I wanted to email some stats to someone?" PID 19.

7.3.5 User Interface

Interface Learning and Help Information

Some participants would have liked to have more time to explore SourceVis. Given the study was controlled in a lab setting with finite time, this limited the amount of time participants could spend learning about the interface.

> "I really see the usefulness but would need more time" PID 35.

> "Some features were somewhat hidden or not obvious to me in the brief session." PID 37.

Participants would have liked more help in certain places to learn the interface faster. We provided some help documentation about the visualizations and how to use the visualizations in certain places. A tap and hold gesture on icons in the startup screen displayed additional information about a visualization. Figure 7.19(a) shows help information in a dialog box about the Toxicity Chart Visualization displayed from the startup screen. A tap and hold gesture on metrics labels and layers in the visualizations displayed additional information about a metric or layer. Figure 7.19(b) shows help information in a dialog box about the accessor layer in the Class Blueprint Visualization.

> "More contextual help without popping up dialogs (e.g. information on touch that disappears when the touch is released, possibly on the background so it doesn't interfere with touch actions, for example something similar to IDE autocomplete suggestions." PID 5.

> "The improvement I can think of at the moment is making it more obvious when I clicked the classes that the metrics on the left changed. Once you have played with with this for an hour - you know this anyway. Working with two people didn't even notice this because both looking at different bits of screen so things changing seemed to be more noticeable." PID 10.

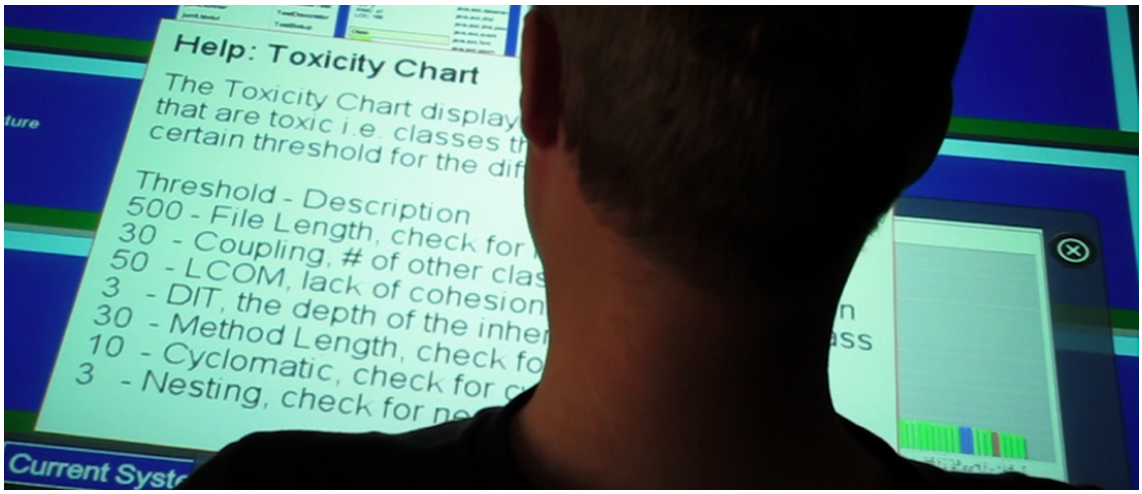
> "I think there could be more information in more places." PID 16.

> "Some way of calling out features to the user, making suggestions perhaps. I would probably be a lot more effective just with a bit more familiarity. Kind of like learning an IDE." PID 37.

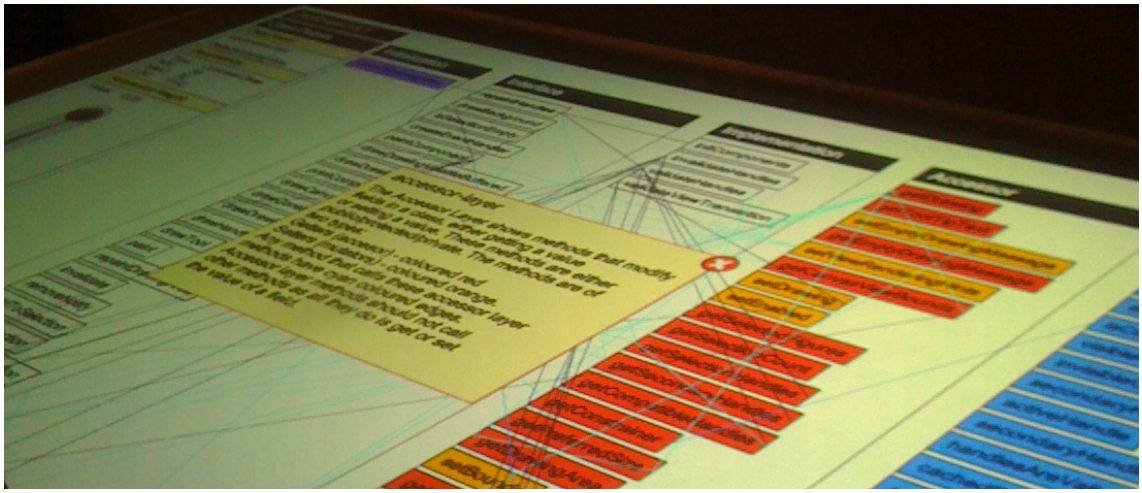
As this is a new interface for participants they suggested more visual feedback when touching objects on the screen. We provided some visual feedback for when a user touches the surface, an object, or button. When performing the tap gesture on the surface a blue circle trace is displayed beneath the user's finger (see Figure 7.20(a)). When performing a tap and hold gesture on an entity the blue circle trace gradually becomes red while holding. Once the circle is completely red the gesture is ended and the action is then executed for the gesture (see Figure 7.20(b)).

> "Everything you touch gives some information." PID 16.

> "More visible feedback when buttons are pressed" PID 27.



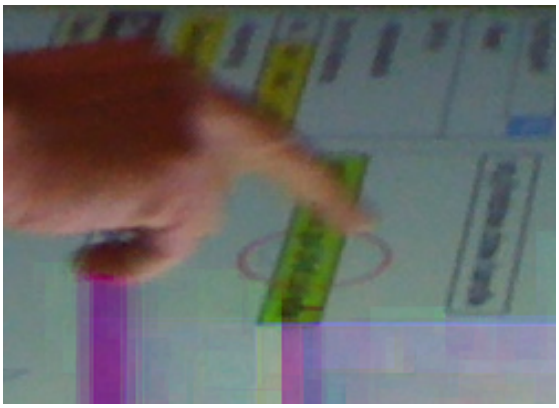
(a) Toxicity Chart Visualization help information displayed from the Startup Screen.



(b) Class Blueprint help information about the accessor layer.

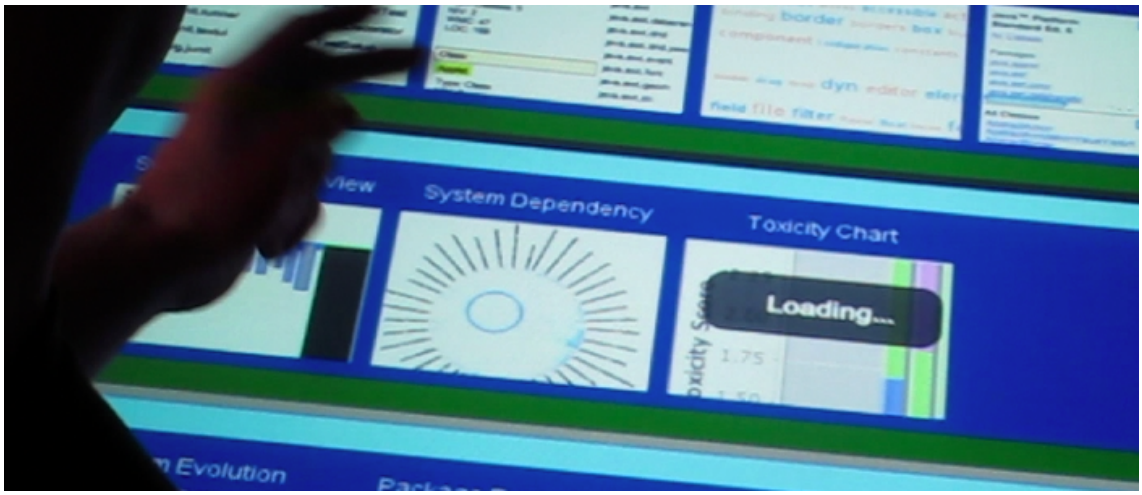
Figure 7.19: Help Information.

(a) Tap gesture, displays blue circle.

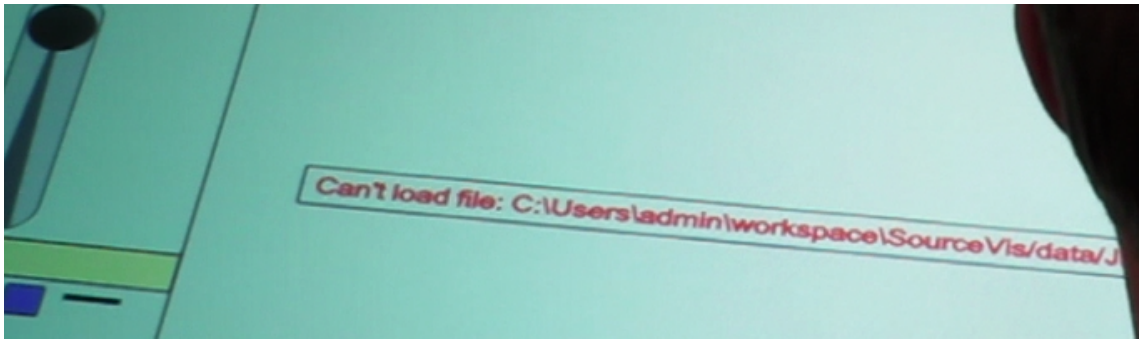


(b) Tap and hold gesture, displays red circle.

Figure 7.20: Visual feedback on touch gestures.



(a) Notification Statement about loading a new System Dependency Visualization.



(b) Error Statement about trying to load a data file in the Class Blueprint Visualization.

Figure 7.21: Visual Notification and Error Statements.

Participants would have liked more visual notification and error statements of incorrect touches and data for user actions. We provided notifications for loading a new visualization, error statements for displaying a new visualization from the startup screen without loading a system first, error statements for displaying a visualization that does not have an associated data file, and error statements when doing an ascending or descending search without selecting a metric. Figure 7.21(a) shows a popup "Loading" box indicating that a new visualization is starting. Figure 7.21(b) shows an error message when a participant has tried to load a Class Blueprint Visualization on a class that does not have an associated data file.

> "When you do something wrong you get an obvious indication of how to correct it (or what to do to make it work)." PID 7.

Remembering Visual Encoding Representation

We adopted Polymetric View [185] encodings to represent packages and classes. As participants were switching between visualizations that contain packages and classes, both using this encoding scheme, they got confused as to what kind of entity they were looking at. The encoding could be augmented with additional markers or icons to signify the difference between these entities.

> "Possibly colour coding or icons to differentiate entities." PID 24.

> "Packages could have icons next to them, quickly allowing the user to identify the difference between classes." PID 27.

> "Although the visualization is specific to one type, they look exactly the same. Some colouring or iconography might help." PID 36.

Menus and Search

The menus showed a lot of information and some participants found the grouping of elements were not well defined. There may have been a better way to style the different components within the menus and to use different grouping options.

> "Slightly clearer labels on the left menu, with more obviously defined breaks between sections of sorting/filtering options." PID 39.

The left hand side menus were in a fixed position. Some participants would have liked for these menus to be more flexible when navigating in a visualization. Some suggested having the menus follow them around the visualization or be able to display them on demand.

> "Some parts of the interface (menus) could reflow when zooming." PID 9.

The pie menu and keyboard are displayed at the current zoom level of the camera rather than relative to the window (see Figures 7.3(b) and 7.9(a)). This caused some issues for participants as the pie menu and keyboard appeared off screen or too small. To improve the usability it would make sense for them to be consistently the same each time they were displayed.

> "The keyboard and pie control menu could spawn at an initial size relative to the window, rather than relative to the item being clicked on making it easier to perform actions at any size would reduce the amount of steps required to complete a task." PID 40.

Some participants were not sure exactly what would be displayed when they selected one of the visualizations from the pie menu and suggested using icons instead of text to represent the visualization type.

> "Found it a bit difficult to decide which options to choose from the pie menu - maybe an iconic representation of which screen I would go to for each option would help." PID 24.

Some participants struggled to find the search box feature to search for entities. For the search box in the Metrics Explorer Visualization we used a search label 'Search for:', a box with an example ('e.g. TestCase'), and a magnifying glass icon which looks similar to the Google search box icon. While for the System Hotspots View, System Package Evolution, and System Class Evolution visualizations had no search label but included a 'Reset' button underneath the search box. Figure 7.22 shows the search box in the Metrics Explorer where a participant has issued a search for class names that include the word 'test'.

> "Improve search boxes so it is clear where to search." PID 7.

The search button on the keyboard was a separate key labelled 'Search' as opposed to pressing the 'Enter' key to execute the search query (see Figure 7.22). This caused participants some confusion as they would use the 'Enter' key to perform the search with unexpected results. It would be possible to modify the keyboard to make a search work on the 'Enter' key.

> "The enter key on the keyboard should run a text search, rather than insert a newline character." PID 27.

Reading Text

For all participants reading text when displayed in small windows was difficult (see Figure 7.15). Some participants suggested using a magnifying glass lens to help with reading. There may have been an option in MT4j to fix this but we were not aware of any at the time. Reading text when a window was at full screen was not a problem.

> "Size of the text, perhaps use a localized magnifying glass." PID 13.

> "I know its an MT4j issue, but it is hard to read text when the windows are too small. This makes the collaboration mode very hard because even with a window at half size, the font is too blurry." PID 22.

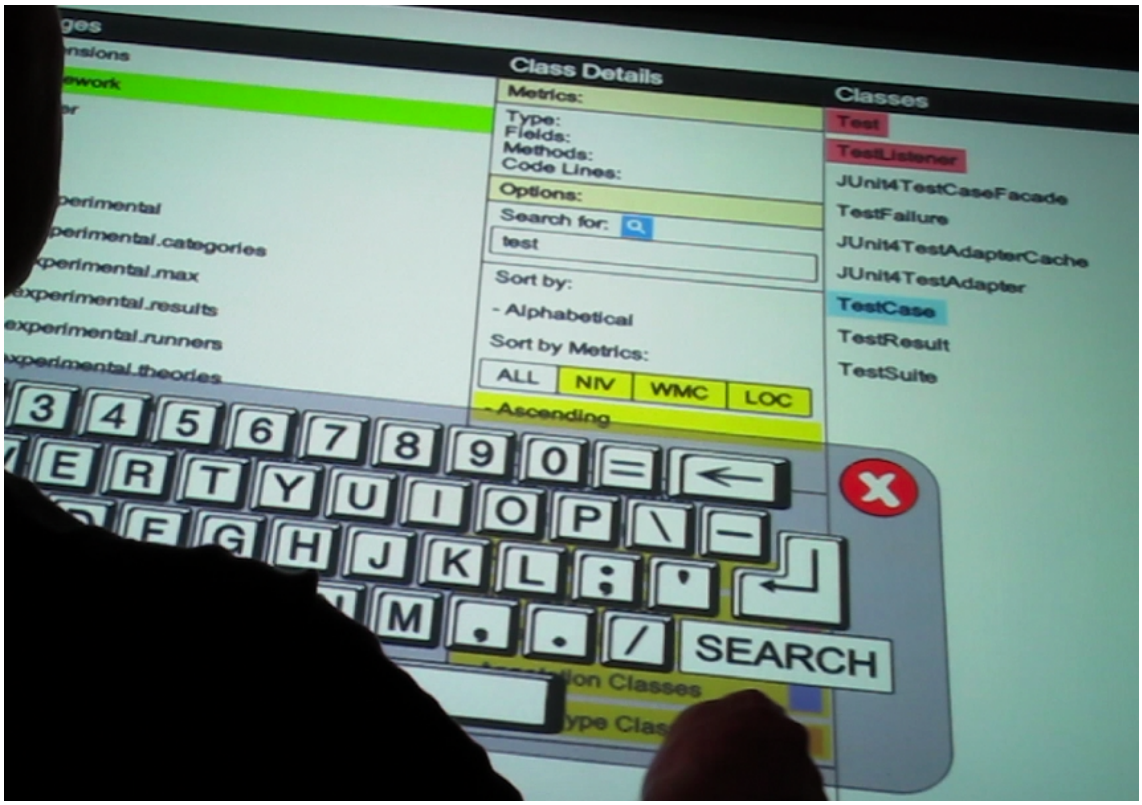


Figure 7.22: Search Box and Search Button on keyboard on the Metrics Explorer Visualization.

UI Consistency

Many participants would have liked a more polished user interface but few offered any specific details for improvement. Given SourceVis is a prototype we would like to work with a UI designer in the future to improve the style and consistency.

> "UI adjustments." PID 2.
>
> "More polished UI." PID 3.
>
> "Minor UI improvements could be seen all over the place." PID 42.

Participants would have liked greater consistency of UI elements and interaction gestures across all the visualizations. For example some objects could be dragged around the screen while on another visualization they could not be. There was some inconsistency with what could be touched and what could be dragged.

> "Consistent interaction with controls such as zooming and moving would also go a long way to reducing frustration." PID 6.
>
> "Need to make it easier to move objects on the screens for example when I select something place that in the middle of screen and resize

> to emphasis the components it relates to instead of making us search them out." PID 19.
>
> "Make it obvious which parts of the interface are selectable. The UI was flat that it was not always clear which parts were selectable." PID 23.
>
> "Consistency across all interfaces for interaction (e.g. click/drag, click-/click)." PID 27.

Some participants felt there was too much text in the UI. For some of the widgets participants would have preferred icons instead of text to denote actions. For the sorting options we could have used "A–Z" instead of "Alphabetical", likewise something similar for ascending and descending.

> "Use standard form components where possible (e.g. buttons/sliders rather than text labels)." PID 27.
>
> "Some of the UI controls for filtering / controlling visualizations feel a bit text-heavy for a visual tool." PID 32.

Startup Screen

If a visualization was displayed at full screen participants found it annoying to navigate back to the startup screen to create a new overview visualization. Having a gesture or menu which would allow displaying new main overview visualizations from an existing visualization would be helpful. Alternatively, providing a gesture to navigate back to the start screen would increase the usability.

> "The startup screen could be a menu brought up by a gesture or something - having to go back to it felt more difficult than on my more usual interfaces." PID 9.
>
> "There could be a way to quickly navigate to start screen." PID 27.

Some participants were not aware that the visualization icons within the categories in the startup screen could be rotated. Figure 7.23 shows one participant who has rotated two of the three categories (Exploration and Structure) on the start screen to point to his direction, denoted by the green borders of the categories pointing to the participant at the top of the Figure, while the other category points to the bottom of the table. Some participants suggested making the icons more flexible so that they can move around the startup screen.

> "Move icons around in the startup screen or at least move them to the other side." PID 21.

Some participants who were aware of the visualization rotating icons on the startup screen suggested that there should should be options on the borders of windows that contain visualizations to rotate windows in a similar fashion.

> "Direction change buttons similar to the start screen would be useful on opened windows for collaboration." PID 22.

Figure 7.23: Rotating visualization icons on the startup screen.

7.4 Team Collaboration

In this section we discuss how the multi-touch table helped with team collaboration (§6.1.5 Q4).

7.4.1 Multi-touch Interaction

The table supported multi-touch interaction which allowed more than one participant to interact at once and they could do this on the same visualization or have their own separate windows. We observed that having multi-touch interaction meant that pairs did not have to rely on their partner to control the interface, instead one participant could interact when they wanted to.

> "Having the table respond to input from multiple people at once – can potentially have sub-windows open in separate parts with different controllers." PID 8.

> "One of the benefits found is that people working together can share the interaction in a session. This is easier than when having a review session as a team and one person pilots and others have to command him to click things they can see on a projection wall. With this tool this saves time." PID 14.

Even though both participants could interact with the multi-touch table at the same time some participants felt that the multi-touch interaction detection support hindered their touch experience especially when there was a visualization at full screen. The multi-touch detection was less of a problem when participants were working on different visualizations in separate windows in parallel.

> "The fact that two people couldn't touch the screen at the same time was a major difficulty. Especially since the table is so large, and supports multiple windows, you forget about this limitation and want to work simultaneously on multiple windows as if they were separate iPads." PID 4.

While other participants found the touch interaction was not a problem which allowed them to work together, and promoted shared and casual interactions.

> "The table worked great with many touches and two of us being able to work on our problems easily." PID 7.

> "Promoted shared interaction and negotiation. PID 41.

> "The environment of a table felt very comfortable and natural to start casual interactions." PID 44.

If the touch experience was better then some participants felt that it could be better than existing practices (i.e. desktop computer with mouse, keyboard, and monitor). We would like to conduct a between subjects study to see if there is any difference between these two interface styles.

> "I really didn't see anything inherent to the touchscreen that made it better for team collaboration with these particular widgets, than a normal monitor and interface. But, I can see how, especially if it supported simultaneous input by multiple users better, it could be useful." PID 4.

7.4.2 Team Work

Participants found that the multi-touch table encouraged them to work together and collaborate with each other.

"The table helped a lot with team collaboration." PID 6.

" As a team building exercise, a discussion could be had over particular parts of the system. PID 28.

"It's great for collaboration to answer the same questions." PID 37.

"Easy to collaborate with a team mate." PID 43.

The multi-touch table allowed participants to work together closely (e.g. group condition) and loosely (e.g. individual condition). For example they could both be working on the interface but looking at different aspects of a system and then come together, share their discoveries, and share ideas with each other.

"Having the personal space to be able to do something on the display, talk to my teammate, and show things to them without being crowded was excellent. This is not the case when showing something on a standard workstation monitor." PID 11.

"Showing other people something is very easy. I imagine that this can also be very fast to use when it's more advanced (more options and filters) and the user has had enough time to become fully familiar with it." PID 12.

"Everyone seeing the same data was useful." PID 28.

"The ability to have people to bounce ideas off a colleague was very useful." PID 34.

7.4.3 Communication

Participants felt that the multi-touch table encouraged them to communicate with each other.

"Encourages discussion and participation." PID 3.

"Helped stimulate discussion." PID 5.

"Encouraged discussion and interaction." PID 23.

"Improved the relationship between the developers as we were both communicating with each together." PID 27.

"The table encourages communication." PID 38.

"The table was a focus for conversation. It was a common space for conversation about the software." PID 41.

When using the table, we asked participants to think aloud. We observed that they would communicate with their colleague to inform them what they were doing with the interface. If the participants were more proficient with the interface it would be interesting to see if there would be the same amount of communication.

> "Allows users to explain properly to each other, what they are attempting to achieve." PID 2.

7.4.4 Different Roles

We observed that the multi-touch table afforded different roles for the participants. Participants could easily switch between the roles and do different roles at the same time.

> "Having a multi-touch table meant that there was no waiting for the teammate to finish doing something." PID 40.

> "The two of us could operate different parts of the interface, taking up different roles." PID 44

With most of our visualizations (except the Package and Class Evolution) the options menus were located on the left hand side. Quite often during group work, this led to the person standing on the left hand side of the table controlling the interface and menus, while the person on the right hand side usually held the question sheet and performed the close, maximize, and minimize options as those controls were generally on that side of the visualizations.

> "Having the menus all on the left allows the user on the left to lead the operation of the software." PID 2.

7.4.5 Coordination

Given that the multi-touch table supported multiple users interacting at the same time, we observed that one participant mainly controlled the interface at any given time during the group condition.

> "Really only one person can be using the visualizations at a time." PID 6.

> "We found a way of working quite quickly where one person would do the driving." PID 16.

> "I think one driver would be a good idea." PID 38.

We observed even though there was one participant controlling the interface most of the time, participants regularly coordinated with their fellow colleague to take turns controlling. The large table size and multi-touch interaction made it a seamless process for participants to swap roles from controlling to observing.

> "There were a few interesting etiquette issues to get used to - who's driving?" PID 24.
>
> "The table has a big screen that we could both interact with. It meant that when you wanted something to happen you could just do it without feeling like you were taking over - which is what working on a single user interface is like." PID 31.
>
> "Even though it is multi-touch, we often took turns to use the table." PID 33.
>
> "Felt a bit odd swapping who drives what, but that's just practice." PID 42.

Participants liked that they did not have to swap physical positions in order to control the the interface.

> "Working together at a moments notice with no need to move to another machine or move to a whiteboard." PID 39.

7.4.6 Awareness

Participants felt the multi-touch table made them more aware of what the other was doing while interacting and navigating within the visualizations, as they were both using the same interface and were within close proximity of each other.

> "Good for seeing what other team member was thinking about when they were interacting and talking." PID 5.
>
> "Because it's a touch interface you can more easily see what the other person is doing with the interface - it's easier to track an arm than a mouse pointer." PID 17.
>
> "Each person can see how the other is navigating." PID 18.

7.5 Summary

In this chapter we presented the *qualitative* findings from the user study involving professional software developers (§6). Table 7.1 summarizes the key findings based on participants' feedback in the post-study questionnaire (§6.3.4), video recordings of the participants, spoken aloud thoughts by the participants, and observations and notes made by us during the study. The next chapter presents the *quantitative* results from the professional user study (§8).

7.5.1 Multi-touch Table

Participants preferred working as a group with the multi-touch table as it seemed to be easier and more effective compared with working as individuals. Participants liked that the table supported multi-touch, multiple users, and the size allowed multi-user interaction and viewing large amounts of data (§6.1.5 Q1). Participants found doing individual work at the same time on the the table difficult. The table suffered from an inconsistent touch experience, precision and accuracy problems with the touch detection, a low screen resolution, and hardware performance issues. Some participants felt uncomfortable after standing for awhile using the table and that prolonged use could be a problem without something to lean or sit on (§6.1.5 Q2). Some participants suggested using a commercial multi-touch table which may offer a better touch experience, better touch precision and accuracy, larger screen resolution, and faster hardware performance (§6.1.5 Q3).

7.5.2 Strengths

There were a number of **strengths** of SourceVis (§6.1.5 Q1).

- *Visualizations:* Participants liked how SourceVis supported multiple visualizations simultaneously. The evolution and chart visualizations were popular.
- *Interaction:* Participants liked the navigation touch gestures for zooming and panning, and the ability to switch between visualizations through windows and the pop up pie menu.
- *Data:* Participants liked being able to display overviews of systems, provide details on demand, manipulate data in multiple ways, change the system version in a visualization, identify outliers, and discover trends.
- *User Interface:* Participants found the user interface to be intuitive, consistent, and the visual encoding representation (e.g. class type and Polymetric View encoding) helped with identifying interesting aspects in the visualizations.

7.5.3 Weaknesses

There were some **weaknesses** of SourceVis (§6.1.5 Q2).

- *Visualizations:* Participants sometimes forgot what visualization technique they were looking at. With visualizations that utilized large amounts of screen space participants were confused as to where they were in a visualization and found it difficult to keep track of what they had seen previously. The System Dependency Visualization was very hard to use and participants struggled with this technique.

- *Interaction:* The navigation (panning and zooming in a visualization) became problematic when two people were both performing one of these navigation gestures at once. There were some inconsistencies of items such as the pie menu and keyboard being displayed at different sizes depending on what the current zoom level was. Manipulating windows to face a direction using two handed gestures was cumbersome and closing full screen windows was problematic due to the touch inaccuracies in the corner of the table. Entering text through the virtual keyboard was difficult for participants. Some participants would have liked to been able to enter numbers through a keyboard rather than use widgets like sliders to manipulate the data. Occasionally users would interfere with each other when interacting, such as taking visualizations off each other. Participants reaching over one another was problematic during the individual condition.

- *Data:* Some participants felt there was too much information being presented at once. Participants did not like having to find information by visually searching (e.g. counting or looking for coloured entities) and would have preferred SourceVis give a precise answer for some of the questions.

- *User Interface:* This was the first time participants had used SourceVis and a large multi-touch table but they quickly learnt how to use the interface. Some participants occasionally forgot what the visual encoding representation meant for the size of entities that used the Polymetric View encodings and what colours were represented by which class type. Some menus were initially displayed offscreen and could accidentally be moved. The search options were not included on all visualizations and not obvious how to start. Text was hard to read if it was displayed in a window and best read when a visualization was displayed at full screen.

7.5.4 Improvements

Participants suggested a number of ways to **improve** SourceVis (§6.1.5 Q3).

- *Visualizations:* Add titles, legends, and breadcrumb navigation to help participants remember the different visualization types. Improve the System Dependency Visualization by having directed edges and a search feature. Improve the Toxicity Chart by including different metrics, ability to adjust threshold values, ability to selecting individual data points, and increase the rendering speed. New visualizations such as displaying source code, inheritance information, and code contributions by developers were suggested. Add new visualization layouts like treemaps and graphs, and link between and synchronize visualizations.

- *Interaction:* Provide control over who can zoom and pan in a visualization at once which would cause less frustration. Add features to control the zoom level rather than relying solely on zoom and pan gestures. When visualizations start, have options to display immediately at full screen or in a set sized window. Provide better options for manipulating windows and closing visualizations at full screen.

- *Data:* Provide additional filters and other widgets to manipulate data. Add more options for summarizing and displaying statistical data about metrics. Integrate other software design aspects on top of the metrics which could make understanding the problems with a system clearer. Add features to save the current view, make notes, and generate reports to inform team members about their discoveries. Integrate SourceVis with IDEs, version control systems, and other tools to provide a more complete application.

- *User Interface:* Add more help documentation, tooltips, and visual feedback. To help users remember the visual encoding representation (e.g. class type and Polymetric View encoding) add legends to the visualizations. Make the options menu more flexible so it can be displayed from anywhere. Pie menus and the keyboard need to be displayed relative to the window and not the camera view port. The search features need improving by resetting the data set each time for a new search query, a clearer location for the search box, and the keyboard adjusted so "enter" starts a search. Text needs to be displayed more clearly to improve reading, especially when text is displayed inside a window. Improve the UI by being more consistent with the style, icons, and interaction capabilities. The visualizations from the startup screen should be able to be started from anywhere in SourceVis.

7.5.5 Team Collaboration

We asked participants how the multi-touch table helped with team collaboration (§6.1.5 Q4). The multi-touch allowed multiple users to interact at once rather than relying on a single user to control the interface. The table encouraged participants to work together, helped improve relationships, made sharing information easy, allowed users to work closely or loosely, and communicate with each other. The table allowed participants to perform different roles when interacting. We observed that most of the time one person was controlling the interface. It was a seamless process for participants to coordinate swapping roles. Working together as a group made participants more aware of what each other was doing as opposed to working separately as individuals.

The next chapter (§8) presents the quantitative findings from the professional user study.

Table 7.1: Summary of qualitative findings based on participants' feedback in the post-study questionnaire, video recordings of the participants, spoken aloud thoughts by the participants, and observations and notes made by us during the study.

	Strengths	**Weaknesses**	**Suggested Improvements**	**Team Collaboration**
Multi-touch Table	Group Work Multi-touch Multiple Users Table Size	Individual Work Touch Experience Touch Precision and Accuracy Screen Resolution Hardware Performance Prolonged Use	Touch Experience Touch Precision and Accuracy Screen Resolution Hardware Performance	Multi-touch Interaction Team Work Communication Different Roles Coordination Awareness
Visualizations	Multiple Visualizations Evolution Visualizations Chart Visualizations	Visualization Context System Dependency Visualization	Visualization Context System Dependency Visualization Toxicity Chart Additional Visualizations Different Layouts and Linking	
Interaction	Navigation Visualization Switching	Navigation Zoom Level Manipulating Windows Entering Text and Data User Interference	Navigation Zoom Level Manipulating Windows	
Data	Data Overview Details on Demand Data Manipulation Data Switching Data Outliers and Trends	Information Overload Finding Information	Data Manipulation Metrics Data Saving and Note Taking Integration with Other Tools	
User Interface	Intuitive User Interface Visual Encoding Representation UI Consistency	Novice Users Remembering Visual Encoding Menus and Search Reading Text	Interface Learning and Help Remembering Visual Encoding Menus and Search Reading Text UI Consistency Startup Screen	

Chapter 8

Professional User Study — Quantitative Findings

Contents

In this chapter we present the *quantitative* findings of the professional user study (§6). We show which study condition combination the participants selected. We show which kind of collaborative coupling categories and styles the participants used. We show which physical arrangement participants favoured when using the multi-touch table. We show the perceived effectiveness of the visualization techniques by the participants.

8.1 Study Condition Combination

Q5 - Which study condition (e.g. Group or Individual) did the pairs favour for Section 3 of the user tasks? (§6.1.1)

For Section 1 of the user tasks all pairs were assigned either the "Group" or "Individual" condition. For Section 2 they worked in the opposite condition from Section 1. Odd numbered pairs started in Group and even numbered pairs started in Individual. For Section 3 each pair had a *choice* of condition. All of the 22 pairs chose the "Group" condition. The pairs were evenly split (11 pairs each) between the combinations GIG or IGG. As in Table 6.3, pairs in the GIG combination are shaded grey while those in IGG are not shaded. Individual was not selected for Section 3 hence neither IGI nor GII combinations eventuated. Participants felt that it was easier and faster to work in a group compared to working as individuals in parallel, hence all pairs selected "Group" for Section 3 of the user tasks. This decision may have been caused by the limitations of the hardware with respect to multi-touch detection and many participants commented in the post-interview that they thought they were partaking in a collaborative group study hence why they selected group.

8.2 Collaborative Coupling Categories

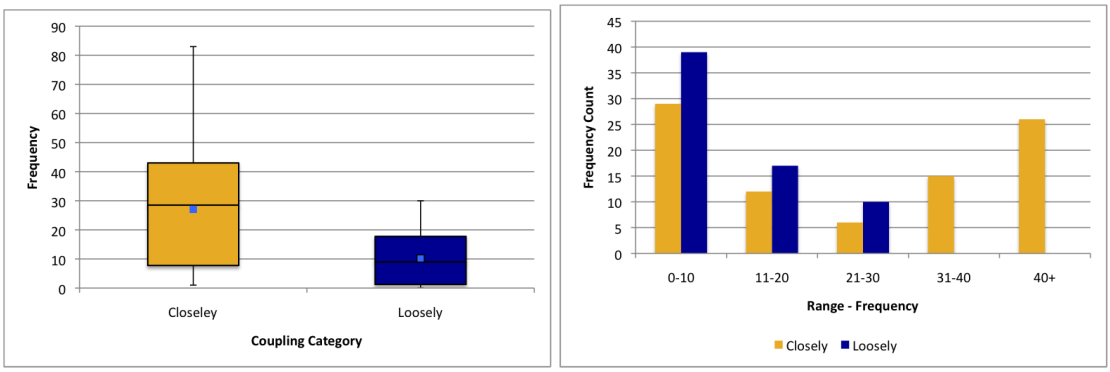
Q6 - Which coupling categories did the participants use? (§6.1.2)

We wanted to know which collaborative coupling categories the participants used, how much time was spent in each category, and what sequence the participants performed the different categories. The coupling categories were closely coupled and loosely coupled and are groupings of the coupling styles described earlier (see Table 6.1). The closely coupled category comprises the coupling styles: Discussion (DISC), Same Problem Same Area (SPSA), Same Problem Different Area (SPDA), and Viewing Engaged (VE). The loosely coupled category comprises the styles: Viewing Disengaged (VD), Different Problem Same Area (DPSA), and Different Problem Different Area (DPDA).

8.2.1 Frequency of Coupling Categories

Figure 8.1(a) shows a boxplot of the frequencies of the coupling categories. 78% of the interaction was closely coupled while 22% were loosely coupled. There were more close collaboration coupling styles (and more variation of styles) used than loose collaboration styles. This would be expected since the pairs conducted more sections in the Group condition than the Individual condition. These findings imply that pairs frequently switched between coupling styles when closely coupled. When pairs were loosely coupled (predominantly when they were in the Individual condition) they tended not to switch to another coupling style.

Figure 8.1(b) shows an histogram of the frequencies of the coupling categories. The X axis shows the bin range of frequencies and Y axis frequency count. The closely coupled category occurs more frequently in the 31+ range. The loosely coupled category occurs more frequently in the less than 30 range. This finding shows that there were more changes in closely than loosely coupled styles. We used R [252], and ran the Wilcoxon rank sum test at 95% confidence to compare if there were any statistical significant differences between the frequency of the coupling categories. We found there were a significant difference between the frequency of the categories (W = 4342, p = 1.506×10^{-07}).



(a) Boxplot of Frequency of Closely and Loosely Coupled Collaboration.

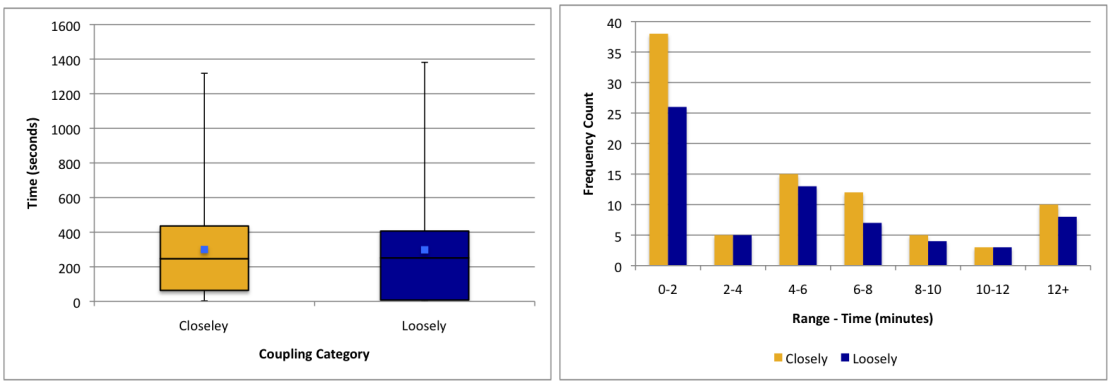
(b) Histogram of Frequency of Closely and Loosely Coupled Collaboration.

Figure 8.1: Observed Frequency of Closely and Loosely Coupled Categories.

8.2.2 Time Spent in Coupling Categories

Figure 8.2(a) shows a boxplot of the amount of time spent in the coupling categories. 57% of time was spent closely coupled and 43% spent loosely coupled. This finding indicates that pairs primarily worked and spent more time closely coupled together. The pairs did two group sections from the condition type combinations (e.g. GIG and IGG). The boxplot shows that there is a similar variance and range in the values, however, pairs did spend more time closely coupled.

Figure 8.2(b) shows an histogram of the amount of time spent in the coupling categories. Closely coupled categories spent the longest time in each of the ranges compared with loosely coupled. For two ranges (2–4 and 10–12 minutes) loosely coupled spent the same amount of time. In combination with Figure 8.1(b) this finding shows that pairs regularly switched between closely coupled styles, and when in a loosely coupled style pairs spent more time in that style and switched less frequently. We ran the Wilcoxon rank sum test at 95% confidence to compare if there were any statistical significant differences between the time spent in the coupling categories. We found no significant differences between the categories ($W = 3201$, $p = 0.2789$).



(a) Boxplot of Time Spent in Closely and Loosely Coupled Collaboration.

(b) Histogram of Time Spent in Closely and Loosely Coupled Collaboration.

Figure 8.2: Observed Time Spent in Closely and Loosely Coupled Categories.

8.2.3 Temporal Sequence of Coupling Categories

Following Isenberg [137], Figure 8.3 shows the temporal sequence of coupling categories by pairs including how much time was spent in each category. GIG pairs (odd numbered and shaded grey rows) are located on the bottom half, and IGG pairs (even numbered and white rows) located on the top half. Loosely coupled is encoded blue and closely coupled yellow. Help was for when the

session instructor had to restart SourceVis as it crashed or clarify an aspect about the user tasks or interface. No Interaction (NI) was for when participants switched between the different conditions as part of the user tasks.

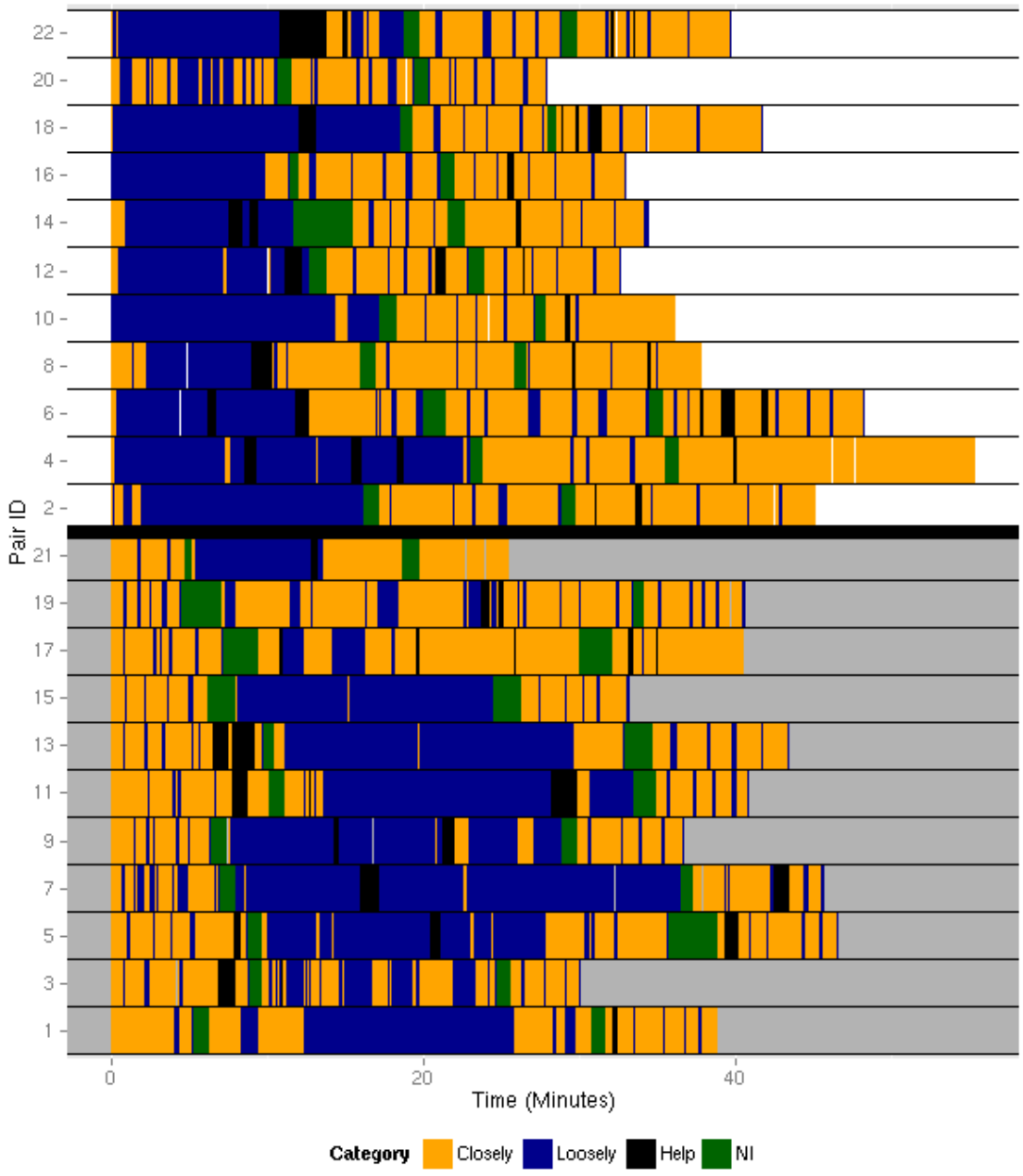


Figure 8.3: Temporal Sequence of Coupling Categories. GIG pairs (odd numbered, shaded grey rows) located bottom half and IGG pairs (even numbered, white rows) top half. Closely Coupled = blue. Loosely Coupled = yellow. Help = black. No Interaction (NI) = green.

Figure 8.3 shows that when pairs were in the Individual condition they were predominantly loosely coupled. When pairs when in the Group condition they were predominantly closely coupled, and occasionally switched to loosely coupled but only for very brief periods. In the Figure IGG pairs have a large amount of blue at the start followed by two sections of yellow. For GIG pairs they have one section of yellow, followed by a blue section, and then another yellow section. GIG pair 3 and IGG pair 20 show about half closely coupled and half loosely coupled in the Individual section. GIG pairs 17 and 19 show working almost exclusively closely coupled in the Individual section. Participants from pairs 19 and 20 commented earlier about having issues working as individuals (§ 7.2.1).

8.3 Collaborative Coupling Styles

Q7 - Which coupling styles did the participants use? (§6.1.2)

We wanted to know which collaborative coupling styles the participants used, how much time was spent in each coupling style, and what sequence the participants performed the different coupling styles. The collaborative coupling styles were described earlier (see Table 6.1).

8.3.1 Frequency of Coupling Styles

Table 8.1 shows the raw data for the observed frequency of each coupling style each pair used. In total using the ELAN tool (§6.3.5) we coded 3053 coupling styles. Yellow cells in the table indicate the largest value in a row. Red cells indicate outlier values. Closely Coupled was the most frequently used category (2381 occurrences) compared to loosely coupled (672). The most frequently used coupling style was VE (1101) followed by DISC (877), VD (438), SPDA (253), DPDA (216), and SPSA (150). The least used coupling style was DPSA (18). The average number of coupling styles used by each pair was 139. Pair 2 (235) used the largest number of coupling styles followed by Pair 3 (169). For 19 of the pairs VE was the most frequently used style. While DISC was the most frequently used style instead of VE, for pairs 7, 9, and 17 who were all from the GIG condition. SPSA was rarely used but Pairs 2 and 3 used this style more than other pairs. SPSA was for when both participants were touching the same area. We observed participants using SPSA when learning from each other. Table I.1 in Appendix I shows a more detailed table of the coupling styles separated by the Group and Individual conditions.

Table 8.1: Frequency of Coupling Style by each pair. GIG = grey shaded rows and IGG = white rows. Largest row value = yellow. Outlier values = red.

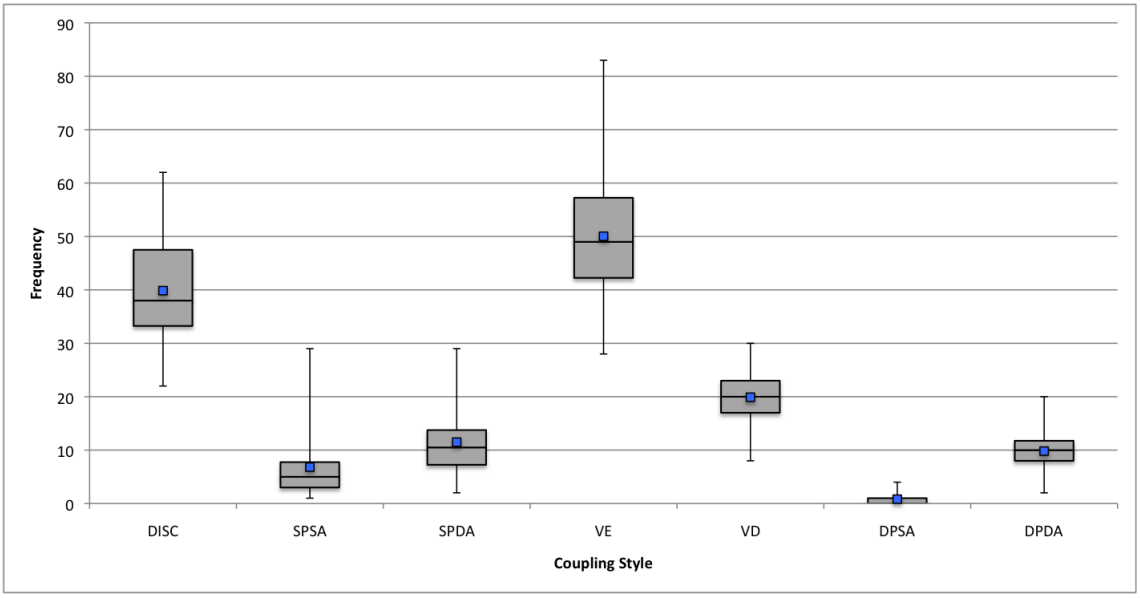
PairID	DISC	SPSA	SPDA	VE	VD	DPSA	DPDA	SUM
1	32	2	7	43	16	0	10	110
2	53	19	29	83	28	4	19	235
3	34	29	8	61	22	3	12	169
4	38	12	21	52	16	1	15	155
5	49	10	12	59	20	1	9	160
6	32	8	19	54	23	2	12	150
7	33	12	7	28	30	3	20	133
8	51	3	18	73	17	1	6	169
9	39	3	6	36	20	0	9	113
10	43	5	13	44	18	0	10	133
11	51	4	11	52	22	0	10	150
12	38	5	9	54	19	2	10	137
13	29	5	10	42	23	0	10	119
14	22	6	3	36	17	0	11	95
15	31	1	2	38	24	1	14	111
16	40	5	11	42	18	0	6	122
17	62	1	11	58	8	0	2	142
18	34	3	14	43	17	0	8	119
19	37	2	6	46	22	0	2	115
20	36	2	10	55	27	0	4	134
21	52	7	9	58	8	0	8	142
22	41	6	17	44	23	0	9	140
SUM:	877	150	253	1101	438	18	216	**3053**
Category:	Closely: 2381				Loosely: 672			

Figure 8.4(a) shows the observed frequency of coupling styles for all pairs and participants. The most frequently used style was VE at 36%, DISC 29%, VD 14%, SPDA 8%, DPDA 7%, SPSA 5%, and DPSA 1%. VE had the most variance in values followed by DISC. VE, DISC, and VD were the only styles that had either one participant or no participants interacting with the table. Therefore 50% (VE and VD) of the frequency styles one participant was interacting with the table and up to 79% (VE, DISC, VD) either one participant interacted or no participants at all. When participants were working on the same question and interacting at the same time they were mainly in different areas (SPDA) and only occasionally in the same area (SPSA). When participants were working on different questions and interacting at once they were mainly in different areas (DPDA). Participants rarely worked on different questions in the same area (DPSA).

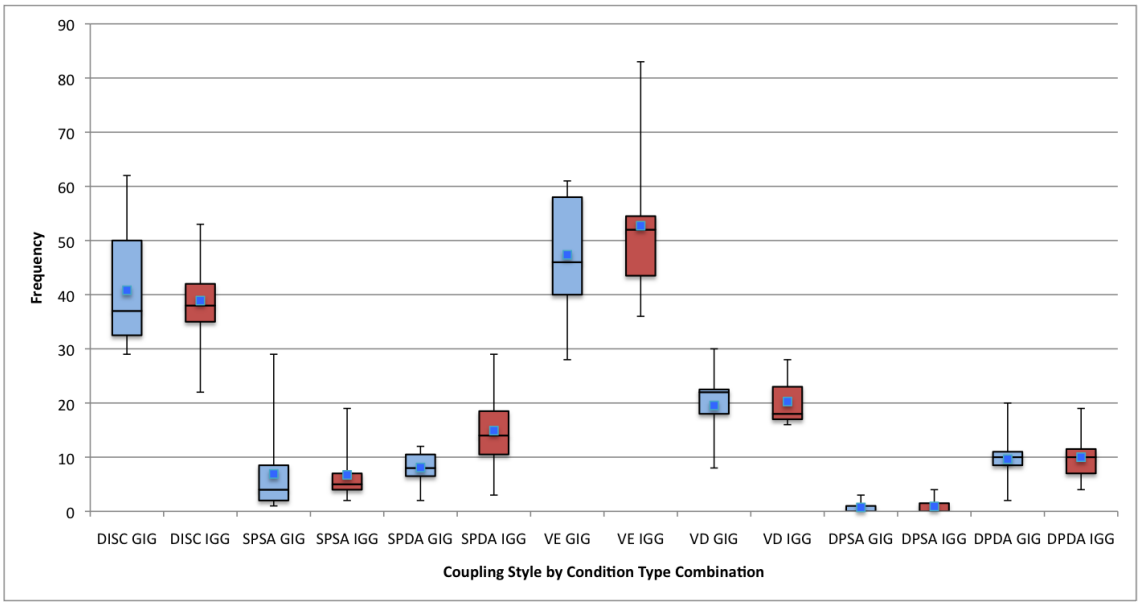
We ran pairwise Wilcoxon rank sum tests at 95% confidence to compare if there were any statistical significant differences between the frequency of coupling styles using a Bonferroni correction to the p-value. Table I.5 in Appendix I shows the results from these tests. We found statistical differences between all styles except for the following styles: DISC and VE (W = 116, p = 0.067116), SPSA and SPDA (W = 113.5, p = 0.054684), SPSA and DPDA (W = 131.5, p = 0.200886), and SPDA and DPDA (W = 273.5, p = 9.7671). This shows that these coupling styles were frequently used about as much as each other.

Figure 8.4(b) shows the observed frequency of the coupling styles by condition type combination (GIG = blue and IGG = red). GIG pairs used 1464 coupling styles and IGG pairs 1589 styles. GIG pairs used VE 36% and IGG 37%. IGG had a greater maximum value for VE while GIG a smaller minimum value. GIG pairs used DISC 31% while IGG 27%. GIG had a larger range for DISC while IGG was more tightly compacted and had a lower minimum value. GIG pairs used VD 15% and IGG 14%. GIG had a larger range for VD. Both GIG and IGG pairs used DPDA 7%. GIG had a larger variance in values for DPDA. GIG used SPDA 6% and IGG 10%. Both GIG and IGG pairs used SPSA 5%. Both GIG and IGG pairs used DPSA less than 1%.

We ran the Wilcoxon rank sum test at 95% confidence to compare if there were any statistical significant differences between the condition type combinations (e.g. GIG vs. IGG) for each of the frequency coupling styles. We found no significant differences between any of the styles, except SPDA. There was a significant difference between GIG and IGG for SPDA (W=20 and p=0.008475). This shows that IGG pairs were more often working on the same question and different areas of a visualization in the Group condition than GIG pairs.



(a) Frequency of Coupling Styles.



(b) Frequency of Coupling Style by Condition Type Combination (e.g. GIG = blue, IGG = red)

Figure 8.4: Observed Frequency of Collaborative Coupling Styles.

8.3.2 Time Spent in Coupling Styles

Table 8.2 shows the raw data for the observed amount of time spent in the coupling styles by each pair used. Yellow cells in the table indicate the largest value in a row. Red cells indicate outlier values. In total there were approximately 13 hours or 46078 seconds spent in the coupling styles. The longest time was spent in VE (14796 just over 4 hours), DPDA (13075 seconds, just over 3.5 hours), DISC (7678 just over two hours), and VD (6512 just under two hours). The average amount

Table 8.2: Time spent (seconds) in Coupling Style by each pair. GIG = grey shaded rows and IGG = white rows. Largest row value = yellow. Outlier values = red.

PairID	DISC	SPSA	SPDA	VE	VD	DPSA	DPDA	SUM
1	291	16	99	814	219	0	759	2198
2	355	50	303	848	361	14	618	2549
3	240	115	47	727	148	13	334	1624
4	423	72	422	909	312	3	942	3083
5	424	52	89	723	268	3	879	2438
6	314	65	434	845	344	8	573	2583
7	254	95	98	230	408	27	1381	2493
8	441	18	111	977	260	9	234	2050
9	343	14	109	357	243	0	935	2001
10	415	28	96	451	360	0	687	2037
11	453	22	93	418	313	0	834	2133
12	362	27	90	495	228	4	498	1704
13	283	15	110	659	403	0	844	2314
14	206	32	25	749	314	0	334	1660
15	211	3	16	478	351	8	713	1780
16	420	36	164	508	243	0	494	1865
17	615	7	118	1075	68	0	211	2094
18	347	11	275	500	380	0	772	2285
19	333	12	81	1318	325	0	114	2183
20	287	9	88	716	374	0	91	1565
21	314	44	59	488	90	0	404	1399
22	347	33	225	511	500	0	424	2040
SUM:	7678	776	3152	14796	6512	89	13075	**46078**
Category:	Closely: 26402				Loosely: 19676			

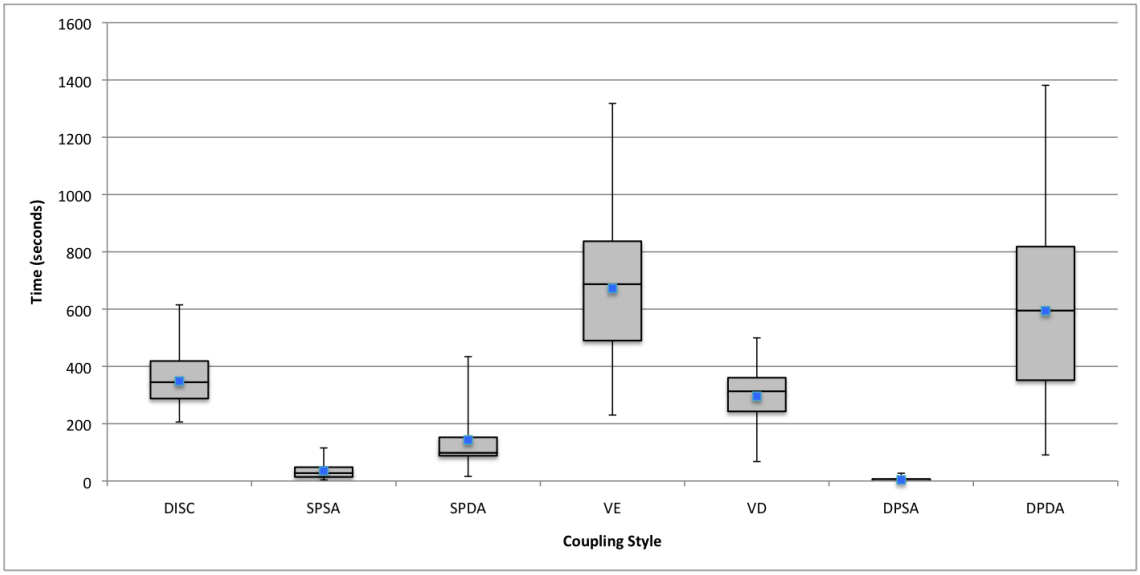
of time spent across all the coupling styles was 2049 seconds, approximately 34 minutes. Pair 4 spent the longest time in the coupling styles (3083), then Pair 6 (2583), and Pair 2 (2549). For 12 pairs VE was where pairs spent the longest time and for 10 pairs it was VE. Pairs 19 and 20 spent the shortest time in DPDA. Pairs 4 and 5 spent the longest time in SPDA. Pair 3 spent the most time in SPSA. Table I.2 in Appendix I shows a more detailed table of the coupling styles separated by the Group and Individual conditions.

Figure 8.5(a) shows the observed amount of time spent in the coupling styles for all pairs and participants. The longest time was spent in VE at 32%, DPDA 28%, DISC 17%, VD 14%, SPDA 7%, SPSA 2%, and DPSA less than 1%. Compared with the frequency of DPDA (see Figure 8.4(a)) participants spent less time changing between this code and others, and more time within DPDA which also had the largest range in values. This is due to the participants working in the Individual condition where there was less opportunity to switch to another coupling style. When pairs were working together 32% (VE) of the time there was one person interacting or 17% (DISC) discussing something about the question or the interface. Only 8% of the time pairs were they both interacting on the table at the same time (SPDA and SPDA). 14% of the time one participant was away from the table writing down answers and the other was at the table (VD).

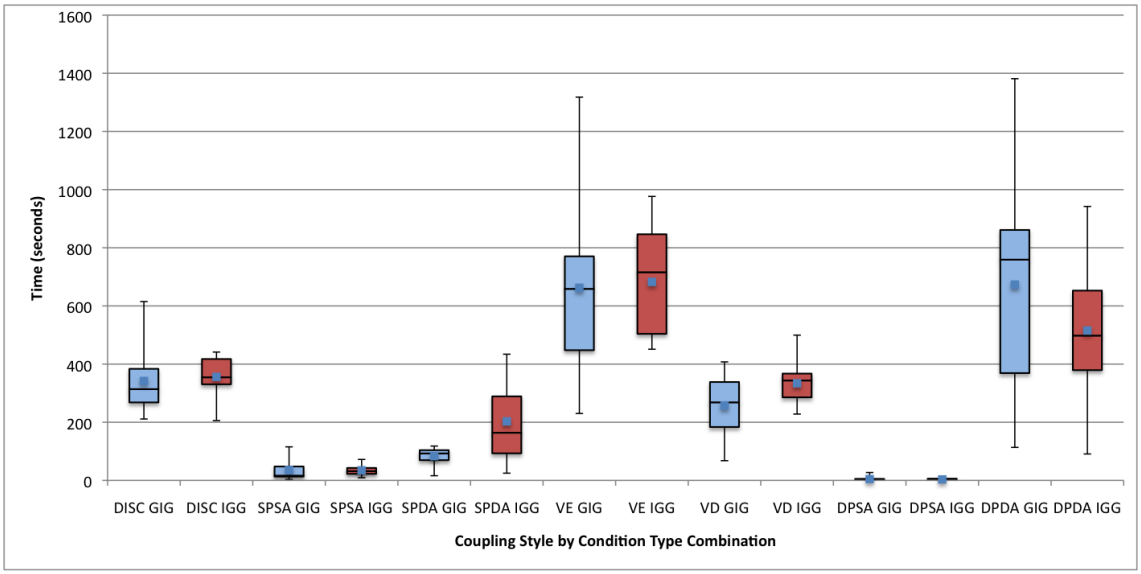
We ran pairwise Wilcoxon rank sum tests at 95% confidence to compare if there were any statistical significant differences between the time spent in the coupling styles using a Bonferroni correction to the p-value. Table I.6 in Appendix I shows the results from these tests. We found statistical differences between all styles except for the following styles: VE and DPDA (W = 281, p = 7.7658), DISC and VD (W = 303, p = 3.297), and DISC and DPDA (W = 123, p = 0.097125). This shows that pairs clearly spent more time in VE and DPDA than the other coupling styles.

Figure 8.5(b) shows the observed amount of time spent in the coupling styles by condition type combination. Both GIG and IGG pairs each spent 32% of time in VE. GIG pairs spent 33% in DPDA and IGG 24%. Both combinations spent 17% in DISC. For VE and DPDA, GIG pairs had a much larger range of values. IGG spent more time in VD at 16% and SPDA at 9%, compared with GIG at 12% and 4% respectively. Both combinations spent less than 2% in SPSA and DPSA.

We ran the Wilcoxon rank sum test at 95% confidence to compare if there were any statistical significant differences between the different condition type combinations for the time spent in the coupling styles. We found no significant differences between any of the combinations, except SPDA. There was a significant difference between GIG and IGG for SPDA (W=29 and p=0.03998). Again this is a similar finding to the frequency coupling style of SPDA for the combinations (§8.3.1). This shows that when IGG pairs were working on the same question they spent more time in different areas of the visualizations than GIG pairs.



(a) Time spent in Coupling Styles.



(b) Time spent in Coupling Style by Condition Type Combination (e.g. GIG = blue, IGG = red)

Figure 8.5: Observed Timing of Collaborative Coupling Styles.

8.3.3 Temporal Sequence of Coupling Styles

Following Isenberg [137], Figure 8.6 shows the temporal sequence of the coupling styles by pairs including how much time was spent in each style. GIG pairs (odd numbered and shaded grey rows) are located on the bottom half, and IGG pairs (even numbered and white rows) located on the top half. Help was for when the session instructor had to restart SourceVis as it crashed or clarify an aspect about the user tasks or interface. No Interaction (NI) was for when participants switched between the different conditions as part of the user tasks.

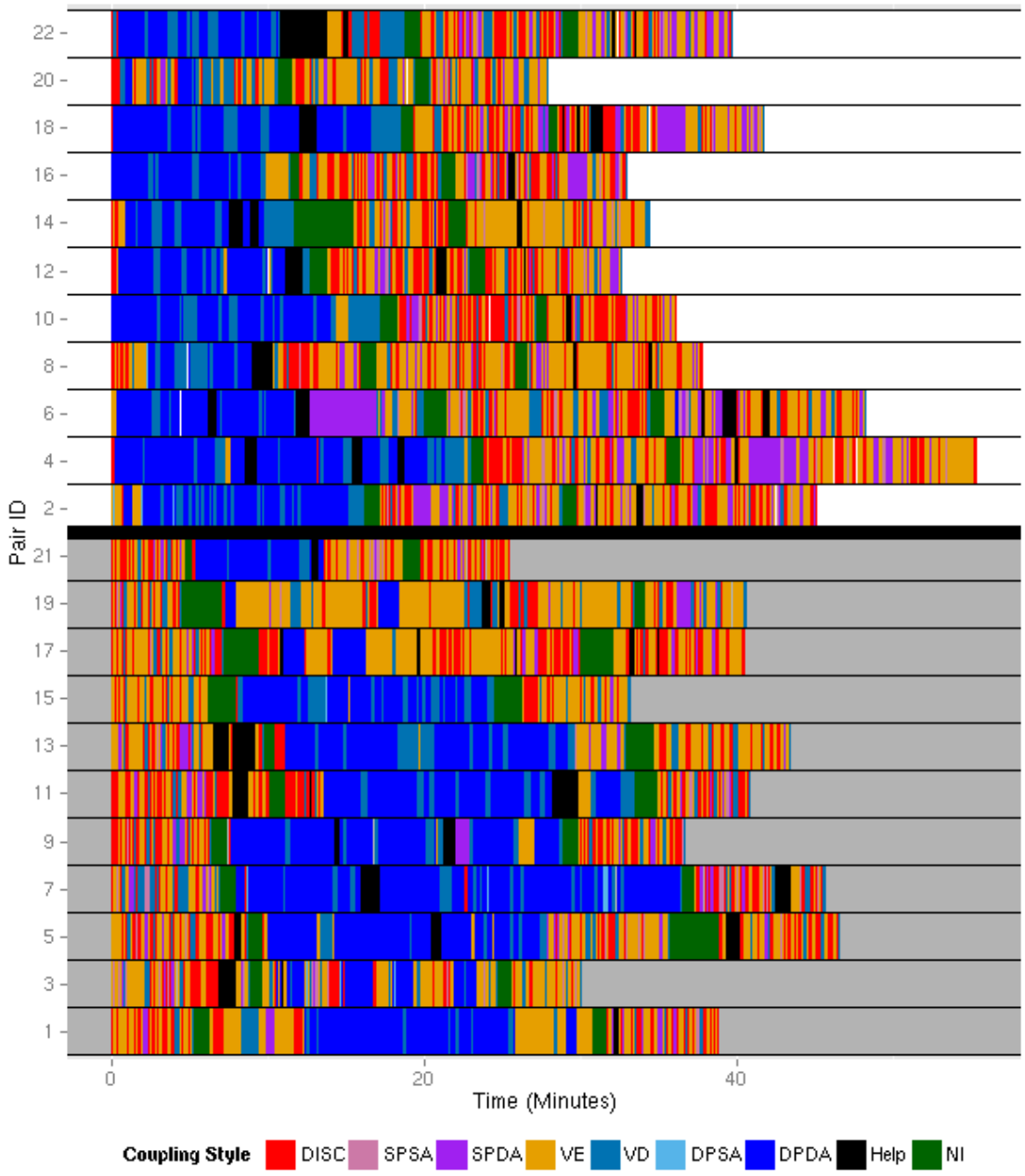


Figure 8.6: Temporal Sequence of Coupling Styles.

The Figure shows a high concentration of loosely coupled styles (blue shading) in the Individual sections and closely coupled styles (yellow to red shading) in the Group sections. When participants were loosely coupled they were in DPDA and remained in that style. When participants did switch styles from DPDA, they predominantly switched to VD and then back to DPDA. When pairs were closely coupled they were mainly in VE. The pairs quite often switched to other closely coupled styles for short amounts of time including: DISC, SPDA, and SPSA. Some

IGG pairs begin with closely coupled styles and then complete the section using loosely coupled styles. IGG pairs (2,4,6,16, and 18) and GIG pairs (9 and 19) exhibit segments of SPDA. In particular Pairs 6 and 9 spent time in SPDA when they were in the Individual section and working on different questions but in fact they were the same question (see questions A9 and B8 from Appendix H).

8.3.4 Frequency vs. Time Spent in Coupling Styles

Figure 8.7 summarises our earlier findings and shows comparing frequency versus time spent in each of the coupling styles for all pairs. The finding shows a similar pattern for all styles where they were more frequently used than the amount of time spent in them, except DPDA. VE was most frequently used at 36% and time spent 32%. DISC was used 29% and time spent 17%. VD was used 14% and time spent 14%. DPDA was was used 7% and time spent 28%. When pairs were in DPDA during the Individual condition they spent more time in the style and switched less often to other styles.

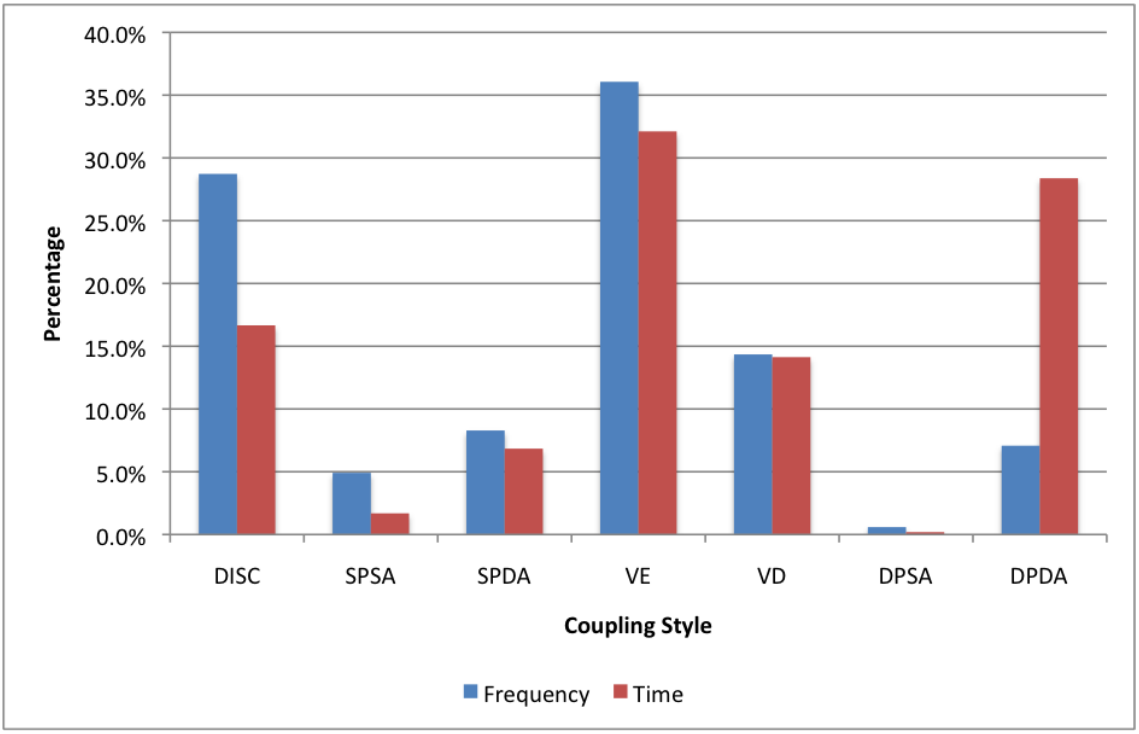


Figure 8.7: Frequency vs. Time Spent in all Coupling Styles by pairs.

8.4 Physical Arrangement Style

Q8 - Which physical arrangement styles did the participants use? (§6.1.3)

We wanted to know what physical arrangement styles participants favoured the most (§6.1.3) and how much time was spent in the arrangement styles. Our table had three sides that participants could use: (A) Bottom, (B) Left, and (C) Right (see Figure 6.3(a)). The fourth side of the table was not accessible as we had camera equipment setup preventing participants from using that side. Subsequently participants could stand in the following positions around the table based on relative positions: (A) Together, (B) Kitty Corner, (C) Side by Side, (D) End Side, and (E) Opposite Ends, which is a subset from Tang et al. [320] (see Figure 6.1).

8.4.1 Frequency of Arrangement Styles

Table 8.3 shows the raw data for the observed frequency of the arrangement styles by each pair. Yellow cells indicate the largest value in a row. Red cells indicate outlier values. In total we coded 1527 arrangement styles. Arrangement styles that supported loosely coupled collaboration styles (798) were slightly more frequently used than styles that supported closely coupled (729). The average number of arrangement styles used by pairs were 69. Pair 7 used the most arrangement styles (102). Apart (434) was the most frequently used style, followed by Side by Side (331) and Together (300). 15 pairs had Apart as their most frequently used style. 6 pairs had Together as their most used style and three pairs Side by Side. Pairs 1 and 21 had both Together and Side by Side as their most frequently used style. Pair 3 had End Side (28) as the most frequently used style, and rarely used Side by Side (4) and Together (1). Pair 10 also rarely used Together (1). Opposite Ends was the more popular style for when participants were working individually. Pairs 8 and 17 did not use Opposite Ends at all, while some pairs used this style fewer than five times. Kitty Corner was the least frequently used style and Pairs 8 and 9 did not use this style at all. Table I.3 in Appendix I shows a more detailed table of the arrangement styles separated by the Group and Individual conditions.

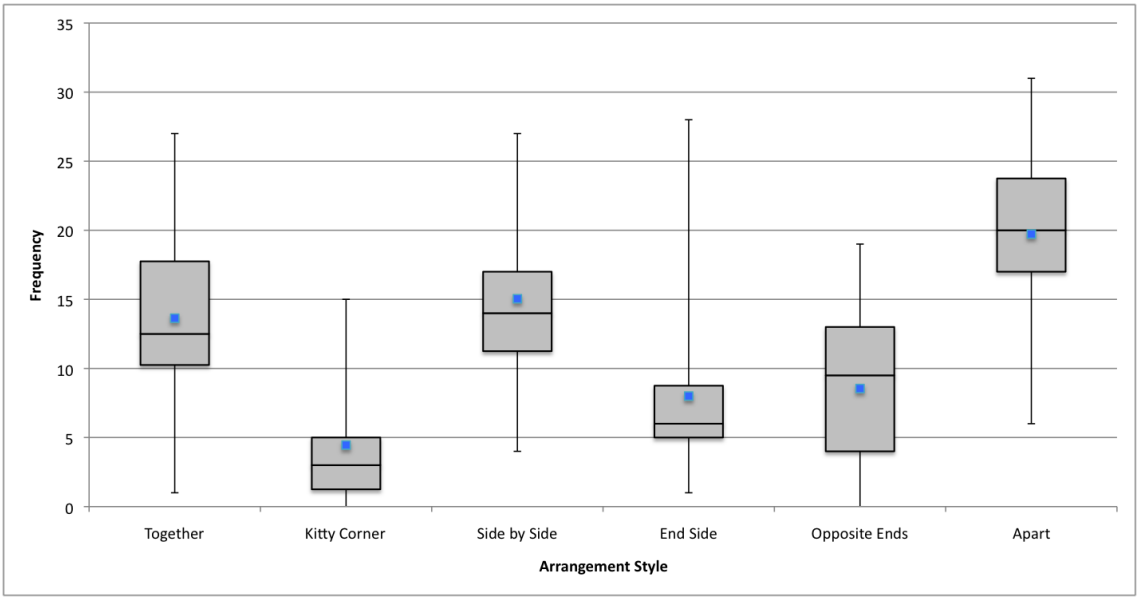
Figure 8.8(a) shows the observed frequency of the arrangement styles for all pairs. The most frequently used style was Apart at 28%, Side By Side 22%, Together 20%, End Side 12%, Opposite Ends 12%, and Kitty Corner 6%. Apart was by far the most frequently used style due to participants leaving the table to write answers down. Excluding the Apart style, Side by Side was the most frequently used style from the styles described by Tang et al. [320]. The table was most accessible from the bottom side which is where we coded Side by Side (labelled A in Figure 6.3(a)). When visualizations were displayed in full screen, Side by Side was the optimal

Table 8.3: Frequency of Arrangement Styles by each pair. GIG = grey shaded rows and IGG = white rows. Largest row value = yellow. Outlier values = red.

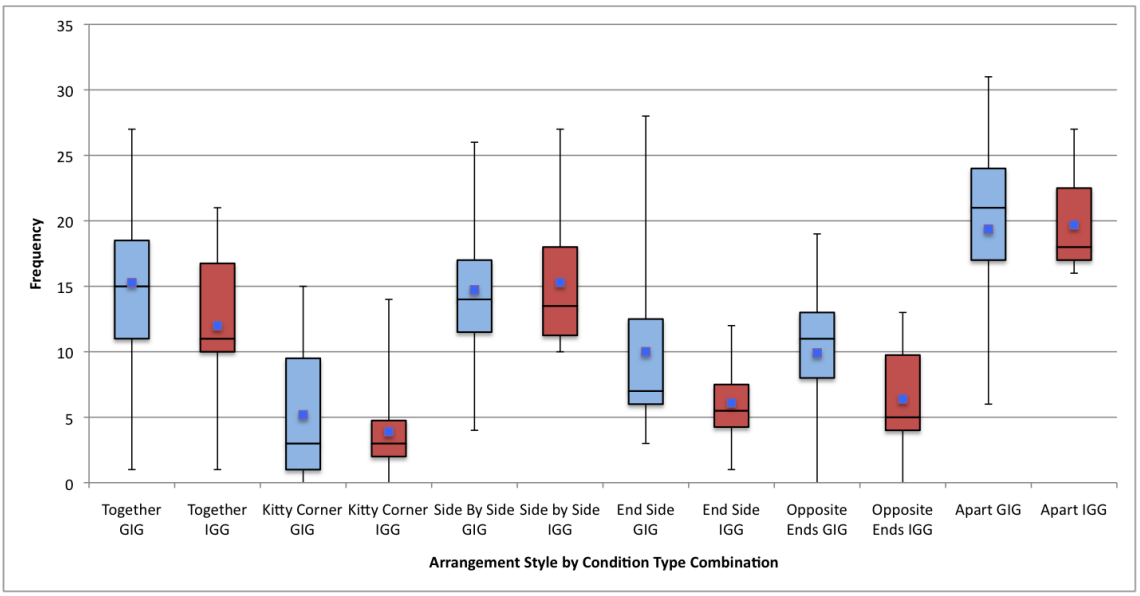
Pair	Together	Kitty Corner	Side by Side	End Side	Opposite Ends	Apart	SUM
1	15	1	15	7	11	15	64
2	12	2	16	5	15	24	74
3	1	12	4	28	10	19	74
4	17	3	12	6	13	16	67
5	27	1	25	9	11	21	94
6	16	5	15	8	10	24	78
7	17	1	17	17	19	31	102
8	21	0	27	1	0	18	67
9	13	0	17	7	13	19	69
10	1	1	10	3	13	18	46
11	27	7	26	4	9	23	96
12	19	4	21	5	6	23	78
13	11	15	11	16	13	25	91
14	4	5	11	12	1	16	49
15	11	3	12	6	13	22	67
16	10	2	13	5	4	17	51
17	19	2	14	3	0	7	45
18	10	3	11	11	4	17	56
19	18	12	12	6	3	25	76
20	11	14	19	6	4	27	81
21	9	3	9	7	7	6	41
22	11	2	14	4	9	21	61
SUM	300	98	331	176	188	434	**1527**
Category:	Closely: 729			Loosely: 798			

position for multiple participants to view and interact with the visualizations. Side by Side and Together show similar variance in values. Together was used for when participants were in discussion. End Side and Opposite Ends were used about the same amount, but End Side had a larger variance in values. Kitty Corner was the least frequently used.

We ran pairwise Wilcoxon rank sum tests at 95% confidence to compare if there were any statistical significant differences between the frequency of arrangement styles using a Bonferroni correction to the p-value. Table I.7 in Appendix I shows the results from these tests. We found statistical differences the following styles: Together and Kitty Corner (W = 407, p = 0.00162), Together and End Side (W = 368.5, p = 0.04563), Kitty Corner and Side by Side (W = 42.5, p = 0.00004263), Kitty Corner and Apart (W = 10, p = 7.90×10^{-07}), Side by Side and End Side (W = 406, p



(a) Frequency of Arrangement Styles.



(b) Frequency of Arrangement Styles by Condition Type Combination (e.g. GIG = blue, IGG = red)

Figure 8.8: Observed Frequency of Arrangement Styles.

= 0.0018105), Side by Side and Opposite Ends (W = 385.5, p = 0.011412), End Side and Apart (W = 45.5, p = 0.000060765), Opposite Ends and Apart (W = 36.5, p = 0.000021645). These tests show that Apart, Side by Side, and Together were quite similar. Opposite Ends, End Side, and Kitty Corner were similar as well.

Figure 8.8(b) shows the observed frequency of the arrangement styles by condition type combination. GIG pairs used 819 arrangement styles and IGG pairs 708 styles. The Figure shows a similar pattern for both combinations. For both combinations Apart was the most used arrangement style 26% for GIG and

31% for IGG. GIG had a larger variance in values for Apart. Side by Side was used 20% for GIG and 24% for IGG. Together was used 21% for GIG and 19% for IGG. End Side was used 13% for GIG and 9% for IGG. Opposite Ends was used 13% for GIG and 11% for IGG. Kitty Corner was used 7% for GIG and 6% for IGG.

We ran the Wilcoxon rank sum test at 95% confidence to compare if there were any statistical significant differences between the different condition type combinations (e.g. GIG and IGG) with the frequency of arrangement styles. We found no significant differences between any of the combinations. Given the box plot in Figure 8.8(b) we thought that there maybe a significant difference for Apart and Kitty Corner but this was not the case ($W = 64$, $p = 0.8435$).

8.4.2 Time Spent in Arrangement Styles

Table 8.4 shows the raw data for the observed amount of time spent in the arrangement styles by each pair. The longest time was spent in the Side by Side arrangement style (17110 seconds, approximately 4 and three quarter hours), then Opposite Ends (10230 seconds, just under 3 hours), Apart (6703 seconds, just under 2 hours), Together (5664 seconds, just over 1 and half hours), End Side (8809 seconds, approximately 1.5 hours), and Kitty Corner (4276 seconds, approximately 45 minutes). More time was spent in arrangement styles that supported closely coupled collaboration styles (25562 second) than styles that supported loosely coupled (22502 seconds). Pairs spent on average 2185 second approximately 36 minutes in the arrangement styles. Pair 4 spent the longest time in the arrangement styles (3214 seconds), followed by Pair 6 (2757 seconds), and both of these pairs were in the IGG condition. 16 pairs spent the longest time in Side by Side. Five pairs spent the longest time in Opposite Ends. Pairs 8 and 17 spent no time in Opposite Ends, but were the two pairs that spent the most time in Side by Side. Pair 14 spent a very short amount of time in Opposite Ends too. Pair 4 spent more time in Opposite Ends compared with other pairs, but spent the longest time in Side by Side. Pairs 8 and 9 spent no time in Kitty Corner. Pair 3 spent the longest time in End Side and the shortest time in Together (5 seconds) and Side by Side (56 seconds). Table I.4 in Appendix I shows a more detailed table of the arrangement styles separated by the Group and Individual conditions.

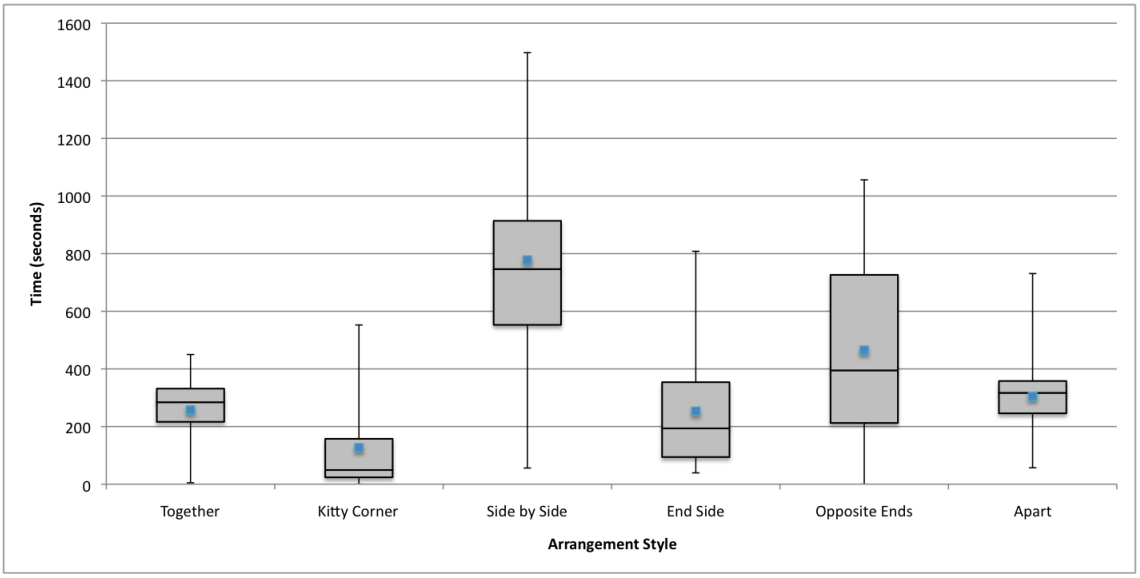
Figure 8.9(a) shows the observed amount of time spent in the arrangement styles for all pairs. The longest time was spent in Side by Side 36%, Opposite Ends 21%, Apart 14%, Together 12%, End Side 11%, and Kitty Corner 6%. Side by Side was the style most of the time was spent in and had the largest variance in values. Most pairs used more arrangement styles than they spent time in them except for Side by Side and Opposite Ends. This was partly due to pairs spending

Table 8.4: Time spent (seconds) in the Arrangement Styles by each pair. GIG = grey shaded rows and IGG = white rows. Largest row value = yellow. Outlier values = red.

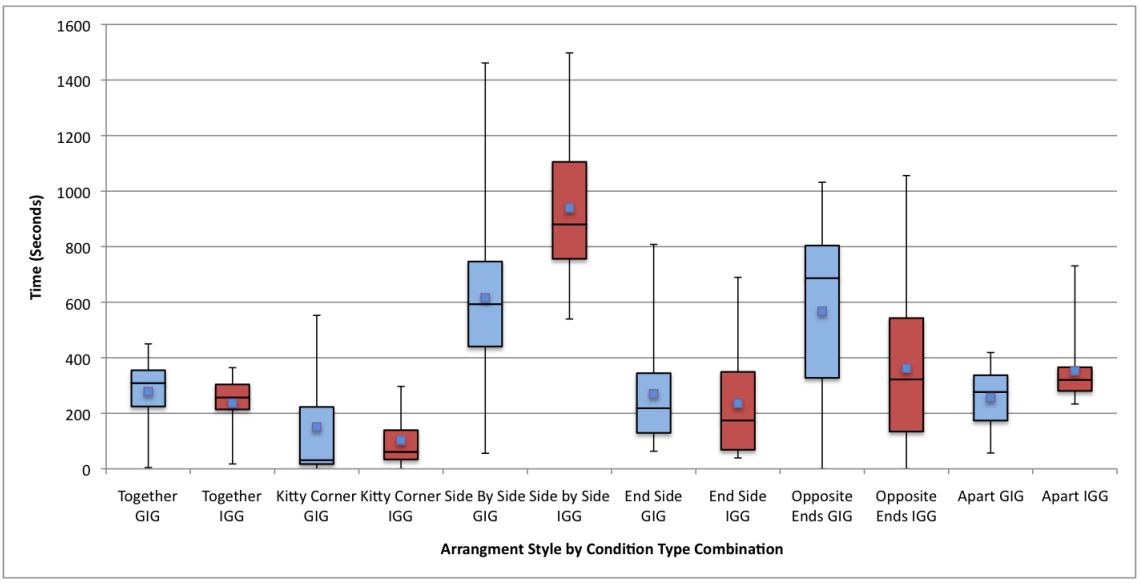
Pair	Together	Kitty Corner	Side by Side	End Side	Opposite Ends	Apart	SUM
1	216	18	669	363	740	214	2220
2	317	32	1255	149	529	314	2596
3	5	359	56	808	336	133	1697
4	291	103	1292	174	1056	298	3214
5	450	7	869	183	761	277	2547
6	334	176	880	460	557	350	2757
7	309	17	438	412	1032	419	2627
8	364	0	1498	39	0	261	2162
9	303	0	469	63	996	241	2072
10	18	36	956	55	635	360	2060
11	360	87	593	86	847	327	2300
12	257	61	718	204	322	263	1826
13	350	489	280	255	687	400	2461
14	78	297	637	433	3	320	1768
15	233	31	443	326	424	325	1782
16	278	75	794	265	248	233	1893
17	398	26	1461	218	0	57	2160
18	211	23	925	690	200	379	2428
19	325	553	767	141	103	347	2236
20	217	289	539	82	68	371	1566
21	110	73	726	118	320	83	1430
22	240	37	844	45	366	731	2263
SUM	5664	2789	17110	5569	10230	6703	**48064**
Category:	Closely: 25562			Loosely: 22502			

long periods of time in Side by Side and Opposite Ends during the Group and Individual conditions. Both of these styles show the largest variance in values. Opposite Ends increased in percentage from its frequency use as it was the style most used during the Individual Condition (Pairs 7,9, and 13), even though Pairs 8 and 17 spent zero time and Pair 14 did not much time in this style. Pairs were in Apart for both conditions and only spent a small amount of time in this style at any one time.

We ran pairwise Wilcoxon rank sum tests at 95% confidence to compare if there were any statistical significant differences between the time spent in the arrangement styles using a Bonferroni correction to the p-value. Table I.8 in Appendix I shows the results from these tests. We found statistical differences



(a) Time spent in Arrangement Styles.



(b) Time spent in Arrangement Styles by Condition Type Combination (e.g. GIG = blue, IGG = red)

Figure 8.9: Observed Time Spent in Arrangement Styles.

between six of the styles: Side by Side and Together (W = 33, p = 7.5735×10^{-07}), Side by Side and Kitty Corner (W = 23, p = 0.000004371), Side by Side and End Side (W = 438, p = 0.000008508), Side by Side and Apart (W = 440, p = 0.000006024), Kitty Corner and Opposite Ends (W = 102, p = 0.01581), and Kitty Corner and Apart (W = 86, p = 0.0039315). These tests show that Side by Side was quite different from all other styles and pairs spent more time in this style. Kitty Corner was also quite different from Side by Side, Opposite Ends, and Apart, and the shortest time was spent in this style.

Figure 8.9(b) shows the observed amount of time spent in the arrangement styles by condition type combination. The Figure shows a similar pattern for both combinations. For both combinations Side by Side was the style pairs spent the longest time in, GIG 29% and IGG 42%. GIG had a larger variance in values for Side by Side. For Opposite Ends GIG spent 27% and IGG 16%. For End Side GIG spent 13% and IGG 11%. For Apart GIG spent 12% and IGG 16%. For Together GIG spent 13% and IGG 11%. For Kitty Corner GIG spent 7% and IGG 5%.

We ran the Wilcoxon rank sum test at 95% confidence interval to compare if there were any statistical significant differences between the different condition type combinations (e.g. GIG and IGG) with the amount of time spent in the arrangement styles. We found no significant differences between any of the combinations, except Side by Side. There was a significant difference between GIG and IGG for Side by Side ($W = 24$, $p = 0.01577$). This shows that IGG pairs significantly spent more time in Side by Side than GIG pairs.

8.4.3 Temporal Sequence of Arrangement Styles

Figure 8.10 shows the temporal sequence of arrangement styles by pairs including how much time was spent in each arrangement style. GIG pairs (odd numbered) are located at the bottom half of the Figure, and IGG pairs (even numbered) located at top half of Figure. No Interaction (NI) was for when participants switched between the different conditions as part of the user tasks.

When pairs were loosely coupled they were primarily in the Individual condition and working at Opposite Ends of the table. This is illustrated by GIG pairs (1, 5, 7, 9, 11, 13, 15, and 21) that have Opposite Ends coded for their middle section (Individual Condition) in their respective rows. This is similar for IGG pairs (2, 4, 6, 10, 12, 16, and 22) that have Opposite Ends coded at the start of their row. Some pairs (3, 6, 14, 15, 16, and 18) spent quite a bit of time in End Side during the Individual condition too. Pair 6 used End Side late in the Individual condition. The other pairs sporadically used End Side, which was quite often when a participant launched a new visualization from the Startup Screen. Some pairs either rarely spent any time (14, 19 and 20) or no time (8, 17) at Opposite Ends which goes against the trend of the style used by all other pairs during the Individual condition. Apart was used in all condition types and was mainly used when one participant was writing down an answer and apart from their colleague.

When pairs were closely coupled they were primarily in the Group condition and working Side by Side at the table. When pairs were in Side by Side they mostly transitioned to the Together arrangement. Some pairs (4, 10, 17, and 21) had large stretches of being in Side by Side without transitioning to another style.

These pairs did discuss with each but instead of facing each other (e.g. Together), continued to look at the table. Pairs used Together for short periods as this style was mainly used to discuss an aspect of the visualization or clarify with their colleague about a question. Some pairs (1, 7, and 9) used Together for short periods during the Individual condition. Kitty Corner was used for short periods primarily in the Group condition, where there were lots of transitions between arrangement styles by some pairs (3, 6, 13, 14, 19 and 20). Other pairs occasionally used Kitty Corner and two pairs (8 and 9) did not use this arrangement style at all.

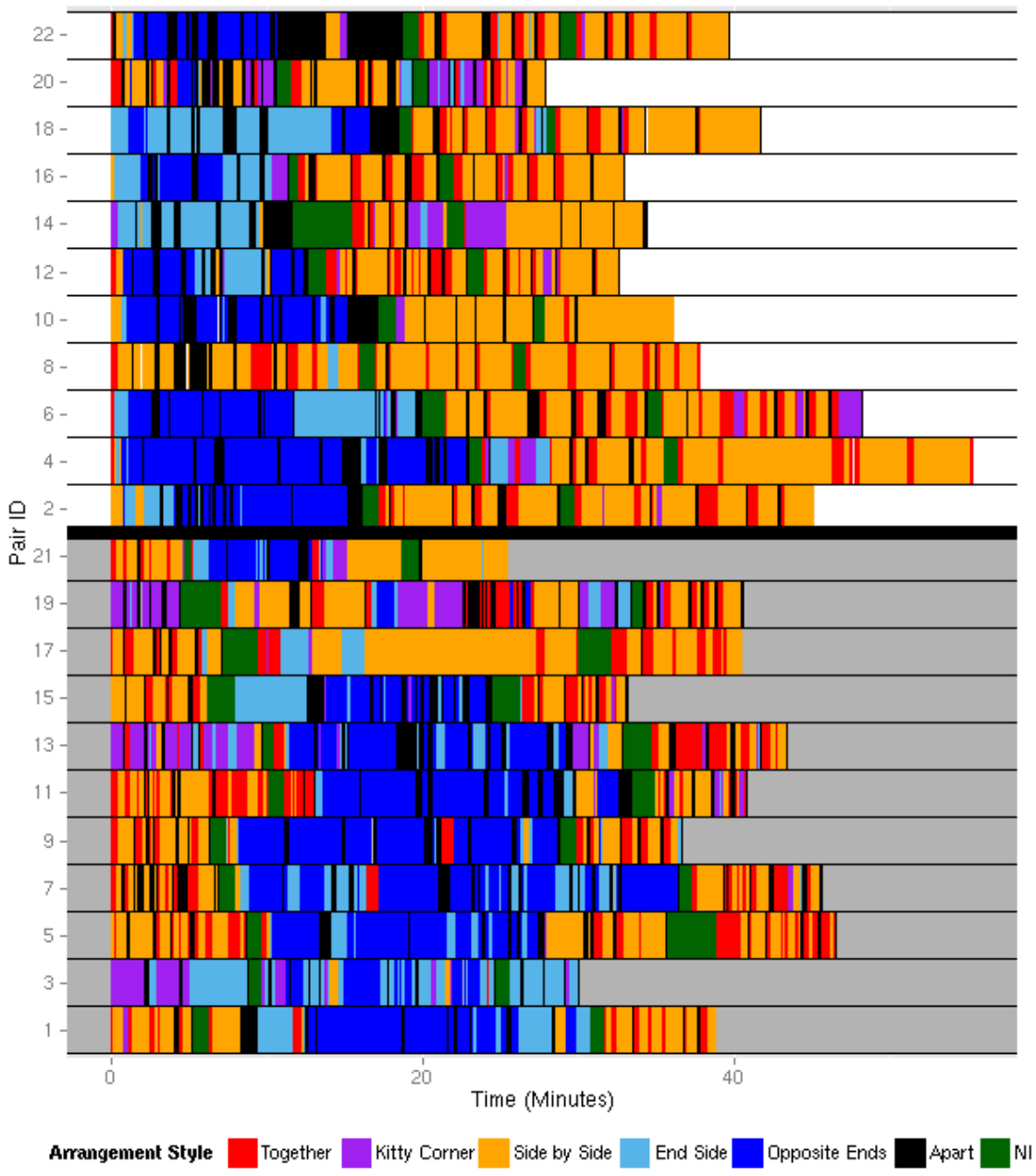


Figure 8.10: Temporal Sequence of Arrangement Styles.

8.4.4 Frequency vs. Time Spent in Arrangement Styles

Figure 8.11 shows comparing frequency versus time spent in each of the arrangement styles by pairs. The Figure shows all styles were more frequently used than the amount of time spent in them, except Side by Side and Opposite Ends. Side by Side was frequently used at 22% and time spent 36%. Apart was used 28% and time spent 14%. Together was used 20% and time spent 12%. Opposite Ends was used 12% and time spent 21%. End Side was used 12% and time spent 11%. Kitty Corner was used 6% and time spent 6%.

This shows that pairs slightly used more arrangement styles that support loosely coupled collaboration at 52% (Apart, Opposite Ends, End Side), than closely coupled collaboration at 48% (Side by Side, Together, Kitty Corner). More time, however, was spent in the arrangement styles that support closely coupled collaboration at 54% , than loosely coupled collaboration 46%.

Apart was an arrangement style that did not have both participants located around the table at once and is not one of the original arrangement styles from Tang et al. [320]. Excluding Apart, arrangement styles that support closely coupled collaboration were 67% frequently used and 62% time spent in those styles. Excluding Apart, arrangement styles that support loosely coupled collaboration were 33% frequently used and 38% time spent in those styles.

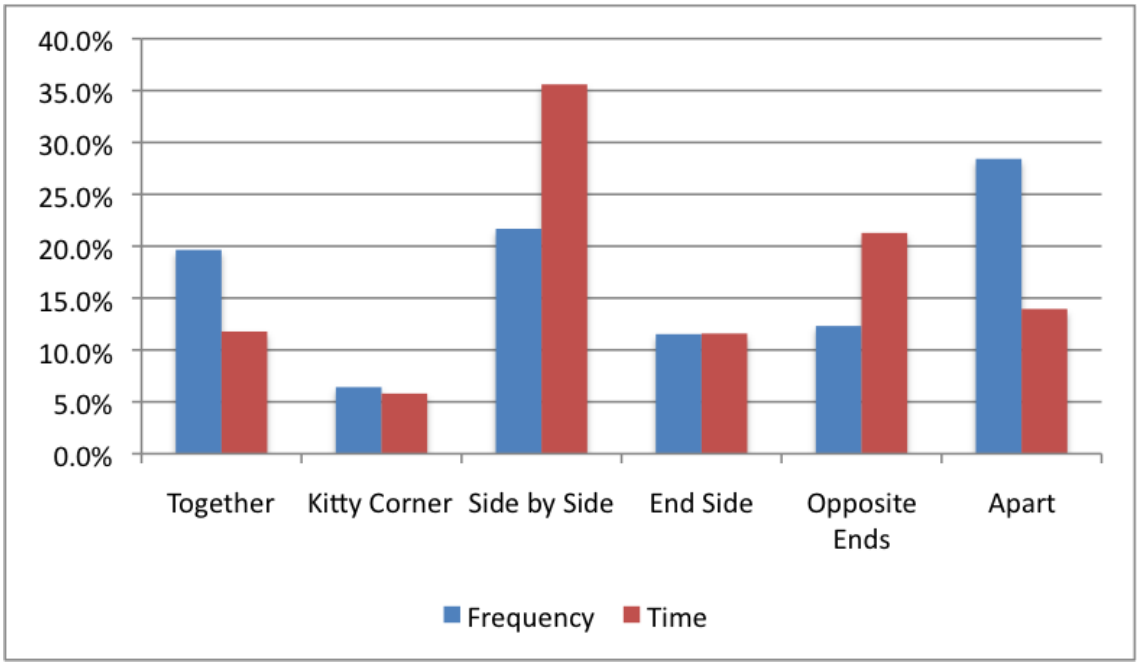


Figure 8.11: Frequency vs. Time Spent in all Arrangement Styles by pairs.

8.4.5 Collaborative Coupling and Physical Arrangement Styles

Table 8.5 shows the percentage of time working in each coupling style and arrangement style. The most amount time was spent in VE (32.1%) and Side by Side (39.3%), with the combination of time spent in both styles being 22.2%. DPDA (25.8%) and Opposite Ends (19.4%) had a combination of 18.6%. VD (15.1%) and Apart (15.1%) had a combination of 14.7%. When pairs were working Side by Side and not in VE they were either in DISC (8.2%) or SPDA (6.4%). If pairs were in DISC but not Side by Side they were Together (6.6%). When participants worked on different questions and not standing at Opposite Ends they were in End Side (6.0%). Note that SPSA and DPSA were the least used coupling styles, and Kitty Corner was the least used arrangement style.

Figure 8.12 shows the percentage of time working in each coupling style by arrangement style for all pairs. VE was the coupling style pairs spent the most time in while in the Side by Side arrangement style. DPDA was the coupling style paris spent the most time in while at Opposite Ends. DISC was the coupling style pairs spent the most time in while Together. VD was the coupling style pairs spent the most time in while Apart. DPDA was the coupling style pairs spent the most time in while in End Side. VE was the coupling style pairs spent the most time in while in Kitty Corner.

Table 8.5 and Figure 8.12 can be summarised as follows. When pairs worked closely coupled they were more closely arranged. When pairs worked loosely coupled they were more loosely arranged.

Table 8.5: Percentage of time working in each coupling and arrangement style for all pairs. White: <1%, Light grey: 1–5%, Dark grey: >5%. Yellow cells indicate the largest values. Red cells indicate outlier values.

	Coupling:								
Arrangement:	DISC	SPSA	SPDA	VE	VD	DPSA	DPDA	**Total**	**Category:**
Together	6.6	0.0	0.0	3.6	0.1	0.0	0.0	10.4	Closely: 55%
Kitty Corner	0.9	0.2	0.4	3.4	0.0	0.0	0.1	5.2	
Side by Side	8.2	1.4	6.4	22.2	0.1	0.0	1.0	39.3	
End Side	0.7	0.2	1.1	2.5	0.1	0.1	6.0	10.6	Loosely: 45%
Opposite Ends	0.2	0.0	0.1	0.3	0.2	0.0	18.6	19.4	
Apart	0.3	0.0	0.0	0.1	14.7	0.0	0.1	15.1	
Total	16.9	1.9	8.0	32.1	15.1	0.2	25.8	**100.0%**	
Category:	Closely: 58.8%				Loosely: 41.2%				

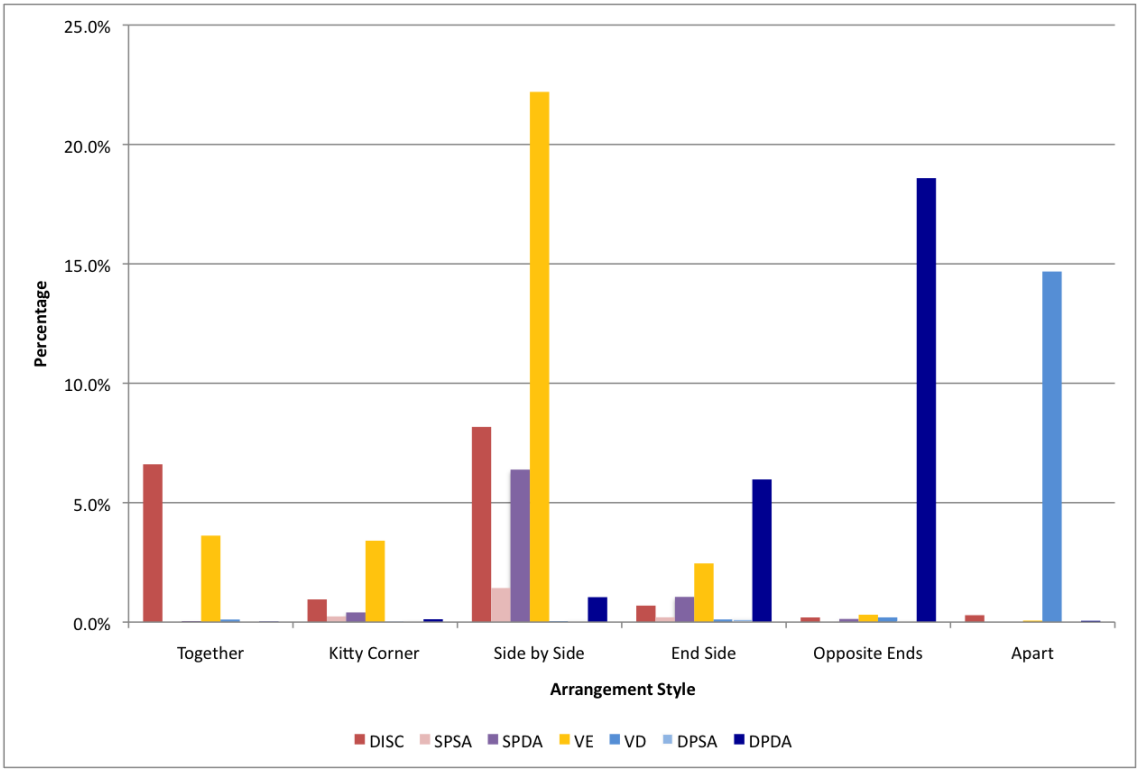


Figure 8.12: Percentage of Time Spent in Coupling and Arrangement Styles.

8.5 Perceived Effectiveness of Techniques

Q9 - Which visualization techniques did the participants perceive to be the most effective? (§6.1.4)

Figure 8.13 shows a boxplot of the perceived effectiveness of the individual visualization techniques each participant stated in the post-survey questionnaire (§6.3). The range was 0 for least effective and 10 for most effective. Most of the visualizations rank somewhere between 6.5 and 8.5 for the average. The overall perceived effectiveness of the visualizations is 7.5 for the average. The Startup Screen is 7.7 for the average.

The visualization perceived most effective was the System Class Evolution View (8.4 average). Three other evolution visualizations (System Evolution, System Package Evolution, and System Class Evolution, all above 8.0 for average) rank slightly below the System Class Evolution. The other two evolution visualizations (Individual Package and Individual Class) rank very similar (7.5–8.0 average) and have a similar range of values. The chart based visualizations rank well (7.5–7.8 average), with the Toxicity Chart (7.8 average) being the most effective of these chart visualizations. The visualizations that employ Polymetric View encodings (the evolution visualizations, and Class Blueprint), rank well, except the System

Hotspots View. The Metrics Explorer (7.4 average), Class Dependency (7.4 average) Vocabulary (7.1 average) all rank about the same.

The System Class Evolution View had the least spread of values. Neither the System Hotspots View nor the overall effectiveness had values that ranked 10. The Metrics Explorer, System Evolution, and Individual Class Evolution had a similar spread of values. The System Dependency View has the largest spread of values. The System Dependency View and Startup Screen had the lowest values (1 and 2).

The visualizations perceived least effective were the System Explorer (6.5 average), System Dependency View (6.8 average), and System Hotspots View (6.8 average). Participants struggled with the System Dependency View especially when trying to find and read the name of a class (§7). The System Hotspots View was large and some participants got lost in this visualization especially if they were trying to find a specific class. The System Explorer was a simple visualization and had two scrollable lists, but ranked the lowest of all the visualizations.

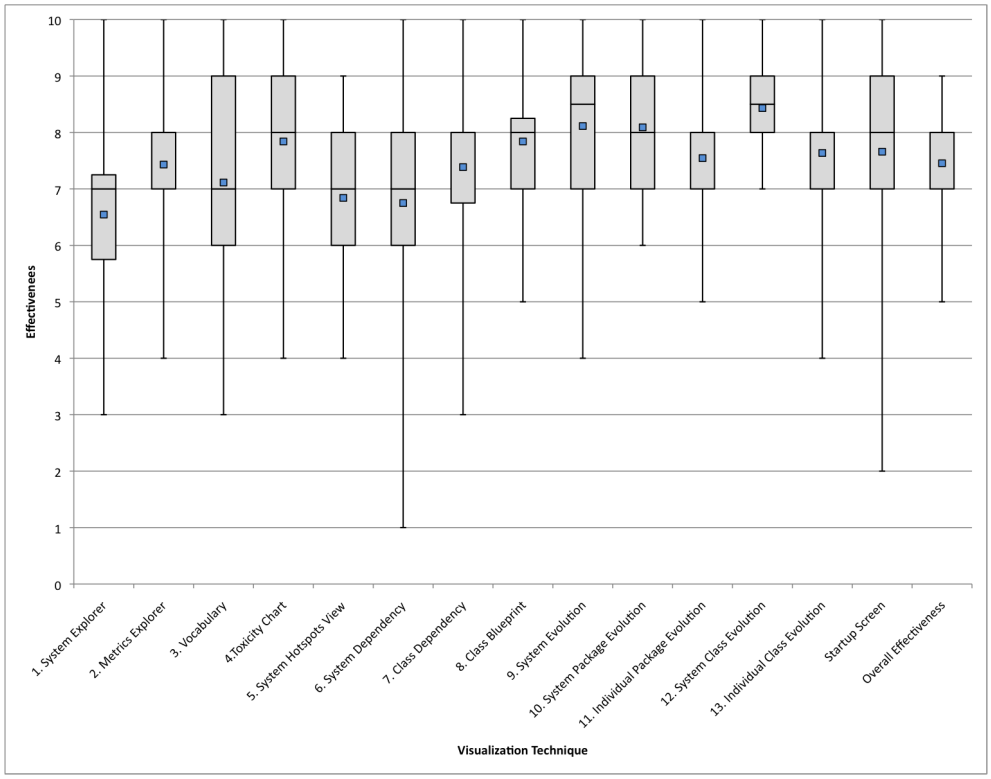


Figure 8.13: Professional User Study - perceived effectiveness of each visualization technique by participants, including the Startup Screen and Overall Effectiveness.

8.6 Summary

In this chapter we presented the *quantitative* findings from the user study involving professional software developers (§6).

All participants had the opportunity to work in both condition types, Group and Individual. All pairs selected the Group condition for Section 3 of the user tasks. This resulted in there only being two combinations GIG and IGG. Each of these combinations had 11 pairs. Neither GII or IGI combinations eventuated.

The most frequently used coupling category was closely coupled (78%) then loosely coupled (22%). The longest time was spent closely coupled (57%) then loosely coupled (43%). When participants were in the Group condition they were closely coupled and when in the Individual condition loosely coupled.

The most frequently used coupling styles were VE (36%), DISC (29%), and VD(14%). The longest time spent in the coupling styles were DPDA (32%), VE (30%), and DISC (17%). When pairs were in the Individual condition they were mainly in DPDA where they stayed a long time and only transitioned to VD. When pairs were in the Group condition they were mainly in VE and then quickly transitioned to other closely coupled styles such as DISC, SPDA, and seldom SPSA. Pairs frequently used more coupling styles than spending time in each coupling style. This was except DPDA which had a low frequency but more time was spent in that style due to the Individual condition.

The most frequently used arrangement styles were Apart (28%), Side by Side (22%), and Together (20%). The longest time spent in the arrangement styles were Side by Side (36%), Opposite Ends (21%), and Apart (14%). When pairs were in the Individual condition they were mainly in Opposite Ends or End Side as they were the optimal locations for utilizing the screen real estate for multiple users working on different tasks. When pairs were in the Group condition they were predominantly Side by Side and quickly transitioned to Together and occasionally Kitty Corner. Pairs frequently used more arrangement styles than spending time in each arrangement style. This was except Side by Side and Opposite Ends which were the dominant styles in the Group and Individual conditions.

When pairs were in VE they were arranged Side by Side (22.2% of total time). When pairs were in DPDA they were arranged at Opposite Ends (18.6% of time). If pairs were arranged Side by Side and not in VE they were in DISC (8.2% of time) or SPDA (6.4% of time). If pairs were in DPDA but not at Opposite Ends they were arranged End Side (6.0% of time). If pairs were Apart they were in VD (14.7% of time). Our findings show that when pairs worked closely coupled they were more closely arranged, and when they worked loosely coupled they were more loosely arranged.

The *perceived* effectiveness of the visualizations rank somewhere between 6.5 and 8.5 for the average (10 = most effective and 0 = least effective). The overall perceived effectiveness of the visualizations is 7.5 for the average, and Startup Screen 7.7 for the average. The visualization perceived most effective was the System Class Evolution View (8.4 average). Three visualizations rank on average above 8 (System Class Evolution, System Evolution and System Package Evolution). The visualizations perceived least effective were the System Explorer (6.5 average), System Dependency View (6.8 average), and System Hotspots View (6.8 average).

The next chapter presents the conclusions, research contributions, key findings, limitations, and future work for this thesis (§9).

Part V

Conclusions

Chapter 9

Conclusions

Contents

In this thesis we have explored the design of software visualization systems for collaborative co-located software development teams on large multi-touch tables. We built our own multi-touch tables, designed a collaborative software visualization application, and conducted user studies. In this chapter we summarise our research contributions, present key findings, discuss limitations, and look at potential directions future work.

9.1 Research Contributions

The thesis contributes a richer understanding of how pairs of developers make use of shared software visualizations on large interactive multi-touch tables to gain insight into how existing software systems are structured and how they have evolved over different versions.

9.1.1 Designing Collaborative Software Visualizations

We built two multi-touch tables following existing multi-touch table hardware guidelines [287, 321]. The first table was medium sized (28 inches, 1024x768 resolution) but was not large enough for collaborative group work, hence we built a much larger sized (48 inches, 1280x800 resolution) table (§3).

We developed multi-touch collaborative software visualization prototypes following an iterative approach [144]. To improve the design of these prototypes we conducted preliminary user studies during the development life-cycle with 18 students (§5). These user studies helped to inform the design of SourceVis and to develop the user study protocol for our subsequent professional user study.

9.1.2 SourceVis: Software Visualization Application

We designed SourceVis which is a collaborative software visualization application for use on large multi-touch tables within a co-located environment (§4). SourceVis allows multiple users to interact simultaneously or separately with the table either as joint group work or parallel individual work.

SourceVis is designed for visualizing two key areas of software systems: the structure and evolution. We adapted 13 information and software visualization techniques to multi-touch interaction [10, 77, 312]. These visualizations allow developers to explore the contents of a system, to examine how a system has been structured, and to see how the structure of a system has evolved over different versions. Multiple visualizations can be displayed at once, displayed in rotatable

and scalable windows, oriented in different directions, and support navigation gestures for zooming and panning. The visualizations provide a high level overview of a system and have features for filtering, searching, and drilling down for details about package and class entities. New visualization techniques can be added by extending existing SourceVis classes.

We demonstrated SourceVis with some Java open source systems from the Qualitas Corpus (Version 20101126) [325]. The systems ranged in size from small ($<$ 10K LOC) to very large ($>$ 200K LOC), and some contain more than 10 versions (see Table 4.1). New systems to be visualized can be added.

9.1.3 Evaluation of Collaborative Software Visualization

To evaluate our multi-touch table and collaborative software visualization designs we conducted a user study with professional software developers (§6).

We recruited 44 participants as 22 pairs. Each pair work or worked in the same team within same organisation. In all 18 organizations from Wellington, New Zealand provided participants. We specifically recruited participants from the same team as we wanted them to have experience working with each other in a professional environment.

The participants in the study worked in pairs performing software understanding tasks with SourceVis and the large multi-touch table. The tasks involved participants answering questions about the software systems from the Qualitas Corpus (Version 20101126) using SourceVis. The questions are similar to the kinds of questions that developers ask within industry [101, 170, 333, 334, 302]. Participants answered the questions either working in a group or as individuals.

We wanted to find out if participants preferred to work as a group or as individuals. We observed which collaborative coupling categories, coupling styles, and physical arrangement styles participants used and how much time was spent in these styles. We asked participants to provide feedback on the strengths and weaknesses of SourceVis and the multi-touch table and to suggest improvements. We asked participants how the multi-touch table helped with team collaboration and what visualization techniques they perceived to be the most effective.

The user study took up to two hours to complete. We video recorded the user study with multiple cameras and asked participants to think aloud so we could capture their thoughts for post-study analysis. We obtained both qualitative (§7) and quantitive findings (§8).

9.2 Key Findings

We present the key findings of this research thesis, and the implications for collaborative software visualization using multi-touch tables. Our research revealed key findings for collaborative group work, designing collaborative software visualizations, and designing multi-touch tables. The findings are based on the qualitative (§7) and quantitative findings (§8) from our user study (§6).

9.2.1 Collaborative Group Work

We wanted to find out if participants preferred group or individual work (§8.1), what coupling categories they used (§8.2), what coupling styles they used (§8.3), and what physical arrangement styles they used (§8.4).

Design for joint group work over parallel individual work

We found that pairs preferred to work in *groups* rather than *individuals* to complete the user tasks (§8.1). Groups worked jointly together and individuals worked separately in parallel. Pairs experienced working in groups and as individuals. All participants preferred to work as a group as evidenced by all 22 pairs working together as a group in Section 3 of the user tasks.

We required pairs to work as individuals for either Section 1 or Section 2 and alternated the order of pairs in which they performed the Individual condition. We found the order in which the pairs performed the Individual condition had no impact on their decision to work as a group for Section 3. Participants claimed that working as a group was easier and more effective. We observed pairs completing questions in the Group condition faster than in the Individual condition.

Some participants appreciated that the multi-touch table allowed them to work as individuals at the same time, but they preferred group work. We observed that most participants struggled performing parallel individual work on the the table. Some pairs even completed the Individual condition as a group.

The key finding is that collaborative software visualization systems should primarily be designed to support joint group work. Individuals working separately should be a secondary design consideration. This reinforces the findings from Isenberg et al. [141, 142] as applied to collaborative software visualization.

Support a flexible variety of coupling styles.

We found that pairs used more closely coupled categories to complete the user tasks (§8.2). We recorded almost 80% of categories used were closely coupled and

20% loosely coupled. We recorded almost 60% of time spent in closely coupled categories and 40% loosely coupled. When pairs were in the Group condition they were mostly closely coupled. When pairs were in the Individual condition they were almost exclusively loosely coupled.

We found that pairs used more closely coupled styles (Viewing Engaged - VE, Discussion - DISC, Same Problem Different Area - SPDA, Same Problem Same Area - SPSA) than loosely coupled styles (Different Problem Different Area - DPDA, Viewing Disengaged - VD, Different Problem Same Area - DPSA) to complete the user tasks (§8.3). Pairs spent the most time in closely coupled styles then loosely coupled styles. When pairs were in the Group condition they were mainly viewing engaged (VE), with one participant controlling the interface, and regularly discussed questions or aspects of the interface (DISC). Occasionally pairs would separate the task and work in different areas on the table (SPDA). When pairs were in the Individual condition they were mainly working on different questions and different areas of the table (DPDA), and occasionally switched to writing answers down (VD). Pairs seldom spent time touching the interface in the same area while working on the same or different questions (SPSA, DPSA) except for learning purposes.

The key finding is that collaborative software visualization systems should support a flexible variety of coupling styles. This reinforces the findings from Tang et al. [320] as applied to collaborative software visualization.

Support fluid transitions between coupling and arrangement styles

We found the most frequently used arrangement styles by pairs were standing Side by Side, standing at Opposite Ends, and standing Apart (§8.4). The most time was spent in the Side by Side and Opposite Ends arrangement styles. When pairs were in the Group condition they were mainly Side by Side and regularly transitioned to Together but rarely to Kitty Corner. When pairs were in the Individual condition working on different questions they were mainly in Opposite Ends and sometimes End Side as these styles were the optimal locations for utilizing the screen real estate. Pairs frequently used more arrangement styles than spending time in each arrangement style. This was except Side by Side and Opposite Ends which were the dominant styles in the Group and Individual conditions respectively.

We compared coupling styles to arrangement styles. Our findings show that when pairs worked closely coupled they were more closely arranged, and when they worked loosely coupled they were more loosely arranged. When pairs were closely coupled they were arranged Side by Side, frequently transitioned to other closely coupled styles, and occasionally transitioned to one loosely coupled style

(VD). When pairs were loosely coupled working on different questions they were arranged at Opposite Ends of the table and less frequently transitioned to other loosely coupled styles.

The key finding is that collaborative software visualizations systems and multi-touch tables should support fluid transitions between both coupling styles and arrangement styles. This reinforces the findings from Tang et al. [320] and Isenberg et al. [141, 142] as applied to collaborative software visualization.

9.2.2 Designing Collaborative Software Visualizations

We wanted to find out from participants their perceptions of the strengths (§7.1) and weaknesses (§7.2) of the visualizations and to suggest improvements (§7.3).

Design visualizations for closely coupled arrangements with rotation features

The preferred coupling style was viewing engaged (VE) and arrangement style Side by Side. This meant one participant was mainly controlling the interface most of the time, the other participant viewing, and both participants standing next to each other on the longer side of the table. When pairs were working on parallel individual work, participants displayed visualizations in windows and manually rotated them to face their direction. This meant participants had the freedom to stand in different arrangements.

The key finding is collaborative software visualizations should primarily be designed to be viewed from a Side by Side arrangement with features to rotate the visualizations. When visualizations need to be viewed from different view points such as in parallel individual work there should be lightweight options for rotating visualization windows to face different directions. There should also be automatic options to perform the rotation operations. This reinforces the findings from Kruger et al. [180] as applied to collaborative software visualization.

Provide functionality in the appropriate locality.

When participants were displaying a visualization and wanted to launch a new overview visualization they they had to navigate to the start up screen rather than having a display on demand menu to launch a visualization. Participants, however, could launch new detailed visualizations through the on demand pie menu by tapping and holding elements in a visualization. When a visualization was at full screen and a participant were on one side of the table and wanted to select an option from a menu on the other side of the table they had to ask their colleague to perform the action for them.

The key finding is collaborative software visualizations should provide functionality in the appropriate locality. For example global functionality like menu options should be available from anywhere in the visualizations and irrespective of where users are physically located. Local functionality should only be available for a single user and close to where they are physically located such as in personal visualization windows. This reinforces the findings from Scott et al. [291] as applied to collaborative software visualization.

Provide consistent user interactions and visual interface design.

We found that the design of the user interactions and visual interface was not as consistent as we expected.

Some participants found the navigation gestures for zooming and panning performed differently across the suite of visualizations. Sometimes the navigation gestures did not perform consistently when more than one participant was interacting simultaneously.

Multiple visualization windows could be displayed at once. Participants found manipulating windows by dragging, resizing, or rotating them not that easy. Occasionally resizing a window to very small made the window disappear.

The pie menu allowed participants to easily switch between visualizations for packages and classes. When participants used the pie menus many participants tried to select a menu item by dragging their finger to a pie segment to perform gestural activation [191], but a separate touch was required.

At times some menus (pie menu and entity properties) were displayed at different zoom levels and occasionally displayed half off the screen. When elements are displayed on demand they need to be displayed at an appropriate zoom level that is consistent with other elements on display.

Many participants found entering text using the virtual keyboard difficult, as the keys did not perform as expected, especially the enter key. Participants suggested using a more familiar consistent physical keyboard to enter text.

This was the first time participants had used SourceVis before and there were many visualizations. We observed that participants occasionally lost context of which visualization they were currently looking at. Adding titles and breadcrumb navigation to the visualizations would aid users in understanding the context of the visualizations.

Some visualizations used charts to display information, which participants perceived to be very effective. Many visualizations utilized Polymetric encodings and colour coded classes by type. Some participants forgot what these visual encodings represented. Adding information such as a legend to the user interface

and visualizations should help users to remember what the visual encodings represent.

The key finding is collaborative software visualizations should provide consistent user interaction and visual interface design across all visualizations. Providing consistent gestural interactions and visual interface design will potentially increase the usability and improve the user experience of a collaborative software visualization system.

9.2.3 Designing Multi-touch Tables

We wanted to find out how the multi-touch table helped with team collaboration when participants completed the user tasks (§7.4).

Provide high resolution workspace.

We found that the resolution of the table impacted displaying multiple visualization windows next to each other. When pairs were at Opposite Ends and visualization windows were oriented to face each end, windows quite often overlapped each other. To prevent windows from overlapping, participants reduced the size of the windows with scale gestures. When pairs were Side by Side visualization windows were displayed over the top of each other which prevented seeing the visualization below.

We found that text was readable when visualizations were displayed at full screen. We found that reading text displayed in visualization windows was difficult when windows were reduced in size.

The key finding is multi-touch tables should provide a high resolution workspace. The resolution of our multi-touch table was limited by the output of the projector which was 1280x800 pixels. The resolution should be equal or greater than contemporary desktop computers. Providing a high resolution workspace would reduce windows overlapping and make text easier to read. This reinforces the findings from Tang et al. [320] as applied to collaborative software visualization.

Provide appropriate table space.

We found that when pairs were performing joint group work the physical size of the multi-touch table was appropriate. The table allowed participants to perform different roles when interacting. It was a seamless process for participants to coordinate swapping roles. Working together as a group made participants more aware of what each other was doing compared with working separately as individuals, and encouraged participants to communicate with each other.

We found that when pairs were performing parallel individual work the size and resolution of the table was not appropriate. This forced pairs to mainly physically be arranged at opposite ends of the table.

The key finding is multi-touch tables should provide appropriate table space. Depending on the task and number of users interacting there should be appropriate table space for joint group work or parallel individual work. The physical size should not prevent users from reaching all parts of the table from any side [331]. This reinforces the findings from Scott et al. [291].

Differentiate between simultaneous user interactions.

While the multi-touch features of the table allowed multiple users to interact at once, we observed that most of the time one participant was controlling the interface. When pairs were interacting simultaneously the system could not differentiate between the touch points of the participants. When pairs performed simultaneous navigation gestures on the canvas of a visualization or manipulated the same element within a visualization the system was confused as to what action to perform, so the navigation gestures and manipulation of elements did not perform as the pairs expected. These issues were partly due to the multi-touch table suffering from an inconsistent touch experience, and on occasions there were problems with the precision and accuracy of detecting touch points.

The key finding is multi-touch tables should differentiate between simultaneous user interactions to detect different users interacting with the system. Differentiating between user interactions will offer a better user experience for multi-user multi-touch table applications. Many approaches have tried to differentiate between multiple users touching a multi-touch table simultaneously by shadows of the arms of users [377], shoes of users [276], fingerprints [129], skin sensing [274], and assigning users to specific physical seats [79].

9.3 Limitations

We now discuss the limitations of our professional user study (§6) with respect to the qualitative findings (§7) and quantitative findings (§8). These limitations relate to the participants, procedure, and apparatus used in the study.

9.3.1 Participants

We recruited 44 participants who worked in 22 pairs. 41 of the participants were male. We would have liked to have recruited more diverse participants. Our participants ranged in age from 18-45+. 60% of our participants were intermediate developers or higher and had more than 5 years of software development experience. We would have liked to have recruited exclusively intermediate developers and higher as they would have more experience. Given how time consuming it was to recruit the participants, and that the study was partly qualitative, we felt we had enough participants. If we were to do a purely quantitative comparative study with SourceVis and another software visualization tool we would hope to recruit more participants.

Most participants were recruited from Wellington. We could have recruited from more locations but that would have been more expensive. We could have conducted the user study in another city but we felt that this would be too time consuming and again more expensive to run.

This was the first time participants had used SourceVis and a large multi-touch table before, hence they were all novice users. Most participants had limited experience with software visualization tools, although some participants had experience with the Gource [59] software evolution visualization tool as it was used within their organisation.

9.3.2 Procedure

The user study had limited time in a fixed location. We gave participants only 15 minutes to explore SourceVis with training systems before conducting the user study. One participant explicitly stated that the study was not enough time to learn SourceVis in full, which is what we expected. The user study was in a lab, and not in the participants' familiar working environment. We asked participants to think aloud which may have changed the way they performed. In the future we would like to conduct field studies over a longer period of time to allow participants to become more familiar with SourceVis and to observe how they perform.

The questions we asked in the study were reflective of the types of questions developers ask about software within industry [101, 170, 333, 334, 302]. The ques-

tions, however, may not necessarily have been the types of questions participants would ask about their own software. The user tasks did not consider the following maintenance tasks: debugging of code, adding in new features to a system, or performing a refactoring as these tasks were not currently supported by SourceVis. Instead the questions we asked were to do with analysis and identifying certain aspects of a system such as large packages and classes. We envisaged that developers would then use other tools like IDEs to perform these aforementioned tasks.

This was a study on understanding software systems participants' may not have been familiar with. We did not confirm if any of the participants were familiar with or had made any contributions to the software systems used in the study. Deploying SourceVis on software developed by participants' and letting them use SourceVis outside a lab study would help them to decide what to look for within a system and determine which visualizations to utilize.

We forced pairs to work in certain conditions either as groups or as individuals. If we had not forced the pairs to works as individuals it is likely they would have completed the user study working as a group as that was the preferred option. Forcing the pairs to work in certain conditions may have affected which coupling and arrangement styles they used.

9.3.3 Apparatus

The multi-touch table suffered from inconsistent touch detection at times. Some participants had problems with the touch detection while others didn't. For participants that did have problems with the touch detection this affected the way they interacted with the table and lead a few pairs to take turns when interacting with the table. Some participants felt the resolution of the table was too low compared with contemporary desktop computer screens. We would like to explore commercial multi-touch tables such as the Microsoft PixelSense[1] table to see if they would alleviate the touch detection, low resolution, and performance issues.

Due to the design of our multi-touch table, we only used three sides of the table because the fourth side was used for the video and audio equipment. If we were to run a similar study again we would like to use a table that allowed all four sides to be utilized.

[1] http://www.microsoft.com/en-us/pixelsense/default.aspx

9.4 Future Work

As part of the post-study questionnaire (§6.3) we asked participants for any further feedback and followed up with a post-interview as the last activity in the study. We would like to implement new software visualizations. We would like to apply a framework for visual information analysis to see how the participants engaged with the visualizations. We would like to conduct further user studies, to compare SourceVis against existing software exploration tools and an ethnographic field study of SourceVis being used by developers in their workplace environment.

9.4.1 New Software Visualizations

Our evolution visualizations focused on structural changes between different versions of a system. Many of the participants were interested with tracking how many contributions they had made on a software development project. We would like to create visualizations of code contributions by developers on a project based on commits to version control systems. We would envisage implementing something similar to Gource [59] or Code Swarm [229] for multi-touch tables.

SourceVis was developed for multi-touch tables. Some participants suggested that a vertical display may fit better with a work environment as software development teams are familiar with whiteboards for visualizing work flow.

We conducted a user study of software visualization prototypes using a large visualization wall but at the time there were issues with the hardware and running software applications [10]. We would like to adapt SourceVis to our large visualization wall since our visualization wall is now more mature and better supports client applications.

We did not compare a horizontal tabletop to a vertical display wall. We would like to conduct a quantitative study with SourceVis between the multi-touch table and visualization wall to see how the physical orientation of the display affects collaborative software visualization and if there are any advantages to using a large visualization wall.

Working as individuals on certain development tasks such as programming is critical for the success of a project. Given our study concerned parallel individual work we would also like to explore personal displays as well as shared surfaces within a multi-surface environment for software analysis tasks.

9.4.2 Apply Visual Information Analysis Framework

We would like to perform further data analysis of how the pairs in our professional user study engaged with our visualizations during their information analysis process to answer the questions. We would like to apply the visual information analysis framework from Isenberg et al [143]. The aim of this framework is to help inform the design of collaborative information visualization applications. The framework involves eight processes: Browse, Parse, Discuss Collaboration Style, Establish Task Strategy, Clarify, Select, Operate, and Validate. We would like to see if these processes are the same for collaborative software visualization and to discover any new ones.

9.4.3 Evaluation of Collaborative Software Visualization

Wettel et al. [362] conducted a quantitative comparative user experiment with their software visualization tool CodeCity which utilizes Polymetric View techniques against two state of the art exploration tools (Eclipse and a spreadsheet of metrics data). The experiment involved only single users. The results of the study validated that CodeCity outperformed Eclipse and the metrics data in both correctness and completion times. We would like to conduct a similar controlled quantitative comparative user experiment between SourceVis, state of the art software exploration tools (e.g. Eclipse and metrics data), and CodeCity to see if pairs of developers using SourceVis outperform these tools.

Our professional user study was in a controlled lab (§6). In the future we would like to conduct ethnographic field studies of SourceVis in a workplace environment similar to the field study conducted with IMPROMPTU [34]. We would like to deploy SourceVis and our multi-touch table to see how teams of professional software developers would use them over a set period of time.

9.5 Summary

Most software visualization systems and tools are designed from a single-user perspective and are bound to the desktop, IDEs, and the web. These design decisions do not allow users to collaboratively analyse software or easily interact and navigate visualizations within a co-located environment at the same time.

This thesis has presented an exploratory study of collaborative software visualization using multi-touch tables in a co-located environment. The thesis contributes a richer understanding of how pairs of developers make use of shared visualizations on large interactive multi-touch tables to gain insight into how software systems are structured and how they have evolved over different versions.

We designed a collaborative software visualization application, called SourceVis, that contained a suite of 13 visualization techniques adapted for multi-touch interaction. We built two large multi-touch tables (28 and 48 inches) following existing hardware designs, to explore and evaluate SourceVis. We then conducted both qualitative and quantitative user studies, culminating in a study of 44 professional software developers working in pairs.

We found that pairs preferred joint group work, used a variety of coupling styles, and made many transitions between coupling and arrangement styles. For collaborative group work we recommend designing for joint group work over parallel individual work, supporting a flexible variety of coupling styles, and supporting fluid transitions between coupling and arrangement styles.

We found that the preferred style for joint group work was closely coupled and arranged side by side. We found some global functionally was not easily accessible. We found some of the user interactions and visual interface were not designed consistently. For the design of collaborative software visualizations we recommend designing visualizations for closely coupled arrangements with rotation features, providing functionality in the appropriate locality, and providing consistent user interactions and visual interface design.

We found sometimes visualization windows overlapped each other and text was hard to read in windows. We found when pairs were performing joint group work the size of the table was appropriate but not for parallel individual. We found that because the table could not differentiate between different simultaneous users that some pair interactions were limited. For the design of multi-touch tables we recommend providing a high resolution workspace, providing appropriate table space, and differentiating between simultaneous user interactions.

We plan to improve SourceVis by creating new visualizations, perform further data analysis of how the pairs engaged with our visualizations, and conduct quantitative comparative user experiments between SourceVis and Eclipse.

Part VI

Appendices

Appendix A

Human Ethics Approval

The following documents are our approved human ethics application for conducting user studies on software visualization tools at Victoria University of Wellington, New Zealand.

VICTORIA UNIVERSITY OF WELLINGTON
Te Whare Wananga o te Upoko o te Ika a Maui

HUMAN ETHICS COMMITTEE
Application for Approval of Research Projects

Please write legibly or type if possible. **Applications must be signed by supervisor (for student projects) and Head of School**

Note: The Human Ethics Committee attempts to have all applications approved within three weeks but a longer period may be necessary if applications require substantial revision.

1 NATURE OF PROPOSED RESEARCH:

(a) Staff Research ☐	Student Research ☒	(tick one)
(b) If Student Research	Degree PhD Thesis	Course Code **COMP690**
(c) Project Title: Visual Software Analytics - User studies of software visualisation tools		

2 INVESTIGATORS:

(a) Principal Investigator	
Name	Craig Anslow
e-mail address	craig@ecs.vuw.ac.nz
School/Dept/Group	Engineering and Computer Science

(b) Other Researchers Name	Position
Professor James Noble	Primary Supervisor
Dr. Stuart Marshall	Supervisor

(c) Supervisor (in the case of student research projects)	
Professor James Noble	

3 DURATION OF RESEARCH

(a) Proposed starting date for data collection once this ethics form has been approved
(Note: that NO part of the research requiring ethical approval may commence prior to approval being given)

(b) Proposed date of completion of project as a whole March 2011

4 PROPOSED SOURCE/S OF FUNDING AND OTHER ETHICAL CONSIDERATIONS

(a) Sources of funding for the project

Please indicate any ethical issues or conflicts of interest that may arise because of sources of funding e.g. restrictions on publication of results

- Telstra Clear Postgraduate Scholarship
- This project is part of a larger multi-University research colloboration. This project is being undertaken by Victoria University as part of the Software Process and Product Improvement (SPPI) project led by the University of Auckland. There are no ethical issues or conflicts of interest arising from this relationship. The SPPI project is funded by the Foundation for Research, Science and Technology (FRST). For further information see, https://wiki.auckland.ac.nz/display/csisppi

(b) Is any professional code of ethics to be followed **Y** ☐ **N** ☒

If yes, **name**

(c) Is ethical approval required from any other body **Y** ☐ **N** ☒

If yes, name and indicate when/if approval will be given

5 DETAILS OF PROJECT

Briefly Outline:

(a) The objectives of the project

The project objective is to understand how software developers use software visualisation tools in their daily work when developing new software and maintaining existing software. This will inform future research into developing better tools to support software visualisation.

b) Method of data collection

We will conduct user studies of developers using software visualisation tools and prototypes. The software used with the tools and prototypes will come from free and open-source software (FOSS) that is freely accessible over the Internet and software from companies that choose to participate in this research. We will collect data from digital and audio recordings, user actions captured using log files, and eye tracking recordings. We will also make observations ourselves when participants are using the software visualisation tools.

(c) The benefits and scientific value of the project

Our user studies will expose how our subjects actually use the software visualisation tools, how software languages have been used, what features of languages are used, and better inform programming pedagogy, software language design, and software understanding.

(d) Characteristics of the participants

Participants will be rofessional or student software developers and other researchers.

We are not sure at this stage of exact numbers as we will be conducting several software visualization usability studies in the course of building our own tools and testing with others.

Ideally we would hope to test our software tools with users from somewhere up to 50 developers throughout the development stages of building our tool.

(e) Method of recruitment

Student software developers will be recruited via our school mailing lists. Software developers will be recruited through our software visualisation research survey and New Zealand companies who have agreed to participate in our joint Software Product and Process Improvement (SPPI) Project.

(f) Payments that are to be made/expenses to be reimbursed to participants

None

(g) Other assistance (e.g. meals, transport) that is to be given to participants

None

(h) Any special hazards and/or inconvenience (including deception) that participants will encounter

none

(i) State whether consent is for:

(i)	the collection of data	**Y** ☒	**N** ☐
(ii)	attribution of opinions or information	**Y** ☐	**N** ☒
(iii)	release of data to others	**Y** ☒	**N** ☐
(iv)	use for a conference report or a publication	**Y** ☒	**N** ☐
(v)	use for some particular purpose (specify)	**Y** ☒	**N** ☐

- user testing of software visualisation tools.

- The data will be shared with my supervisors and other investigators working on the SPPI project, including:

Dr. Ewan Tempero, Professor John Grundy, and Professor John Hosking (University of Auckland)
Dr. Jens Dietrich (Massey University)
Dr. Neville Churcher (University of Canterbury)

Attach a copy of any questionnaire or interview schedule to the application

(j) How is informed consent to be obtained (see sections 4.1, 4.5(d) and 4.8(g) of the Human Ethics Policy)

(i) the research is strictly anonymous, an information sheet is supplied and informed consent is implied by voluntary participation in filling out a questionnaire for example (include a copy of the information sheet) **Y** ☐ **N** ☒

(ii) the research is not anonymous but is confidential and informed consent will be obtained through a signed consent form (include a copy of the consent form and information sheet) **Y** ☒ **N** ☐

(iii) the research is neither anonymous or confidential and informed consent will be obtained through a signed consent form (include a copy of the consent form and information sheet) **Y** ☐ **N** ☒

(iv) informed consent will be obtained by some other method (please specify and provide details) **Y** ☐ **N** ☒

With the exception of anonymous research as in (i), if it is proposed that written consent will not be obtained, please explain why

(k) If the research will not be conducted on a strictly anonymous basis state how issues of confidentiality of participants are to be ensured if this is intended. (See section 4..1(e) of the Human Ethics Policy). (e.g. who will listen to tapes, see questionnaires or have access to data). Please ensure that you distinguish clearly between anonymity and confidentiality. Indicate which of these are applicable.

(i) access to the research data will be restricted to the investigator **Y** ☐ **N** ☒

(ii) access to the research data will be restricted to the investigator and their supervisor (student research) **Y** ☐ **N** ☒

(iii) all opinions and data will be reported in aggregated form in such a way that individual persons or organisations are not identifiable **Y** ☒ **N** ☐

(iv) Other (please specify)

This project is part of a larger multi-University research being done by the investigators in colloboration with Dr Ewan Tempero, Dr John Grundy, and Dr John Hosking of the University of Auckland, Dr Neville Churcher of the University of Canterbury, and Dr Jens Dietrich of Massey University. We will ask for consent that these five academics may also have access to the raw data.

(l) Procedure for the storage of, access to and disposal of data, both during and at the conclusion of the research. (see section 4.12 of the Human Ethics Policy). Indicate which are applicable:

(i) all written material (questionnaires, interview notes, etc) will be kept in a locked file and access is restricted to the investigator **Y** ☒ **N** ☐

(ii) all electronic information will be kept in a password-protected file and access will be restricted to the investigator **Y** ☒ **N** ☐

(iii) all questionnaires, interview notes and similar materials will be destroyed:

(a) at the conclusion of the research **Y** ☐ **N** ☒

or (b) 3 years after the conclusion of the research **Y** ☒ **N** ☐

(iv) any audio or video recordings will be returned to participants and/or electronically wiped **Y** ☒ **N** ☐

(v) other procedures (please specify):

If data and material are not to be destroyed please indicate why and the procedures envisaged for ongoing storage and security

(m) Feedback procedures (See section 7 of Appendix 1 of the Human Ethics Policy). You should indicate whether feedback will be provided to participants and in what form. If feedback will not be given, indicate the reasons why.

The research will be made available online and all participants will be given options to where they can locate the published papers / technical reports.

(n) Reporting and publication of results. Please indicate which of the following are appropriate. The proposed form of publications should be indicated on the information sheet and/or consent form.

(i) publication in academic or professional journals **Y** ☒ **N** ☐
(ii) dissemination at academic or professional conferences **Y** ☒ **N** ☐
(iii) deposit of the research paper or thesis in the University Library (student research) **Y** ☒ **N** ☐
(iv) other (please specify)

The results of the user studies will contribute towards a PhD thesis.

Signature of investigators as listed on page 1 **(including supervisors) and Head of School.**

NB: <u>All investigators and the Head of School must sign before an application is submitted for approval</u>

	Date	
Craig Anslow (PhD Student)	Date	
Professor James Noble and Dr. Stuart Marshall	Date	

Head of School:

Professor John Hine	Date	

TE WHARE WĀNANGA O TE ŪPOKO O TE IKA A MĀUI
VICTORIA
UNIVERSITY OF WELLINGTON

Phone 0-4-463 5676
Fax 0-4-463 5209
Email Allison.kirkman@vuw.ac.nz

MEMORANDUM

TO	Craig Anslow
COPY TO	Dr Stuart Marshall, Professor James Noble, Supervisors
FROM	Dr Allison Kirkman, Convener, Human Ethics Committee

DATE	January 16, 2009
PAGES	1

SUBJECT	**Ethics Approval: No 16262, Visual Software Analytics - User studies of software visualization tools**

Thank you for your application for ethical approval, which has now been considered by the Standing Committee of the Human Ethics Committee.

Your application has been approved from the above date and this approval continues until 30 March 2011. If your data collection is not completed by this date you should apply to the Human Ethics Committee for an extension to this approval.

Best wishes with the research.

Allison Kirkman
Convener

Appendix B

User Study Information and Consent Forms

The following documents are information and consent forms for participants to read and consent to participate in our user studies.

TE WHARE WĀNANGA O TE ŪPOKO O TE IKA A MĀUI
VICTORIA
UNIVERSITY OF WELLINGTON

School of Engineering and Computer Science

User Studies of Software Visualization Tools
Information Sheet

Introduction

This study contributes towards the overall completion of a PhD. The topic of the PhD is Multi-touch Table User Interfaces for Collaborative Software Visualization. Craig Anslow from the School of Engineering and Computer Science is conducting this study. The purpose of this particular study is to better understand how developers use collaborative software visualization techniques and tools when developing and analyzing software.

An enhanced understanding of the use of visualization techniques and tools will facilitate research into building better software visualization tools and multi-touch user interfaces.

This study is part of larger multi-University research collaboration. This study is being undertaken by Victoria University of Wellington as part of the Software Process and Product Improvement (SPPI) project led by the University of Auckland. The Ministry of Science and Innovation (MSI) now called The Ministry of Business, Innovation, and Employment (MBIE) fund the SPPI project. For further information see, https://wiki.auckland.ac.nz/display/csisppi

The study will be conducted with the approval of the Victoria University of Wellington's Human Ethics Committee.

Participation

This study will consist of you interacting with a software visualization tool. We will video and audio record your activities and encourage you to “Think Aloud”. We will take photos of you during our study to illustrate how users interact with our visualizations. We will conduct a questionnaire and de-brief session at the end of the study regarding the use of the software visualization tool.

Your participation in the study will take about 90 minutes, though may take longer if agreed. Your participation is completely voluntary.

In terms of protecting your anonymity, you will be assigned a unique ID so that your identity will only be accessible to the research investigators. The unique identifiers will be used to aggregate your submissions but will in no way be used to identify you personally or your company. No names or other methods of identifying you in reports will be used. At no time will anyone be able to identify any of the participants by using any of the reported material.

All data disclosed to researchers in this project is confidential. Only researchers on this project will be analyzing the data. The researchers are neutral third parties and are not responsible for any evaluation of your work. The data will be used to improve future tools to support software visualization and multi-touch user interfaces. The data may also be presented in aggregate form in academic papers and presentations. All data will be kept secure and protected at all times in password-protected files on a secure server. The results of the study data will be kept for three years. At the end of this time electronic data files will be deleted.

Please do not hesitate to ask any questions you may have about this study at any time.

Withdraw

If you choose, you can withdraw from the study up to one month after your participation in the study. Please contact the principal investigator to withdraw.

Results

Aggregated results from the study will be published in academic journals, conferences, and technical reports. Those participants who are keen to see the overall results may supply their email address via the consent form.

The results of the user studies will also contribute towards a PhD thesis.

We would like to sincerely thank you for your participation in this user study.

Researchers

Principal Investigator
Craig Anslow
PhD Student
School of Engineering and Computer Science
Victoria University of Wellington
Email: craig@ecs.vuw.ac.nz

Investigator
Dr. Stuart Marshall
School of Engineering and Computer Science
Victoria University of Wellington
Phone: 463 6730
Email: stuart@ecs.vuw.ac.nz

Investigator
Professor James Noble
School of Engineering and Computer Science
Victoria University of Wellington
Phone: 463 6736
Email: kjx@ecs.vuw.ac.nz

Investigator
Professor Robert Biddle
Department of Computer Science
Carleton University Canada

Investigator Assistant
Roger Cliffe
School of Engineering and Computer Science
Victoria University of Wellington

School of Engineering and Computer Science

User Studies of Software Visualization Tools

Written Consent Form

I have been given and have understood an explanation of this research project and the confidentiality conditions. I have had an opportunity to ask questions and have had them answered to my satisfaction.

I agree to participate in a user study session and follow-up debrief session for the purpose of this research and resulting publications. I consent to the collection, recording, and use of observations of my behaviour, my perceptions, experiences, opinions, and information in this research.

I understand that I may withdraw from this study up to one month from today's date without explanation and that in that case no data relating to my participation will appear in the final results.

(Please circle)

Do you agree to have this session digitally video-recorded? YES NO

Do you agree to have this session and debrief session audio-recorded? YES NO

Do you agree to have one or more photographs of you taken during this session? YES NO

Do you agree to the data of this user study being released to other investigators working on this project? YES NO

Would you like to receive publications resulting from your involvement in this study? YES NO

If so, please provide and email address and/or telephone number at which you can be contacted to arrange sending you a copy of the publications in question:

_____

_____

Please sign below to indicate your agreement to the all of the above.

Date:

Participant ID:

Participant Name:

Signature: _____

Appendix C

User Study Recruitment Email

Subject: Request for Participants: Study on Tools for Co-Located Software Development Teams using large iPad like devices

Dear Sir/Madam,

I got your contact details from VicCareers at Victoria University if Wellington.

I was wondering if any software developers at your company XYZ would be interested in participating in my exciting research project? It would be fantastic to get some participants from your company!

Would anyone like to come and play with some big shiny interactive toys (e.g. large MS Surface / Apple iPad tablet like devices)?

My name is Craig Anslow and I am doing a PhD at Victoria University in software engineering within the School of Engineering and Computer Science. I am looking for some software developers (programming language agnostic), software architects, or experienced technical people with software development experience to come and test out my PhD research software tool. If people are interested or know someone who might be could you please let me know as that would be much appreciated. Below is a blurb about what I am looking for.

Most software development is conducted within teams and are quite often in co-located environments. The tools used by these teams, however, are single-user focused (e.g. Eclipse). We are researching ways to help improve software development teams collaborate and have developed a multi-user, multi-touch tool for teams to help analyse their software. We want to observe how teams could potentially use technology such as our tool for their own software development projects. If you would like to participate in the user study please contact us.

Details:
We require pairs of participants (software developers - programming language agnostic, software architects, or experienced technical people with software development experience) from the same

organisation who work in the same team and know each other to voluntarily participate in the study. The study will take up to 90 minutes. A small reward will be given for participation in the study.

When and Where:
Anytime participants are available, either during or after work hours or weekends, throughout October-December 2012. The study will take place in the School of Engineering and Computer Science at Victoria University of Wellington.

Contact Details:
Please contact Craig Anslow email craig@ecs.vuw.ac.nz or phone 04 463 9998 to arrange a time to participate in the user study.

About:
This research is being conducted by Victoria University of Wellington with Human Ethics Approval as part of the New Zealand Ministry of Science and Innovations Software Process and Product Improvement project.

No software from your organisation will be used. Instead a baseline of example open source software will be used in the study. We are looking at how participants use our tools and how they collaborate together as a proof of concept, not on how skilled participants are with our tools. All participants and organisation details will be anonymous in the reporting of the results.

The investigators on the project are Craig Anslow (PhD Student), Dr. Stuart Marshall, Professor James Noble from the School of Engineering and Computer Science at Victoria University of Wellington, New Zealand, and Professor Robert Biddle from the Department of Computer Science, at Carleton University, Canada.

More info:
`http://homepages.ecs.vuw.ac.nz/~craig/Site/Study.html`

Many thanks.

Kind regards,
Craig

Craig Anslow
PhD Thesis Student
School of Engineering and Computer Science
Victoria University of Wellington
New Zealand
http://homepages.ecs.vuw.ac.nz/ craig

Appendix D

User Study Questionnaires

The following are the pre-survey and post-survey questionnaires participants completed for our user studies.

SourceVis Pre-Study

Please complete the following pre-study questionnaire.
* Required

Participant ID *

Gender *
Are you Male or Female?

- Male
- Female

Age *
How old are you?

- 18-24
- 25-34
- 35-44
- 45+

Height *
How tall are you?

- 5ft 4in (162cm)
- 5ft 5in (165cm)
- 5ft 6in (167cm)
- 5ft 7in (170cm)
- 5ft 8in (172cm)
- 5ft 9in (175cm)
- 5ft 10in (177cm)
- 5ft 11in (180cm)
- 6ft (183cm)
- 6ft 1in (185cm)
- 6ft 2in (187cm)
- 6ft 3in (190cm)
- 6ft 4in (193cm)
- 6ft 5 in (195cm)
- greater than 6ft 5 in
- Other:

Highest Level of Education *
What is your highest level of Education?

- ○ High School
- ○ Bachelors
- ○ Honours
- ○ Masters
- ○ PhD
- ○ Other:

Subject of your degree *
If you have a degree what subject was it in? e.g. computer science, engineering, physics

Occupation *
What is your current occupation and/or job title? e.g. student, programmer, developer

Development Experience *
How long have you been developing software professionally?

- ○ Less than 2 years
- ○ 3-5 years
- ○ 5-10 years
- ○ 10-20 years
- ○ 20+ years

Programming Language Experience *
What programming languages do you develop in?

- ☐ PHP
- ☐ Perl
- ☐ Python
- ☐ Ruby
- ☐ Java
- ☐ C/C++
- ☐ C# / VB / .Net
- ☐ Scala
- ☐ Objective C
- ☐ JavaScript
- ☐ Other:

Team Collaboration - Personel *
How many years have you known your fellow colleague? If less than year type number of months

Team Collaboration - Location *
Is your team located co-located or distributed? Co-located is in the same building, distributed is people on your team working from different buildings and or different cities

- ○ Co-located
- ○ Distributed

Team Collaboration - Programming *
How often do you program with others on your software development team at the same time? e.g. Pair Programming

- ☐ Never
- ☐ Daily
- ☐ Weekly
- ☐ Monthly
- ☐ Other:

Team Collaboration - Meetings *
How often do you meet with others on your software development team at the same time? e.g. in the same room or video conferencing

- ☐ Never
- ☐ Hourly
- ☐ Daily
- ☐ Weekly
- ☐ Monthly
- ☐ Other:

Code Review Experience - By Developers *
How often do you review your code with other developers on your software development team?

- ☐ Never
- ☐ Hourly
- ☐ Daily
- ☐ Weekly
- ☐ Monthly
- ☐ Other:

Code Review Experience - By Management *

How often do you review your code with project managers or management within your company?

- [ ] Never
- [ ] Hourly
- [ ] Daily
- [ ] Weekly
- [ ] Monthly
- [ ] Other:

Code Review Experience - Tools *

What software tools / tools do you use for reviewing and analysing your code? e.g. meeting room with projector or large display screen, IDEs, bug tracking tools, code review tools, version control software tools

Software Visualization Tools Experience *

Have you used any software visualizations techniques or tools before? If so please list.

Mobile Phone - Touch *

What kind of mobile phone do you own / use that is **Touch** enabled?

- [ ] Apple iPhone
- [ ] Samsung
- [ ] Nokia
- [ ] RIM Blackberry
- [ ] Huawei
- [ ] Sony
- [ ] HTC
- [ ] None
- [ ] Other:

Touch Screens - Work *

Do you use touch screens for your daily work tasks? other than mobile phones but including tablets

- ○ Yes
- ○ No

Touch Screens - Personal *

Do you use touch screens for personal or home use? other than mobile phones but including tablets

- ○ Yes
- ○ No

Tablet - Touch *

What kind of tablet do you own / use?

- ☐ Apple iPad
- ☐ Samsung Galaxy
- ☐ Asus Nexus 7
- ☐ Microsoft Surface
- ☐ RIM Playbook
- ☐ Amazon Kindle
- ☐ Sony
- ☐ None
- ☐ Other:

Touch Screens - Experience

What kind of touch screens do you use on a regular basis (work or personal) other than mobile phones or tablets? Please list the names of the touch screens if known.

Submit

SourceVis Post-Study

Please complete the post-study questionnaire.
* Required

Participant ID *

Effectiveness *
Overall how effective do you find these software visualizations for answering the questions in the user study? (1 - poor, 10 effective)

	1	2	3	4	5	6	7	8	9	10	
Poor	○	○	○	○	○	○	○	○	○	○	Effective

Startup Screen *
Rate Effectiveness (1 - poor, 10 effective)

	1	2	3	4	5	6	7	8	9	10	
Poor	○	○	○	○	○	○	○	○	○	○	Effective

1. System Explorer *
Rate Effectiveness (1 - poor, 10 effective)

	1	2	3	4	5	6	7	8	9	10	
Poor	○	○	○	○	○	○	○	○	○	○	Effective

2. Metrics Explorer *
Rate Effectiveness (1 - poor, 10 effective)

	1	2	3	4	5	6	7	8	9	10	
Poor	○	○	○	○	○	○	○	○	○	○	Effective

3. Toxicity Chart *
Rate Effectiveness (1 - poor, 10 effective)

	1	2	3	4	5	6	7	8	9	10	
Poor	○	○	○	○	○	○	○	○	○	○	Effective

4. Vocabulary *
Rate Effectiveness (1 - poor, 10 effective)

1	2	3	4	5	6	7	8	9	10

	1	2	3	4	5	6	7	8	9	10	
Poor	○	○	○	○	○	○	○	○	○	○	Effective

5. System Hotspots View *
Rate Effectiveness (1 - poor, 10 effective)

	1	2	3	4	5	6	7	8	9	10	
Poor	○	○	○	○	○	○	○	○	○	○	Effective

6. System Dependency *
Rate Effectiveness (1 - poor, 10 effective)

	1	2	3	4	5	6	7	8	9	10	
Poor	○	○	○	○	○	○	○	○	○	○	Effective

7. Class Dependency *
Rate Effectiveness (1 - poor, 10 effective)

	1	2	3	4	5	6	7	8	9	10	
Poor	○	○	○	○	○	○	○	○	○	○	Effective

8. Class Blueprint *
Rate Effectiveness (1 - poor, 10 effective)

	1	2	3	4	5	6	7	8	9	10	
Poor	○	○	○	○	○	○	○	○	○	○	Effective

9. System Evolution *
Rate Effectiveness (1 - poor, 10 effective)

	1	2	3	4	5	6	7	8	9	10	
Poor	○	○	○	○	○	○	○	○	○	○	Effective

10. System Package Evolution *
Rate Effectiveness (1 - poor, 10 effective)

	1	2	3	4	5	6	7	8	9	10	
Poor	○	○	○	○	○	○	○	○	○	○	Effective

11. System Class Evolution *
Rate Effectiveness (1 - poor, 10 effective)

1 2 3 4 5 6 7 8 9 10

Poor	○	○	○	○	○	○	○	○	○	○	Effective

12. Individual Package Evolution *

Rate Effectiveness (1 - poor, 10 effective)

	1	2	3	4	5	6	7	8	9	10	
Poor	○	○	○	○	○	○	○	○	○	○	Effective

13. Individual Class Evolution *

Rate Effectiveness (1 - poor, 10 effective)

	1	2	3	4	5	6	7	8	9	10	
Poor	○	○	○	○	○	○	○	○	○	○	Effective

Strengths *

What were the strengths of these software visualizations? e.g. how did they help you answer the questions?

Weaknesses *

What were the weaknesses of these software visualizations?

Improvements *

What could be improved for these software visualizations?

Multi-touch Table - Team Collaboration *

How did the Multi-touch Table device help with team collaboration? What were the strengths and weaknesses?

Multi-touch Table - Work in Practice *

If you had access to a large multi-touch table for your work how would you envisage your team using a device of this nature? (select all that apply)

- [ ] planning - team development tasks
- [ ] meetings - face-face / video conferencing
- [ ] design - architecture / modelling
- [ ] implementation - programming / pair programming
- [ ] analysis - code review
- [ ] testing - debugging
- [ ] not at all
- [ ] Other:

Any other feedback?

Submit

Appendix E

Preliminary User Study 1 Questions

Imagine a scenario where you are working on a co-located software development team to maintain some existing software in Java. You have been asked to improve the software so that it is more easily maintainable by identifying places in which the code could be refactored (e.g. identifying large classes).

Please answer the following questions using SourceVis on the multi-touch table.

Word Cloud Visualization (in terms of font size)

- What are the 2 largest words in the packages word cloud? (java.awt, java.util)
- What are the 5 largest words in the classes word cloud? (conurrent, regex, pattern, (anon_1), image)

Wordle Visualization (in terms of font size)

- What is the largest word? (AttributeList)
- Name the three classes that start with the letter "B". (BindingType, Binding, Bounds)

Metrics Explorer

- How many classes does java.awt.font contain? (29)
- How many fields does java.awt.image.ColorModel contain? (13)

Class Blueprint

- How many accessor methods are called? (1, setHasUnsavedChanges)
- Which method refers to both the "file" and "prefs" attribute? (setFile)

System Hotspots View

- What is the biggest class? (java.awt.Component)
- How many methods does this class contain? (323)
- How many methods does java.awt.Window contain? (135)
- How many classes in java.util contain more than 1000 lines of code? (8)

Javadoc Web Browser

- How many classes does org.jhotdraw.samples.javadraw contain? (9)
- The package org.jhotdraw.samples.pert.PertFigure contains a class called PertFigure which has a method called read(). What kind of exception does read() throw? (java.io.IOException)

Appendix F

Preliminary User Study 2 Questions

Imagine a scenario where you are working on a co-located software development team to maintain some existing software in Java. You have been asked to improve the software so that it is more easily maintainable by identifying places in which the code could be refactored (e.g. identifying large classes, classes with many dependencies).

Please answer the following questions using the SourceVis application on the multi-touch table. First load the JHotDraw system by selecting the load menu option. Then answer the following questions about the JHotDraw System using the visualizations noted in the question. Some questions will involve using a different system, notably JUnit.

1. System Explorer Visualization
   In JHotDraw version 7.5.1, select package org.jhotdraw.geom, how many classes are there in package org.jhotdraw.geom?

2. Metrics Explorer Visualization
   What is the largest Package in JHotDraw version 6.0.1 (total metrics)

3. Metrics Explorer Visualization
   How many interfaces does the largest package in JHotDraw version 6.0.1 contain?

4. Metrics Explorer Visualization
   From that package what is the largest class in JHotDraw version 6.0.1? (total metrics)

5. Toxicity Chart Visualization
   In JHotDraw version 7.5.1 how many classes have a toxicity score greater than 5.0 for the metric value “File Length”?

6. Vocabulary Visualization
   In JHotDraw version 7.5.1 what are the four largest words used in “Class Names”?

7. Vocabulary Visualization
   In JHotDraw version 7.5.1 what are the two largest “Classes”?

8. Systems Hotspots View
   In JHotDraw version 7.5.1 what is the largest “Package”? (total metrics)

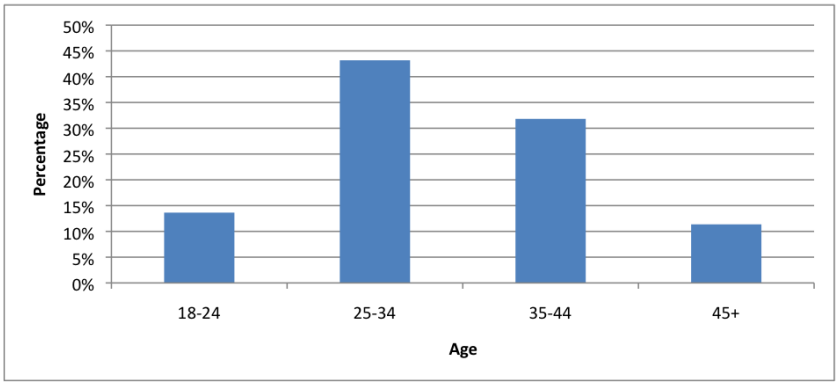
9. Systems Hotspots View
   In JHotDraw version 7.5.1 in the largest Package (same package as previous question), what is the largest “Class”? (total metrics)

10. Class Blueprint Visualization
    In JHotDraw version 7.5.1 in the largest Package and then largest Class (same class as previous question), how many acccessor methods are called by the public method "repaintHandles"?
11. Class Blueprint Visualization
    In JHotDraw version 7.5.1 in the largest Package and then largest Class (same class as previous question), how many interfaces does this class depend on?
12. Individual Class Evolution Visualization
    In JHotDraw version 7.5.1 in the largest Package and then largest Class (same class as previous question), how many versions does this class appear in?
13. System Dependency Visualization
    Load JUnit system. In JUnit 4.8.2 how many classes have no dependencies?
14. System Evolution Visualization
    In JUnit what major version contains the most amount of classes?
15. System Class Evolution Visualization
    In JUnit how many versions contain class junit.swingui.TestRunner?
16. System Class Evolution Visualization
    In JUnit how many versions contain "annotation classes"?

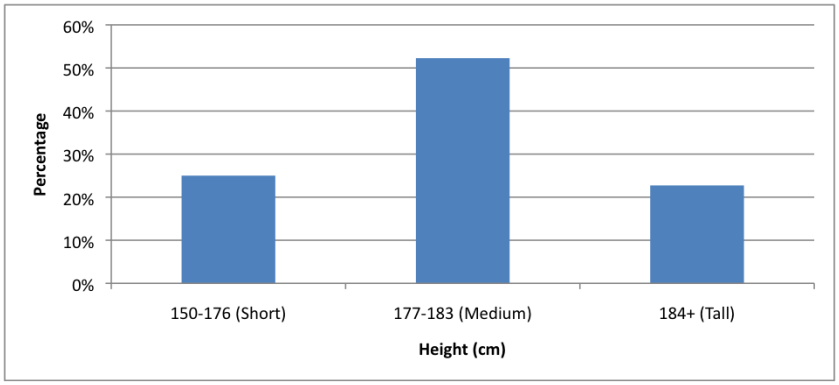
Appendix G

Professional User Study Participant Demographics

This appendix presents the demographics of the participants from the Professional User Study (§6) including participant demographics, participant's education, participant experience and skills, participant's devices, and participant's team details.

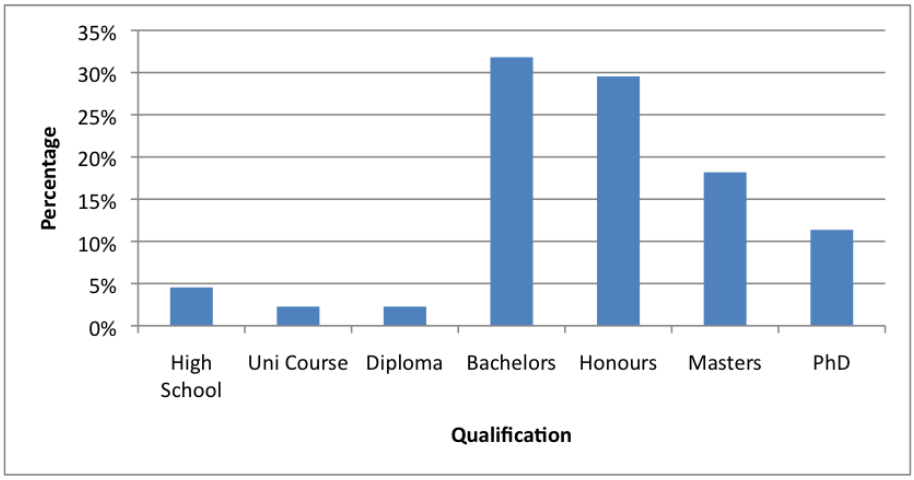


(a) Age.

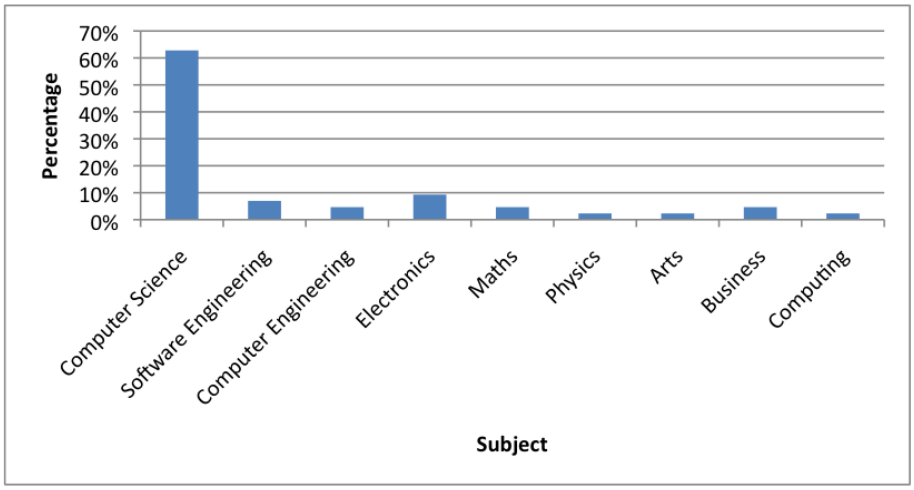


(b) Height.

Figure G.1: Participant Characteristics.

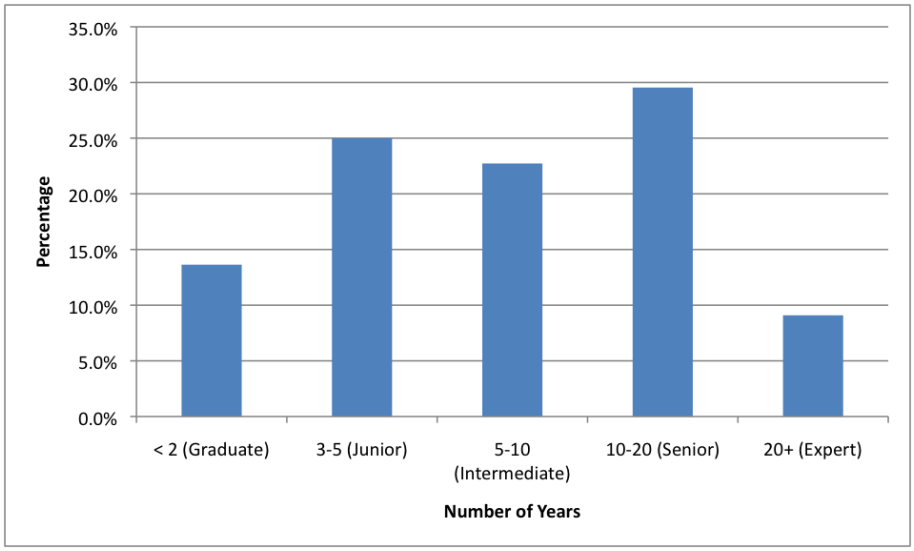


(a) Qualification.

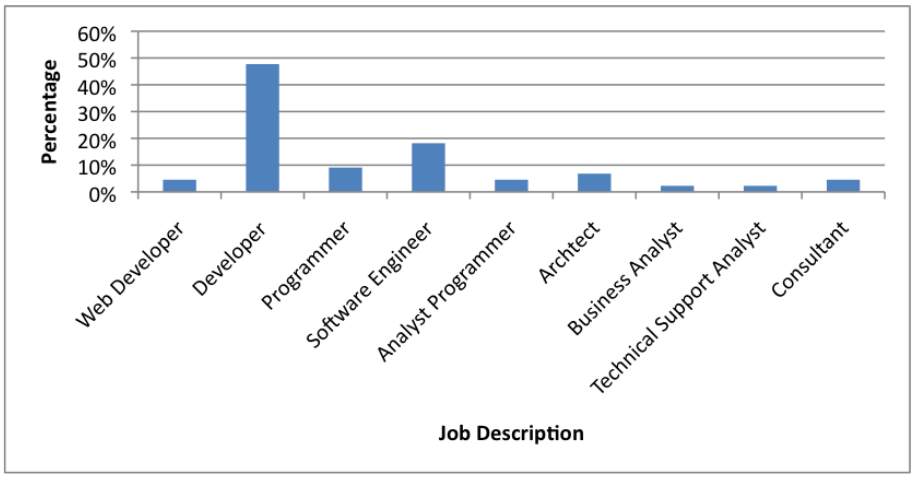


(b) Subject.

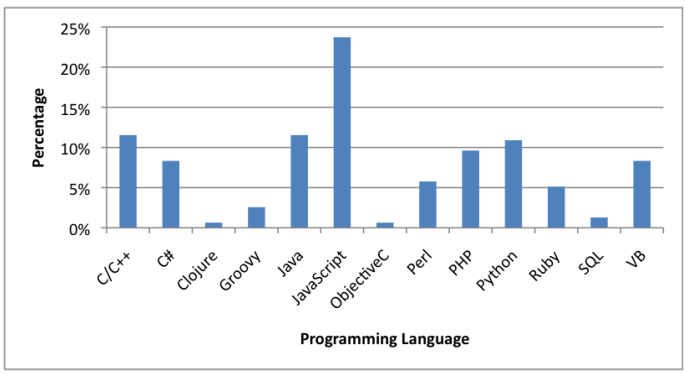
Figure G.2: Participant Education.



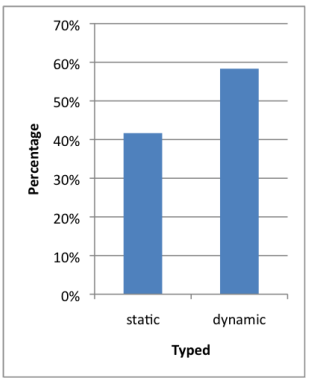
(a) Software development experience.



(b) Job Description.

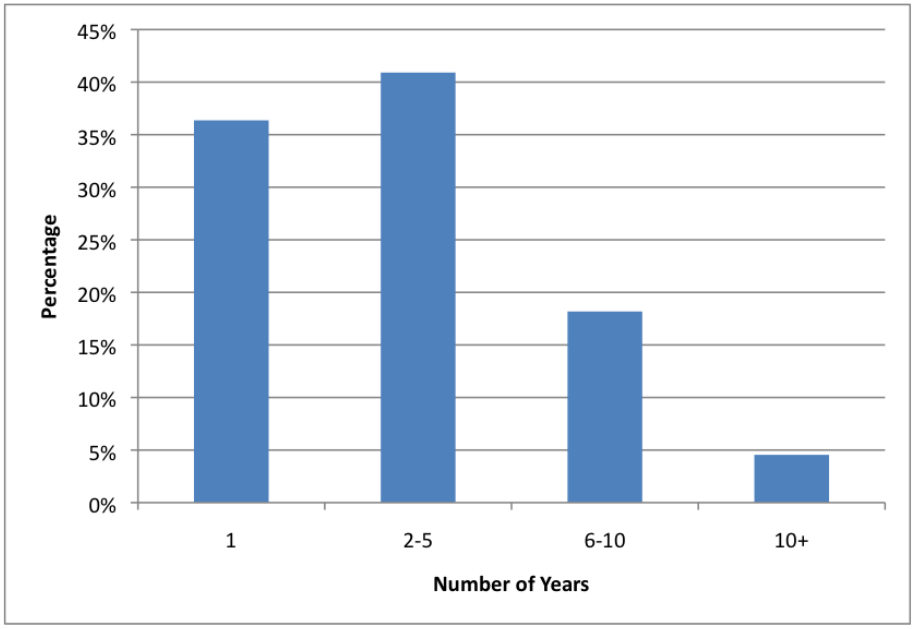


(c) Programming Languages.

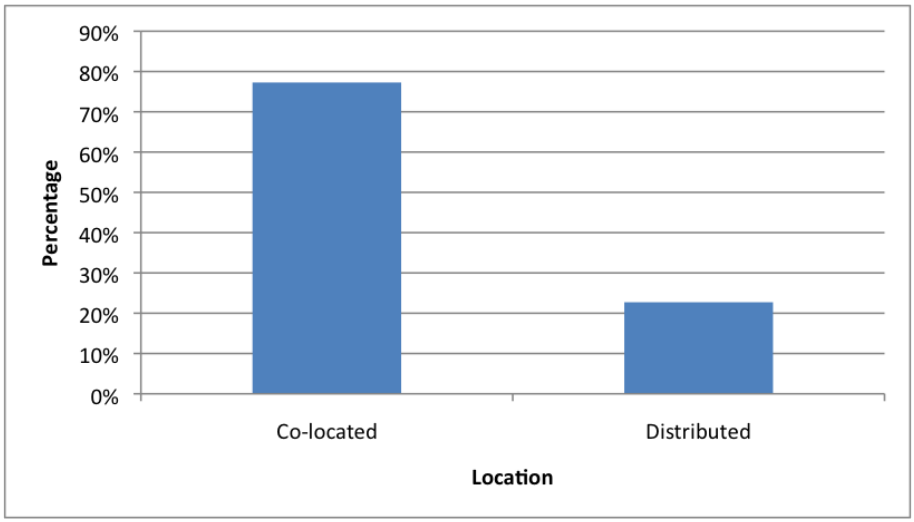


(d) Typed Programming Languages.

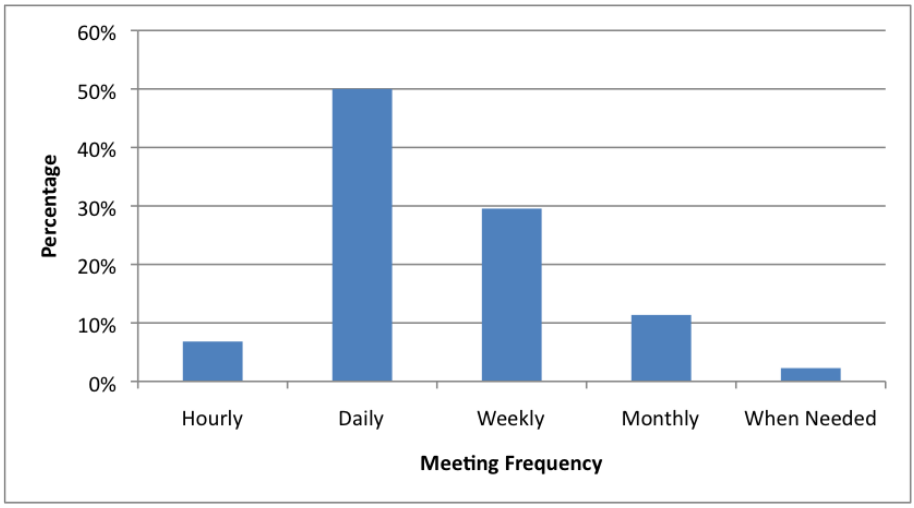
Figure G.3: Participant Experience and Skills.



(a) Number of years known colleague.

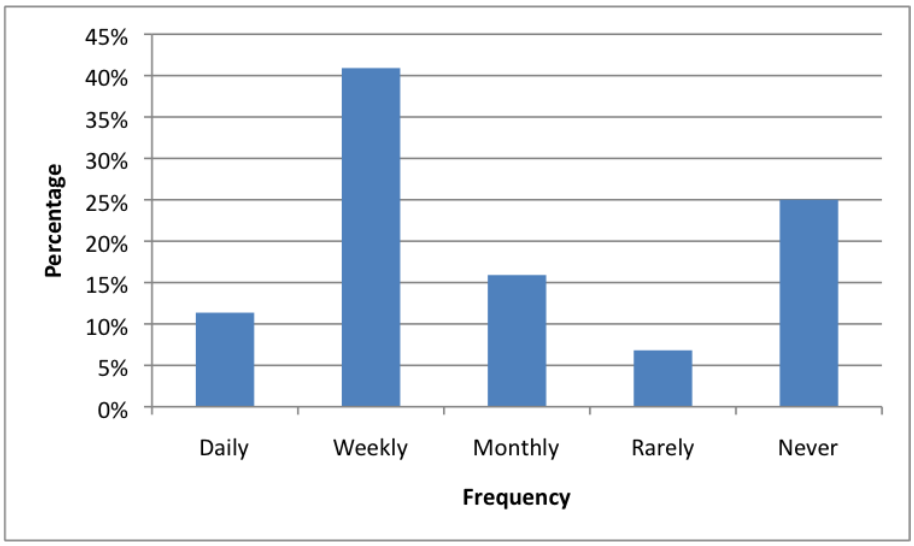


(b) Location.

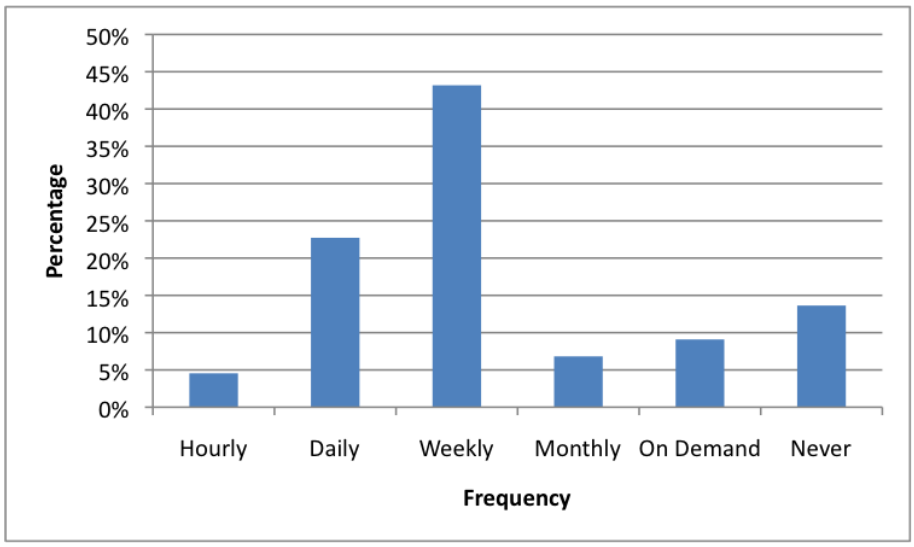


(c) Meetings.

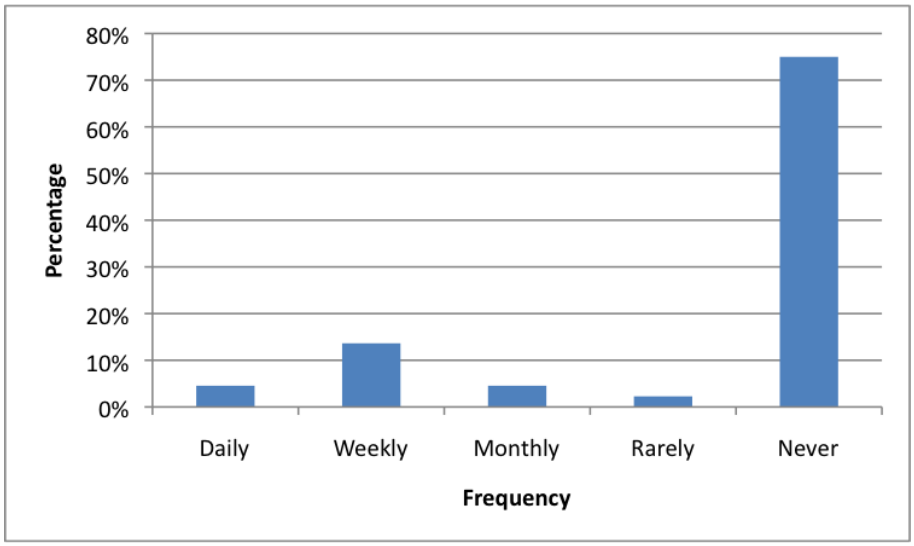
Figure G.4: Team Details.



(a) Collaborative Programming.

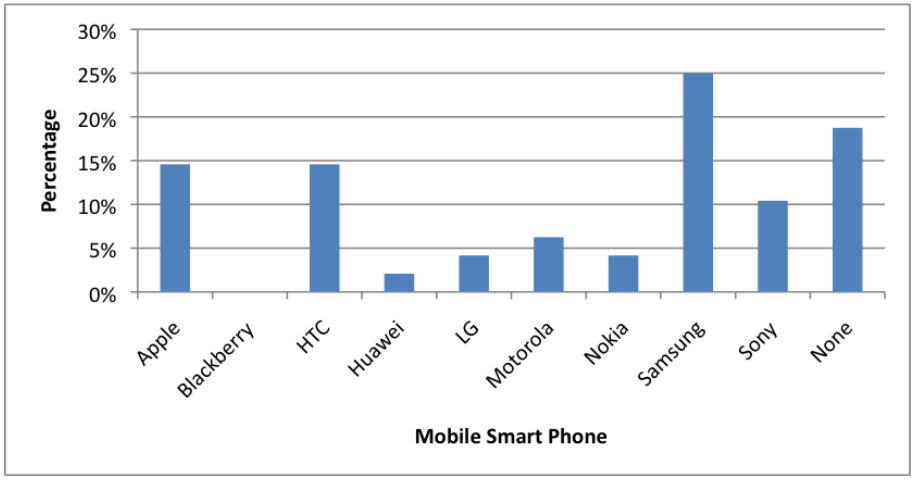


(b) Code Review by Developers.

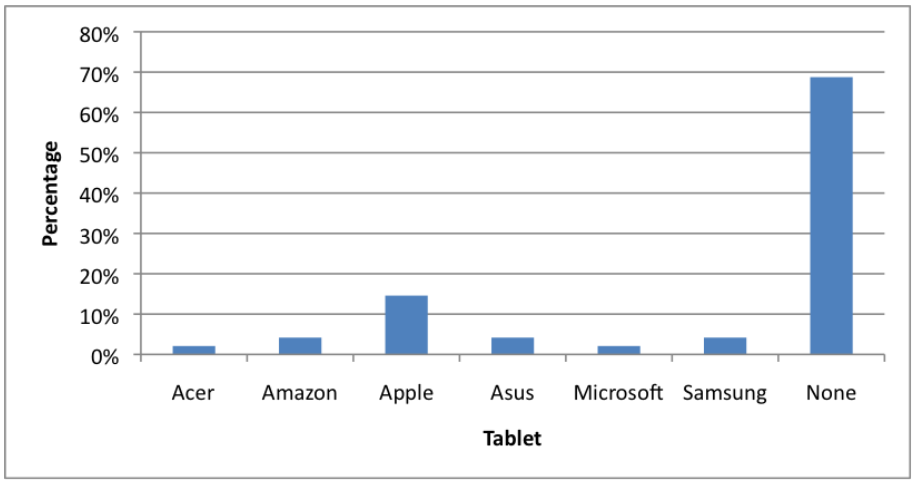


(c) Code Review by Management.

Figure G.5: Team Programming and Code Review Experience.



(a) Mobile Smart Phone.



(b) Tablet.

Figure G.6: Participant Devices.

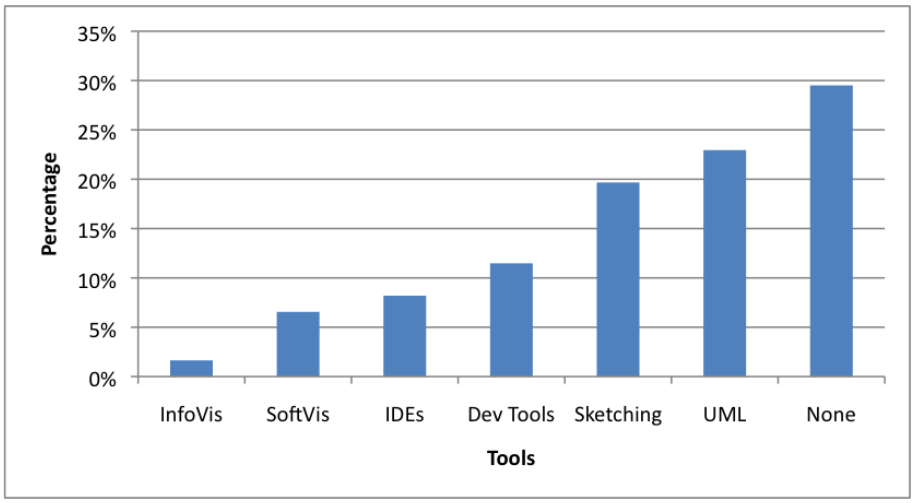


Figure G.7: Tools for Software Visualization.

Appendix H

Professional User Study Questions

H.1 Group Questions

Imagine a scenario where you are working on a co-located software development team to maintain some existing Java software system. You have been asked to improve the software so that it is more easily maintainable by identifying places in which the code could be refactored (e.g. identifying large classes, classes with many dependencies).

Please answer the following questions using the SourceVis application on the multi-touch table. First load the JHotDraw system by selecting the load menu option. Then answer the following questions about the JHotDraw System using the visualizations noted in the question.

1. System Explorer Visualization
   - Load JHotDraw System
   - Open the System Explorer Visualization
   - Select JHotDraw Version 7.5.1
   - Select package org.jhotdraw.geom

   How many classes are there in this package?
2. Metrics Explorer Visualization
   - Open the Metrics Explorer Visualization
   - Select JHotDraw version 6.0.1
   - Select All Metrics

   What is the largest package?
3. Metrics Explorer Visualization
   - Using the same Metrics Explorer Visualization
   - Using JHotDraw Version 6.0.1

   How many interfaces does the largest package from the previous question contain?
4. Vocabulary Visualization
   - Select JHotDraw Version 7.5.1

What are the two largest Classes?

5. Toxicity Chart Visualization
   - Open Toxicity Chart Visualization
   - Select JHotDraw Version 7.5.1

   How many classes have a toxicity score greater than 5.0 for the metric value File Length?
6. Systems Hotspots View
   - Load JHotDraw System
   - Select JHotDraw Version 7.5.1
   - Select All Metrics for packages

   What is the largest package (by all metrics)?
7. Class Blueprint Visualization
   - From the same Systems Hotspots Visualization
   - Using JHotDraw Version 7.5.1
   - In the largest package
   - Select the largest class (by all metrics)
   - Open Class Blueprint Visualization

   How many accessor methods are called by the public method repaintHandles()?
8. Class Dependency
   - From the same Systems Hotspots Visualization
   - Using JHotDraw Version 7.5.1
   - In the largest package
   - Select the largest class (by all metrics)
   - Open Class Dependency Visualization

   Which class does the largest class depend upon the most?
9. System Evolution Visualization
   - Open the System Evolution Visualization
   - Using JHotDraw

   Approximately how many lines of code has JHotDraw increased by between version 6.0.1 and version 7.5.1?
10. System Class Evolution Visualization
    - Open the System Class Evolution Visualization
    - Using JHotDraw

    How many versions does the largest class (by all metrics) in version 5.2.0 appear in?
11. System Dependency Visualization

- Load JHotDraw
- Open System Dependency Visualization
- Load JHotDraw Version 5.2.0

How many interfaces have no dependencies?

12. Class Dependency Visualization
    - From the same Systems Dependency Visualization
    - Using JHotDraw Version 5.2.0
    - Open Class Dependency Visualization for class CH.ifa.draw.application.DrawApplication

    How many interfaces does the class CH.ifa.draw.application.DrawApplication depend on?
13. Individual Class Evolution Visualization
    - From the same Systems Dependency Visualization
    - OR Class Dependency Visualization
    - Open Class Evolution Visualization

    Which version of the class CH.ifa.draw.application.DrawApplication is the largest (by all metrics)?
14. System Package Evolution Visualization
    - Open System Package Evolution Visualization

    What version has the most amount of packages?
15. Individual Package Evolution Visualization
    - From same System Package Evolution Visualization
    - Open Package Evolution for package CH.ifa.draw.standard

    How many versions contain the package named CH.ifa.draw.standard?

H.2 Individual A Questions

Imagine a scenario where you are working on a co-located software development team to maintain some existing Java software system. You have been asked to improve the software so that it is more easily maintainable by identifying places in which the code could be refactored (e.g. identifying large classes, classes with many dependencies).

Please answer the following questions using SourceVis application on the multi-touch table. First load the JUnit system by selecting the load menu option. Then answer the following questions about the JUnit System using the visualizations noted in the question.

1. System Explorer Visualization
    - Load Junit System
    - Open the System Explorer Visualization

- Select JUnit Version 4.8.2
- Select package junit.framework

How many classes are there in this package?

2. Vocabulary Visualization
   - Open the Vocabulary Visualization
   - Select Junit Version 4.8.2

   What are the five most frequently used words in classnames?
3. Vocabulary Visualization
   - Open the Vocabulary Visualization
   - Select Junit Version 4.8.2
   - Select Size of Packages

   What are the two largest Packages?
4. Metrics Explorer Visualization
   - Open the Metrics Explorer Visualization
   - Select Junit version 4.8.2
   - Select All Metrics

   In the largest package, how many concrete classes are there?
5. Toxicity Chart Visualization
   - Open Toxicity Chart Visualization
   - Select Junit Version 4.8.2

   Which class has a large file length?
6. Systems Hotspots View
   - Load Junit System
   - Open Systems Hotspots View
   - Select JUnit Version 4.8.2

   How many packages have the word "runner" in the package name?
7. Class Blueprint Visualization
   - From the same Systems Hotspots Visualization
   - Using Junit Version 4.8.2
   - In the org.junit.runners package
   - Select the class BlockJUnit4ClassRunner
   - Open Class Blueprint Visualization

   How many different accessor methods are called?

8. System Dependency Visualization

   - Open the System Dependency Visualization
   - Select JUnit Version 4.8.2
   - Use the slider

   What two classes have the highest dependency weight?

9. System Evolution Visualization

   - Open the System Evolution Visualization

   What major version of JUnit contains the most amount of classes?

10. System Package Evolution Visualization

    - Open the System Package Evolution Visualization

    How many versions does junit.framework appear in?

11. System Dependency Visualization

    - Load Junit System
    - Open System Dependency Visualization
    - Select JUnit Version 4.8.2

    How many concrete classes have no dependencies?

12. Class Dependency Visualization

    - From the same System Dependency Visualization
    - Select JUnit Version 4.8.2
    - Open the Class Dependency Visualization for org.junit.runners.BlockJUnit4ClassRunner

    What many abstract classes does this class depend on?

13. System Evolution Visualization

    - Open the System Evolution Visualization

    What major version of JUnit contains the most amount of classes?

14. System Package Evolution Visualization

    - Open the System Package Evolution Visualization

    After what version does junit.ui stop appearing in?

15. System Class Evolution Visualization

    - Open the System Class Evolution Visualization

    How many versions in JUnit contain annotation classes?

H.3 Individual B Questions

Imagine a scenario where you are working on a co-located software development team to maintain some existing Java software system. You have been asked to improve the software so that it is more easily maintainable by identifying places in which the code could be refactored (e.g. identifying large classes, classes with many dependencies).

Please answer the following questions using the SourceVis application on the multi-touch table. First load the JUnit system by selecting the load menu option. Then answer the following questions about the JUnit System using the visualizations noted in the question.

1. Metrics Explorer Visualization

   - Load Junit System
   - Open the Metrics Explorer Visualization
   - Select Junit version 4.8.2
   - Select All Metrics

   What is the largest package?

2. Metrics Explorer Visualization

   - Open the Metrics Explorer Visualization
   - Select Junit version 4.8.2
   - Use Keyboard search and then sort

   How many packages have the word ”runner” in the package name?

3. Toxicity Chart Visualization

   - Open Toxicity Chart Visualization
   - Select Junit Version 4.8.2

   How many classes have a toxicity score greater than 2.0?

4. Toxicity Chart Visualization

   - Open Toxicity Chart Visualization

   Since version 4.0 (including) of Junit how many versions have number of classes greater than 60 with a toxicity score of 1.0?

5. Vocabulary Visualization

   - Open the Vocabulary Visualization
   - Select Junit Version 4.8.2

   What are the three largest classes?

6. System Dependency Visualization

   - Load Junit System
   - Open System Dependency Visualization
   - Select JUnit Version 4.8.2

- Select no dependencies

How many concrete classes have no dependencies?

7. Class Dependency Visualization
   - From the same System Dependency Visualization
   - Select JUnit Version 4.8.2
   - Open the Class Dependency Visualization for org.junit.runners.BlockJUnit4ClassRunner

   How many abstract classes does this class depend on?
8. System Evolution Visualization
   - Open the System Evolution Visualization

   What major version of JUnit contains the most amount of classes?
9. System Package Evolution Visualization
   - Open the System Package Evolution Visualization

   After what version does junit.ui stop appearing in?
10. System Class Evolution Visualization
    - Open the System Class Evolution Visualization

    How many versions in JUnit contain annotation classes?
11. Systems Hotspots View
    - Load Junit System
    - Open Systems Hotspots View
    - Select JUnit Version 4.8.2

    How many packages have the word "runner" in the package name?
12. Class Blueprint Visualization
    - From the same Systems Hotspots Visualization
    - Using Junit Version 4.8.2
    - In the org.junit.runners package
    - Select the class BlockJUnit4ClassRunner
    - Open Class Blueprint Visualization

    How many different accessor methods are called?
13. System Dependency Visualization
    - Open the System Dependency Visualization
    - Select JUnit Version 4.8.2

    What two classes have the highest dependency weight?

14. System Evolution Visualization

    - Open the System Evolution Visualization

    What major version of JUnit contains the most amount of classes?

15. System Package Evolution Visualization

    - Open the System Package Evolution Visualization

    How many versions does junit.framework appear in?

Appendix I

Quantitative Findings - Additional Tables

This Appendix shows additional tables for Coupling Styles, Arrangement Styles, and Wilcoxon Rank Sum Tests that were performed for Chapter 8:

- Frequency of Coupling Styles (Table I.1).
- Amount of Time spent in Coupling Styles (Table I.2).
- Frequency of Arrangement Styles (Table I.3).
- Amount of Time Spent in Arrangement Styles (Table I.4).
- Wilcoxon Rank Sum Test for Frequency of Coupling Styles (Table I.5).
- Wilcoxon Rank Sum Test for Time Spent in Coupling Styles (Table I.6).
- Wilcoxon Rank Sum Test for Frequency of Arrangement Styles (Table I.7).
- Wilcoxon Rank Sum Test for Time Spent in Arrangement Styles (Table I.8).

Table I.1: Frequency of Coupling Styles by each pair. Each row is separated into Group (G), Individual (I), and Total amount of time (Tot) for each style. GIG = grey shaded rows and IGG = white rows.

Style:	**DISC**			**SPSA**			**SPDA**			**VE**			**VD**			**DPSA**			**DPDA**			
Pair	G	I	Tot	G	I	Tot	G	I	Tot	G	I	Tot	G	I	Tot	G	I	Tot	G	I	Tot	**SUM**
1	30	2	32	2	0	2	6	1	7	33	10	43	6	10	16	0	0	0	0	10	10	110
2	52	1	53	19	0	19	29	0	29	78	5	83	11	17	28	0	4	4	0	19	19	235
3	23	11	34	22	7	29	7	1	8	39	22	61	10	12	22	0	3	3	0	12	12	169
4	36	2	38	12	0	12	21	0	21	50	2	52	3	13	16	0	1	1	0	15	15	155
5	38	11	49	5	5	10	6	6	12	38	21	59	10	10	20	0	1	1	0	9	9	160
6	30	2	32	6	2	8	16	3	19	48	6	54	11	12	23	0	2	2	1	11	12	150
7	31	2	33	12	0	12	7	0	7	27	1	28	16	14	30	0	3	3	0	20	20	133
8	39	12	51	2	1	3	15	3	18	56	17	73	6	11	17	0	1	1	0	6	6	169
9	38	1	39	3	0	3	5	1	6	34	2	36	12	8	20	0	0	0	0	9	9	113
10	43	0	43	5	0	5	13	0	13	43	1	44	7	11	18	0	0	0	0	10	10	133
11	44	7	51	4	0	4	11	0	11	47	5	52	10	12	22	0	0	0	0	10	10	150
12	36	2	38	5	0	5	9	0	9	50	4	54	10	9	19	0	2	2	0	10	10	137
13	28	1	29	4	1	5	6	4	10	36	6	42	11	12	23	0	0	0	0	10	10	119
14	20	2	22	6	0	6	3	0	3	34	2	36	9	8	17	0	0	0	1	10	11	95
15	30	1	31	1	0	1	2	0	2	37	1	38	10	14	24	0	1	1	0	14	14	111
16	40	0	40	5	0	5	11	0	11	41	1	42	11	7	18	0	0	0	0	6	6	122
17	38	24	62	1	0	1	6	5	11	32	26	58	7	1	8	0	0	0	0	2	2	142
18	33	1	34	3	0	3	14	0	14	43	0	43	10	7	17	0	0	0	0	8	8	119
19	21	16	37	1	1	2	4	2	6	24	22	46	10	12	22	0	0	0	0	2	2	115
20	24	12	36	2	0	2	9	1	10	36	19	55	14	13	27	0	0	0	0	4	4	134
21	37	15	52	1	6	7	6	3	9	37	21	58	2	6	8	0	0	0	0	8	8	142
22	35	6	41	6	0	6	17	0	17	43	1	44	10	13	23	0	0	0	0	9	9	140
SUM	746	131	877	127	23	150	223	30	253	906	195	1101	206	232	438	0	18	18	2	214	216	**3053**
Category:	**Closely: 2381**												**Loosely: 672**									

Table I.2: Time Spent in Coupling Styles by each pair. Each row is separated into Group (G), Individual (I) and Total amount of time (Tot) for each style. GIG = grey shaded rows and IGG = white rows.

Style:	**DISC**			**SPSA**			**SPDA**			**VE**			**VD**			**DPSA**			**DPDA**			
Pair	G	I	Tot	G	I	Tot	G	I	Tot	G	I	Tot	G	I	Tot	G	I	Tot	G	I	Tot	**SUM**
1	223	68	291	16	0	16	68	31	99	366	448	814	56	163	219	0	0	0	0	759	759	2199
2	348	7	355	50	0	50	303	0	303	771	77	848	109	252	361	0	14	14	0	618	618	2549
3	151	89	240	72	43	115	40	8	47	395	332	727	79	69	148	0	13	13	0	334	334	1625
4	402	21	423	72	0	72	422	0	422	876	33	909	50	262	312	0	3	3	0	942	942	3082
5	331	93	424	25	27	52	47	42	89	400	324	723	109	160	268	0	3	3	0	879	879	2438
6	304	10	314	56	10	65	160	275	434	731	114	845	204	140	344	0	8	8	10	562	573	2583
7	235	19	254	95	0	95	98	0	98	228	2	230	199	208	408	0	27	27	0	1381	1381	2493
8	269	173	441	12	6	18	85	27	111	751	226	977	55	205	260	0	9	9	0	234	234	2050
9	333	11	343	14	0	14	55	54	109	289	68	357	101	142	243	0	0	0	0	935	935	2001
10	415	0	415	28	0	28	96	0	96	408	43	451	62	298	360	0	0	0	0	687	687	2037
11	351	102	453	22	0	22	93	0	93	335	83	418	91	222	313	0	0	0	0	834	834	2132
12	342	20	362	27	0	27	90	0	90	464	31	495	88	140	228	0	4	4	0	498	498	1705
13	244	39	283	11	4	15	68	42	110	512	147	659	137	266	403	0	0	0	0	844	844	2313
14	183	23	206	32	0	32	25	0	25	717	32	749	89	224	314	0	0	0	3	332	334	1660
15	203	8	211	3	0	3	16	0	16	471	7	478	100	251	351	0	8	8	0	713	713	1781
16	420	0	420	36	0	36	164	0	164	422	86	508	134	109	243	0	0	0	0	494	494	1865
17	319	295	615	7	0	7	60	58	118	457	618	1075	57	11	68	0	0	0	0	211	211	2093
18	337	10	347	11	0	11	275	0	275	500	0	500	116	263	380	0	0	0	0	772	772	2284
19	146	187	333	6	6	12	72	8	81	311	1007	1318	123	203	325	0	0	0	0	114	114	2183
20	182	105	287	9	0	9	71	17	88	495	221	716	170	204	374	0	0	0	0	91	91	1565
21	238	76	314	2	41	44	44	14	59	312	177	488	25	64	90	0	0	0	0	404	404	1398
22	250	97	347	33	0	33	225	0	225	456	55	511	142	357	500	0	0	0	0	424	424	2040
SUM	6225	1452	7677	640	137	777	2576	575	3152	10665	4131	14796	2296	4214	6510	0	88	88	13	13063	13075	**46076**
Category:	**Closely: 26402**												**Loosely: 19674**									

Table I.3: Frequency of Arrangement Styles by each pair. Each row is separated into Group (G), Individual (I) and Total amount of time (Tot) for each style. GIG = grey shaded rows and IGG = white rows.

Style:	**Together**			**Kitty Corner**			**Side by Side**			**End Side**			**Opposite Ends**			**Apart**			
Pair	G	I	Tot	G	I	Tot	G	I	Tot	G	I	Tot	G	I	Tot	G	I	Tot	**SUM**
1	13	2	15	1	0	1	12	3	15	0	7	7	0	11	11	5	10	15	64
2	12	0	12	2	0	2	14	2	16	0	5	5	0	15	15	11	13	24	74
3	0	1	1	7	5	12	0	4	4	9	19	28	0	10	10	9	10	19	74
4	16	1	17	3	0	3	10	2	12	2	4	6	0	13	13	4	12	16	67
5	23	4	27	0	1	1	19	6	25	0	9	9	0	11	11	11	10	21	94
6	15	1	16	3	2	5	15	0	15	0	8	8	0	10	10	11	13	24	78
7	16	1	17	1	0	1	16	1	17	1	16	17	0	19	19	17	14	31	102
8	16	5	21	0	0	0	12	15	27	0	1	1	0	0	0	5	13	18	67
9	11	2	13	0	0	0	14	3	17	2	5	7	1	12	13	11	8	19	69
10	1	0	1	1	0	1	9	1	10	0	3	3	0	13	13	7	11	18	46
11	23	4	27	5	2	7	22	4	26	1	3	4	0	9	9	12	11	23	96
12	17	2	19	3	1	4	17	4	21	0	5	5	0	6	6	11	12	23	78
13	10	1	11	11	4	15	8	3	11	5	11	16	0	13	13	11	14	25	91
14	4	0	4	4	1	5	9	2	11	1	11	12	0	1	1	7	9	16	49
15	11	0	11	1	2	3	12	0	12	1	5	6	0	13	13	10	12	22	67
16	10	0	10	1	1	2	12	1	13	0	5	5	0	4	4	11	6	17	51
17	14	5	19	0	2	2	11	3	14	1	2	3	0	0	0	7	0	7	45
18	10	0	10	3	0	3	11	0	11	3	8	11	0	4	4	10	7	17	56
19	5	13	18	7	5	12	5	7	12	1	5	6	0	3	3	11	14	25	76
20	7	4	11	8	6	14	11	8	19	4	2	6	0	4	4	14	13	27	81
21	6	3	9	1	2	3	7	2	9	1	6	7	0	7	7	2	4	6	41
22	10	1	11	1	1	2	11	3	14	2	2	4	0	9	9	9	12	21	61
SUM	250	50	300	63	35	98	257	74	331	34	142	176	1	187	188	206	228	434	**1527**
Category:	**Closely: 729**									**Loosely: 798**									

Table I.4: Time Spent in Arrangement Styles by each pair. Each row is separated into Group (G), Individual (I) and Total amount of time (Tot) for each style. GIG = grey shaded rows and IGG = white rows.

Style:	**Together**			**Kitty Corner**			**Side by Side**			**End Side**			**Opposite Ends**			**Apart**			
Pair	G	I	Tot	G	I	Tot	G	I	Tot	G	I	Tot	G	I	Tot	G	I	Tot	**SUM**
1	166	50	216	18	0	18	516	153 b	669	0	363	363	0	740	740	51	163	214	2220
2	317	0	317	32	0	32	1168	87	1255	0	149	149	0	529	529	109	204	314	2594
3	0	5	5	264	95	359	0	56	56	465	344	808	0	336	336	73	61	133	1697
4	277	14	291	103	0	103	1280	12	1292	124	51	174	0	1056	1056	55	243	298	3215
5	362	88	450	0	7	7	509	361	869	0	183	183	0	761	761	117	159	277	2548
6	319	15	334	156	20	176	880	0	880	0	460	460	0	557	557	203	147	350	2757
7	266	42	309	17	0	17	417	21	438	8	404	412	0	1032	1032	211	208	419	2626
8	189	176	364	0	0	0	965	533	1498	0	39	39	0	0	0	51	211	261	2163
9	248	54	303	0	0	0	416	52	469	18	45	63	12	984	996	99	142	241	2071
10	18	0	18	36	0	36	916	40	956	0	55	55	0	635	635	62	298	360	2060
11	265	95	360	73	14	87	507	86	593	8	78	86	0	847	847	107	220	327	2300
12	241	16	257	55	7	61	666	52	718	0	204	204	0	322	322	104	159	263	1826
13	310	39	350	392	97	489	204	77	280	75	179	255	0	687	687	137	263	400	2461
14	78	0	78	268	29	297	620	16	637	30	404	433	0	3	3	74	247	320	1769
15	233	0	233	12	19	31	443	0	443	6	320	326	0	424	424	100	225	325	1782
16	278	0	278	13	63	75	781	14	794	0	265	265	0	248	248	134	100	233	1894
17	285	113	398	0	26	26	566	896	1461	23	195	218	0	0	0	57	0	57	2160
18	211	0	211	23	0	23	925	0	925	38	652	690	0	200	200	116	263	379	2428
19	109	216	325	221	331	553	193	574	767	5	136	141	0	103	103	131	216	347	2235
20	131	86	217	185	104	289	375	165	539	68	14	82	0	68	68	170	201	371	1567
21	74	37	110	5	68	73	514	212	726	9	109	118	0	320	320	23	60	83	1429
22	230	11	240	12	25	37	743	100	844	20	25	45	0	366	366	130	601	731	2262
SUM	4605	1057	5662	1884	904	2788	13603	3507	17110	897	4673	5569	12	10218	10230	2314	4392	6705	**48064**
Category:	**Closely: 25560**									**Loosely: 22504**									

Table I.5: Frequency of Coupling Styles - Wilcoxon Rank Sum Tests at 95% confidence using a Bonferroni correction to the p-value. The grey cells represent the intersection of styles that have significant differences between each other.

Coupling:	value	**DISC**	**SPSA**	**SPDA**	**VE**	**VD**	**DPSA**	**DPDA**
DISC	W		482.5	482.5	116	474.5	474.5	484
	p		3.5994×10^{-07}	3.5994×10^{-07}	0.067116	0.00000105399	1.91793×10^{-07}	2.9127×10^{-07}
SPSA	W			113.5	1	37.5	451.5	131.5
	p			0.054684	3.3558×10^{-07}	0.000034104	0.000012957	0.200886
SPDA	W				1	77	478	273.5
	p				3.4062e-7	0.0023205	4.3638×10^{-07}	9.7671
VE	W					482.5	484	484
	p					3.6078×10^{-07}	1.91079×10^{-07}	2.9022×10^{-07}
VD	W						484	435.5
	p						1.88979×10^{-07}	0.000118587
DPSA	W							8.5
	p							5.9829×10^{-07}
DPDA	W							
	p							

Table I.6: Time Spent in Coupling Styles - Wilcoxon Rank Sum Tests at 95% confidence using a Bonferroni correction to the p-value. The grey cells represent the intersection of styles that have significant differences between each other.

Coupling:	value	**DISC**	**SPSA**	**SPDA**	**VE**	**VD**	**DPSA**	**DPDA**
DISC	W		484	435	42	303	484	123
	p		1.99605×10^{-11}	0.0000196728	0.0000059199	3.297	1.95594×10^{-07}	0.097125
SPSA	W			53	0	5	456	3
	p			0.00003738	1.99605×10^{-11}	3.7926×10^{-10}	0.0000080556	1.39734×10^{-10}
SPDA	W				8	88	482	37
	p				1.33749×10^{-9}	0.0036582	2.5872×10^{-7}	0.0000023394
VE	W					455	484	281
	p					4.5654×10^{-07}	1.95594×10^{-07}	7.7658
VD	W						484	99
	p						1.95594×10^{-07}	0.0115143
DPSA	W							0
	p							1.95594×10^{-07}
DPDA	W							
	p							

Table I.7: Frequency of Arrangement Styles - Wilcoxon Rank Sum Tests at 95% confidence using a Bonferroni correction to the p-value. The grey cells represent the intersection of styles that have significant differences between each other.

Arrangement :	value	**Together**	**Kitty Corner**	**Side by Side**	**End Side**	**Opposite Ends**	**Apart**
Together	W		407	215	368.5	346.5	117.5
	p		0.00162	7.9875	0.04563	0.21405	0.05319
Kitty Corner	W			42.5	119.5	136.5	10
	p			0.00004263	0.060555	0.20085	7.90×10^{-7}
Side by Side	W				406	385.5	124.5
	p				0.0018105	0.011412	0.089265
End Side	W					209	45.5
	p					6.6615	0.000060765
Opposite Ends	W						36.5
	p						0.000021645
Apart	W						
	p						

Table I.8: Time Spent in Arrangement Styles - Wilcoxon Rank Sum Tests at 95% confidence using a Bonferroni correction to the p-value. The grey cells represent the intersection of styles that have significant differences between each other.

Arrangement :	value	**Together**	**Kitty Corner**	**Side by Side**	**End Side**	**Opposite Ends**	**Apart**
Together	W		367	33	277	158	193
	p		0.052095	7.5735×10^{-7}	6.3285	0.74985	3.867
Kitty Corner	W			23	119	102	86
	p			0.000004371	0.06051	0.01581	0.0039315
Side by Side	W				438	353	440
	p				0.000008508	0.14238	0.0000060245
End Side	W					159	174
	p					0.792	1.7055
Opposite Ends	W						303
	p						2.334
Apart	W						
	p						

Glossary

Apart participants were standing apart, physical arrangement style.

CCV Community Core Vision, detection software.

Closely Coupled Category coupling styles that are closely coupled (DISC, VE, SPDA, SPSA).

DISC participants were in discussion, coupling style.

DPDA participants were working on different problems and different areas of the table, coupling style.

DPSA participants were working on different problems and same area of the table, coupling style.

End Side participants were arranged by end side, arrangement style.

Kitty Corner participants were arranged by kitty corner, arrangement style.

Loosely Coupled Category coupling styles that are loosely coupled (VD, DPDA, DPSA).

MT4j Multi-touch For Java [188], toolkit used for implementing SourceVis.

Opposite Ends participants were arranged at opposite ends of the table, arrangement style.

Side by Side participants were arranged side by side, arrangement style.

SPDA participants were working on the same problem different area, coupling style.

SPSA participants were working on the same problem same area, coupling style.

Together participants were arranged together, arrangement style.

VE one participant was interacting while the other was viewing engaged, coupling style.

Bibliography

[1] K. Alfert and F. Engelen. Three-dimensional visualization of Java class relations. In *Proceedings of the World Conference on Integrated Design & Process Technology (IDPT)*, 2000.

[2] K. Alfert and F. Engelen. Experiences in 3-Dimensional visualization of Java class relations. *Journal of Design & Process Science*, 5(3):91–106, 2001.

[3] K. Alfert and A. Fronk. Manipulation of three-dimensional visualization of Java class relations. In *Proceedings of the World Conference on Integrated Design & Process Technology (IDPT)*, 2002.

[4] T. Alspaugh, B. Tomlinson, and E. Baumer. Using social agents to visualize software scenarios. In *Proceedings of the ACM Symposium on Software Visualization (SOFTVIS)*, pages 87–94. ACM Press, 2006.

[5] R. Amar, J. Eagan, and J. Stasko. Low-level components of analytic activity in information visualization. In *Proceedings of the IEEE Symposium on Information Visualization (INFOVIS)*, pages 111–117. IEEE Press, 2005.

[6] R. Amar and J. Stasko. A knowledge task-based framework for design and evaluation of information visualizations. In *Proceedings of the IEEE Symposium on Information Visualization (INFOVIS)*, pages 143–150. IEEE Press, 2004.

[7] C. Andrews, A. Endert, and C. North. Space to think: large high-resolution displays for sensemaking. In *Proceedings of the ACM International Conference on Human Factors in Computing Systems (CHI)*, pages 55–64. ACM Press, 2010.

[8] C. Anslow. Evaluating Extensible 3D (X3D) graphics for use in software visualisation. Master's thesis, Victoria University of Wellington, 2008.

[9] C. Anslow, J. Noble, S. Marshall, and R. Biddle. Web software visualization using extensible 3D (X3D) graphics. In *Proceedings of the ACM Symposium on Software Visualization (SOFTVIS)*, pages 213–214. ACM Press, 2008.

[10] C. Anslow, J. Noble, S. Marshall, E. Tempero, and R. Biddle. User evaluation of polymetric views using a large visualization wall. In *Proceedings of the ACM Symposium on Software Visualization (SOFTVIS)*, pages 25–34. ACM Press, 2010.

[11] R. Baecker. Sorting out sorting. 30 minute colour sound videotape, 1981. Presented at ACM SIGGRAPH '81 and excerpted and reprinted in ACM SIGGRAPH Video Review 7, 1983.

[12] R. Baecker. *Software Visualization*, chapter Sorting Out Sorting: A Case Study of Software Visualisation for Teaching Computer Science, pages 369–381. MIT Press, 1998.

[13] M. Baker and S. Eick. Visualizing software systems. In *Proceedings of the International Conference on Software Engineering (ICSE)*, pages 59–67. IEEE Press, 1994.

[14] R. Ball, C. North, and D. Bowman. Move to improve: promoting physical navigation to increase user performance with large displays. In *Proceedings of the ACM International Conference on Human Factors in Computing Systems (CHI)*, pages 191–200. ACM Press, 2007.

[15] M. Balzer and O. Deussen. Hierarchy based 3D visualization of large software structures. In *Proceedings of the IEEE Conference on Visualization (VIS)*, page 598.4. IEEE Press, 2004.

[16] M. Balzer, A. Noack, O. Deussen, and C. Lewerentz. Software landscapes: Visualizing the structure of large software systems. In *In Proceedings of the Symposium on Visualization (VisSym)*, pages 261–266. Eurographics Association, 2004.

[17] M. Basheri, N. Baghaei, L. Burd, and M. Munro. Collaborative learning skills in multi-touch tables for uml software design. *International Journal of Advanced Computer Science and Applications*, 4(3):60–66, 2013.

[18] M. Basheri and L. Burd. Exploring the significance of multi-touch tables in enhancing collaborative software design using UML. In *Proceedings of the Frontiers in Education Conference (FIE)*, pages 1–5. IEEE Press, 2012.

[19] M. Basheri, L. Burd, and N. Baghaei. MT-CollabUML: Collaborative software design using multi-touch tables. In *Proceedings of the International Congress on Engineering Education (ICEED)*, 2012.

[20] M. Basheri, L. Burd, and N. Baghaei. A multi-touch interface for enhancing collaborative UML diagramming. In *Proceedings of the Australian Computer-Human Interaction Conference (OzCHI)*, pages 30–33. ACM Press, 2012.

[21] P. Baudisch, T. Becker, and F. Rudeck. Lumino: tangible blocks for tabletop computers based on glass fiber bundles. In *Proceedings of the ACM International Conference on Human Factors in Computing Systems (CHI)*, pages 1165–1174. ACM Press, 2010.

[22] K. Beck. *Extreme Programming Explained: Embrace Change*. Addison Wesley, 1999.

[23] B. Bederson, J. Grosjean, and J. Meyer. Toolkit design for interactive structured graphics. *IEEE Transactions on Software Engineering*, 30(8):535–546, 2004.

[24] B. Bederson and J. Hollan. Pad++: a zooming graphical interface for exploring alternate interface physics. In *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST)*. ACM Press, 1994.

[25] B. Bederson and B. Shneiderman, editors. *The Craft of Information Visualization: Readings and Reflections*. Morgan Kaufmann, 2003.

[26] R. Bencina, M. Kaltenbrunner, and S. Jorda. Improved topological fiducial tracking in the reactivision system. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, page 99. IEEE Press, 2005.

[27] A. Bergel, R. Robbes, and W. Binder. Visualizing dynamic metrics with profiling blueprints. In *Proceedings of TOOLS Europe*, pages 291–309. Springer Verlag, 2010.

[28] T. Bernardin, B. Budge, and B. Hamann. Stacked-widget visualization of scheduling-based algorithms. In *Proceedings of the ACM Symposium on Software Visualization (SOFTVIS)*, pages 165–174. ACM Press, 2008.

[29] D. Beyer. Co-change visualization. In *Proceedings of the IEEE International Conference on Software Maintenance (ICSM)*, pages 89–92. IEEE Press, 2005.

[30] D. Beyer. Co-change visualization applied to postgresql and argouml: (msr challenge report). In *Proceedings of the International Workshop on Mining Software Repositories (MSR)*, pages 165–166. ACM Press, 2006.

[31] D. Beyer and A. Noack. Clustering software artifacts based on frequent common changes. In *Proceedings of the IEEE International Workshop on Program Comprehension (IWPC)*, pages 259–268. IEEE Press, 2005.

[32] X. Bi, S.-H. Bae, and R. Balakrishnan. Effects of interior bezels of tiled-monitor large displays on visual search, tunnel steering, and target selection. In *Proceedings of the ACM International Conference on Human Factors in Computing Systems (CHI)*, pages 65–74. ACM Press, 2010.

[33] X. Bi and R. Balakrishnan. Comparing usage of a large high-resolution display to single or dual desktop displays for daily work. In *Proceedings of the ACM International Conference on Human Factors in Computing Systems (CHI)*, pages 1005–1014. ACM Press, 2009.

[34] J. Biehl, W. Baker, B. Bailey, D. Tan, K. Inkpen, and M. Czerwinski. Impromptu: a new interaction framework for supporting collaboration in multiple display environments and its field evaluation for co-located software development. In *Proceedings of the ACM International Conference on Human Factors in Computing Systems (CHI)*, pages 939–948. ACM Press, 2008.

[35] J. Biehl, M. Czerwinski, G. Smith, G., and Robertson. Fastdash: a visual dashboard for fostering awareness in software teams. In *Proceedings of the ACM International Conference on Human Factors in Computing Systems (CHI)*, pages 1313–1322. ACM Press, 2007.

[36] M. Blumenkrants, H. Starovisky, and A. Shamir. Narrative algorithm visualization. In *Proceedings of the ACM Symposium on Software Visualization (SOFTVIS)*, pages 17–26. ACM Press, 2006.

[37] S. Boccuzzo and H. Gall. Cocoviz: Supported cognitive software visualization. In *Proceedings of the Working Conference on Reverse Engineering (WCRE)*, pages 273–274. IEEE Press, 2007.

[38] S. Boccuzzo and H. Gall. Cocoviz: Towards cognitive software visualization. In *Proceedings of the International Workshop on Visualizing Software for Understanding and Analysis (VISSOFT)*, pages 72–79. IEEE Press, 2007.

[39] S. Boccuzzo and H. Gall. Software visualization with audio supported cognitive glyphs. In *Proceedings of the IEEE International Conference on Software Maintenance (ICSM)*, pages 366–375. IEEE Press, 2008.

[40] S. Boccuzzo and H. Gall. Cocoviz with ambient audio software exploration. In *Proceedings of the International Conference on Software Engineering (ICSE)*, pages 571–574. IEEE Press, 2009.

[41] S. Boccuzzo and H. Gall. Multi-touch collaboration for software exploration. In *Proceedings of the IEEE International Conference on Program Comprehension (ICPC)*. IEEE Press, 2010.

[42] J. Bohnet and J. Döllner. Visual exploration of function call graphs for feature location in complex software systems. In *Proceedings of the ACM Symposium on Software Visualization (SOFTVIS)*, pages 95–104. ACM Press, 2006.

[43] J. Bohnet, M. Koeleman, and J. Döllner. Visualizing massively pruned execution traces to facilitate trace exploration. In *Proceedings of the International Workshop on Visualizing Software for Understanding and Analysis (VISSOFT)*, pages 57–64. IEEE Press, 2009.

[44] M. Bostock, V. Ogievetsky, and J. Heer. D3 data-driven documents. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2301–2309, Dec. 2011.

[45] F. Bott, S. Diehl, and R. Lutz. CREWW: collaborative requirements engineering with Wii-remotes (NIER track). In *Proceedings of the International Conference on Software Engineering (ICSE)*, pages 852–855. ACM Press, 2011.

[46] A. Bragdon, R. DeLine, K. Hinckley, and M. Morris. Code space: touch + air gesture hybrid interactions for supporting developer meetings. In *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces (ITS)*, pages 212–221. ACM Press, 2011.

[47] J. Brittle and C. Boldyreff. Self-organizing maps applied in visualizing large software collections. In *Proceedings of the International Workshop on Visualizing Software for Understanding and Analysis (VISSOFT)*, pages 104–109. IEEE Press, 2003.

[48] M. Brown. *Algorithm Animation*. PhD thesis, Brown University, 1987. MIT Press.

[49] M. Brown. Zeus: A system for algorithm animation and multi-view editing. In *Proceedings of the IEEE Workshop on Visual Languages (VL)*, pages 4–9. IEEE Press, 1991.

[50] M. Brown and M. Najork. Algorithm animation using 3D interactive graphics. In *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST)*, pages 93–100. ACM Press, 1993.

[51] M. Brown and R. Sedgewick. A system for algorithm animation. In *Proceedings of the ACM Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, pages 177–186. ACM Press, 1984.

[52] B. Bruegge and A. Dutoit. *Object-Oriented Software Engineering: Using UML, Patterns and Java*. Prentice Hall, 2003.

[53] H. Byelas and A. Telea. Visualization of areas of interest in software architecture diagrams. In *Proceedings of the ACM Symposium on Software Visualization (SOFTVIS)*, pages 105–114. ACM Press, 2006.

[54] H. Byelas and A. Telea. Texture-based visualization of metrics on software architectures. In *Proceedings of the ACM Symposium on Software Visualization (SOFTVIS)*, pages 205–206. ACM Press, 2008.

[55] M. Callaghan and H. Hirschmuller. 3D visualisation of design patterns and Java programs in computer science education. In *Proceedings of the ACM Conference on Integrating Technology into Computer Science Education (ITiCSE)*, pages 37–40. ACM Press, 1998.

[56] S. Card, J. Mackinlay, and B. Shneiderman. *Readings in information visualization: using vision to think*. Morgan Kaufmann, 1999.

[57] S. Carpendale, D. Cowperthwaite, and D. Fracchia. Extending distortion viewing from 2D to 3D. *IEEE Computer Graphics Applications*, 17(4):42–51, 1997.

[58] K. Casey and C. Exton. A Java 3D implementation of a geon based visualisation tool for UML. In *Proceedings of the International Conference on Principles and Practice of Programming in Java (PPPJ)*, pages 63–65. Computer Science Press, Inc., 2003.

[59] A. Caudwell. Gource: visualizing software version control history. In *Companion to the ACM Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA)*, SPLASH, pages 73–74. ACM Press, 2010.

[60] S. H.-H. Chang, L. Stuart, B. Plimmer, and B. Wuensche. Origami simulator: a multi-touch experience. In *Proceedings of the ACM International Conference Extended Abstracts on Human Factors in Computing Systems (CHI)*, pages 3889–3894. ACM Press, 2009.

[61] S. Charters, C. Knight, N. Thomas, and M. Munro. Visualisation for informed decision making; from code to components. In *Proceedings of the ACM Conference on Software Engineering and Knowledge Engineering (SEKE)*, pages 765–772. ACM Press, 2002.

[62] C. Chen. *Information Visualization: Beyond The Horizon*. Springer Verlag, 2006.

[63] F. Chen, P. Eades, J. Epps, S. Lichman, B. Close, P. Hutterer, M. Takatsuka, B. Thomas, and M. Wu. Vicat: Visualisation and interaction on a collaborative access table. In *Proceedings of the IEEE International Workshop on Horizontal Interactive Human Computer Systems (TABLETOP)*, pages 59–62. IEEE Press, 2006.

[64] S. Chidamber and C. Kemerer. Towards a metrics suite for object oriented design. In *Proceedings of ACM Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA)*, pages 197–211. ACM Press, 1991.

[65] N. Churcher, W. Irwin, and R. Kriz. Visualising class cohesion with virtual worlds. In *Proceedings of the Asia-Pacific Symposium on Information Visualisation (APVIS)*, pages 89–97. Australian Computer Society, Inc, 2003.

[66] N. Churcher, L. Keown, and W. Irwin. Virtual worlds for software visualisation. In *Proceedings of the Workshop on Software Visualisation Workshop (SoftVis)*, pages 9–16, 1999.

[67] A. Cockburn. *Agile Software Development*. Addison Wesley, 2001.

[68] N. Couture, G. Rivière, and P. Reuter. Geotui: a tangible user interface for geoscience. In *Proceedings of the International Conference on Tangible and Embedded Interaction (TEI)*, pages 89–96. ACM Press, 2008.

[69] K. Cox and G.-C. Roman. Abstraction in algorithm animation. In *Proceedings of the IEEE Workshop on Visual Languages (VL)*, pages 18–24. IEEE Press, 1992.

[70] B. Craft and P. Cairns. Beyond guidelines: What can we learn from the visual information seeking mantra? In *Proceedings of the IEEE International Conference on Information Visualization (IV)*, pages 110–118. IEEE Press, 2005.

[71] J. Creswell. *Research Design: Qualitative, Quantitative, and Mixed Methods Approaches*. SAGE Publications, 2008.

[72] R. Dachselt, M. Frisch, and E. Decker. Enhancing uml sketch tools with digital pens and paper. In *Proceedings of the ACM Symposium on Software Visualization (SOFTVIS)*, pages 203–204. ACM Press, 2008.

[73] C. Danis, F. Viegas, M. Wattenberg, and J. Kriss. Your place or mine?: visualization as a community component. In *Proceedings of the ACM International Conference on Human Factors in Computing Systems (CHI)*, pages 275–284. ACM Press, 2008.

[74] W. Dann, S. Cooper, and R. Pausch. Using visualization to teach novices recursion. *ACM SIGCSE Bulletin*, 33(3):109–112, 2001.

[75] T. Davis, K. Pestka, and A. Kaplan. Kscope: A modularized tool for 3D visualization of object-oriented programs. In *Proceedings of the International Workshop on Visualizing Software for Understanding and Analysis (VISSOFT)*, pages 98–103. IEEE Press, 2003.

[76] S. Diehl. *Revised Lectures on Software Visualization, International Seminar*. Springer-Verlag, 2002.

[77] S. Diehl. *Software Visualization: Visualizing the Structure, Behaviour, and Evolution of Software*. Springer Verlag, 2007.

[78] J. Dietrich, V. Yakovlev, C. McCartin, G. Jenson, and M. Duchrow. Cluster analysis of Java dependency graphs. In *Proceedings of the ACM Symposium on Software Visualization (SOFTVIS)*, pages 91–94. ACM Press, 2008.

[79] P. Dietz and D. Leigh. Diamondtouch: a multi-user touch technology. In *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST)*, pages 219–226. ACM Press, 2001.

[80] A. Dix, J. Finlay, G. Abowd, and R. Beale. *Human-Compuer Interaction*. Prentice Hall, 2003.

[81] K. Dohse, T. Dohse, J. Still, and D. Parkhurst. Enhancing multi-user interaction with multi-touch tabletop displays using hand tracking. In *Proceedings of the International Conference on Advances in Computer-Human Interaction (ACHI)*, pages 297–302. IEEE Press, 2008.

[82] N. Drew and R. Hendley. Visualisation of complex systems. Technical report, University of Birmingham, 1995.

[83] N. Drew and R. Hendley. Visualising complex interacting systems. In *Proceedings of the ACM International Conference on Human Factors in Computing Systems (CHI)*, pages 204–205. ACM Press, 1995.

[84] M. Duignan. Evaluating scalable vector graphics for software visualisation. Master's thesis, Victoria University of Wellington, 2003.

[85] M. Duignan, R. Biddle, and E. Tempero. Evaluating scalable vector graphics for use in software visualisation. In *Proceedings of the Australasian Symposium on Information Visualisation (INVIS)*, pages 127–136. Australian Computer Society, Inc, 2003.

[86] T. Dwyer. Three dimensional UML using force directed layout. In *Proceedings of the Australasian Symposium on Information Visualisation (INVIS)*, pages 77–85. Australian Computer Society, Inc, 2001.

[87] F. Echtler, A. Dippon, M. Tönnis, and G. Klinker. Inverted FTIR: Easy multitouch sensing for flatscreens. In *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces (ITS)*, pages 29–32. ACM Press, 2009.

[88] F. Echtler, M. Huber, and G. Klinker. Shadow tracking on multi-touch tables. In *Proceedings of the Working Conference on Advanced Visual Interfaces (AVI)*, pages 388–391. ACM Press, 2008.

[89] F. Echtler and G. Klinker. A multitouch software architecture. In *Proceedings of the Nordic conference on Human-computer interaction (NordiCHI)*, pages 463–466. ACM Press, 2008.

[90] F. Echtler, S. Nestler, A. Dippon, and G. Klinker. Supporting casual interactions between board games on public tabletop displays and mobile devices. *Personal Ubiquitous Comput.*, 13(8):609–617, 2009.

[91] H. Eichelberger. Automatic layout of UML use case diagrams. In *Proceedings of the ACM Symposium on Software Visualization (SOFTVIS)*, pages 105–114. ACM Press, 2008.

[92] S. Eick, J. Steffen, and E. Sumner. Seesoft-a tool for visualizing line oriented software statistics. *IEEE Transactions on Software Engineering*, 18(11):957–968, 1992.

[93] G. Ellis and A. Dix. An explorative analysis of user evaluation studies in information visualisation. In *Proceedings of the AVI Workshop on BEyond time and errors: novel evaLuation methods for Information Visualization (BELIV)*, pages 1–7. ACM Press, 2006.

[94] J. Epps, S. Lichman, and M. Wu. A study of hand shape use in tabletop gesture interaction. In *Proceedings of the ACM International Conference Extended Abstracts on Human Factors in Computing Systems (CHI)*, pages 748–753. ACM Press, 2006.

[95] L. Feijs and R. D. Jong. 3D visualization of software architectures. *Communications of the ACM*, 41(12):73–78, 1998.

[96] N. Fenton and S. Pfleeger. *Software Metrics: A Rigorous and Practical Approach*. PWS Publishing, 1998.

[97] G. Franck, M. Sardesai, and C. Ware. Layout and structuring object oriented software in three dimensions. In *Proceedings of the Conference of the Centre for Advanced Studies on Collaborative Research (CASCON)*, pages 22–31. IBM Press, 1995.

[98] M. Frisch and R. Dachselt. Towards a framework for supporting software modeling activities through novel interaction and visualization techniques. In *ICSE Doctoral Symposium*. IEEE Press, 2009.

[99] M. Frisch, R. Dachselt, and T. Brückmann. Towards seamless semantic zooming techniques for UML diagrams. In *Proceedings of the ACM Symposium on Software Visualization (SOFTVIS)*, pages 207–208. ACM Press, 2008.

[100] M. Frisch, S. Schmidt, J. Heydekorn, M. Nacenta, R. Dachselt, and S. Carpendale. Editing and exploring node-link diagrams on pen- and multi-touch-operated tabletops. In *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces (ITS)*, pages 304–304. ACM Press, 2010.

[101] T. Fritz and G. Murphy. Using information fragments to answer the questions developers ask. In *Proceedings of the International Conference on Software Engineering (ICSE)*, pages 175–184. ACM Press, 2010.

[102] A. Fronk. Evaluating 3D-visualisation of code structures in the context of reverse engineering. In *Proceedings of the Workshop on Empirical Studies in Reverse Engineering (WESRE)*. IEEE Press, 2006.

[103] A. Fronk, A. Bruckhoff, and M. Kern. 3D visualisation of code structures in Java software systems. In *Proceedings of the ACM Symposium on Software Visualization (SOFTVIS)*, pages 145–146. ACM Press, 2006.

[104] B. Fry. *Visualizing Data*. O'Reilly, 2008.

[105] K. Garland. *Mr. Beck's Underground Map*. Capital Transport Publishing, 1994.

[106] Y. Ghanam, X. Wang, and F. Maurer. Utilizing digital tabletops in collocated agile planning meetings. In *Proceedings of the International Conference on Agile*, pages 51–62. IEEE Press, 2008.

[107] J. Gil and S. Kent. Three dimensional software modelling. In *Proceedings of the International Conference on Software Engineering (ICSE)*, pages 105–114. IEEE Press, 1998.

[108] M. Gogolla, O. Radfelder, and M. Richters. Towards three-dimensional animation of UML diagrams. In *Proceedings of the International Conference on Unified Modeling Language (UML)*, pages 489–502. Springer Verlag, 1999.

[109] A. Gokcezade, J. Leitner, and M. Haller. Lighttracker: An open-source multitouch toolkit. *Computers in Entertainment (CIE)*, 8(3):19:1–19:16, Dec. 2010.

[110] D. Goldman, R. Eckert, and M. Cohen. Three-dimensional computation visualization for computer graphics rendering algorithms. In *Proceedings of the ACM Symposium on Computer Science Education (SIGCSE)*, pages 358–362. ACM Press, 1996.

[111] H. Graham, H. Y. Yang, and R. Berrigan. A solar system metaphor for 3D visualisation of object oriented software metrics. In *Proceedings of the Australasian Symposium on Information Visualisation (INVIS)*, pages 53–59. Australian Computer Society, Inc., 2004.

[112] O. Greevy, M. Lanza, and C. Wysseier. Visualizing feature interaction in 3-D. In *Proceedings of the International Workshop on Visualizing Software for Understanding and Analysis (VISSOFT)*, pages 114–119. IEEE Press, 2005.

[113] O. Greevy, M. Lanza, and C. Wysseier. Visualizing live software systems in 3D. In *Proceedings of the ACM Symposium on Software Visualization (SOFTVIS)*, pages 47–56. ACM Press, 2006.

[114] P. Guo. Online python tutor: embeddable web-based program visualization for cs education. In *Proceedings of the ACM Symposium on Computer Science Education (SIGCSE)*, pages 579–584. ACM Press, 2013.

[115] J. Han. Low-cost multi-touch sensing through frustrated total internal reflection. In *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST)*, pages 115–118. ACM Press, 2005.

[116] J. Han. Multi-touch interaction wall. In *ACM SIGGRAPH Emerging technologies*, page 25. ACM Press, 2006.

[117] J. Hardy, C. Bull, G. Kotonya, and J. Whittle. Digitally annexing desk space for software development (NIER track). In *Proceedings of the International Conference on Software Engineering (ICSE)*, pages 812–815. ACM Press, 2011.

[118] D. Harel and I. Segall. Visualizing inter-dependencies between scenarios. In *Proceedings of the ACM Symposium on Software Visualization (SOFTVIS)*, pages 145–153. ACM Press, 2008.

[119] R. Harris. *Information graphics a comprehensive illustrated reference: visual tools for analyzing, managing, and communicating*. Oxford University Press, 1999.

[120] D. Hartley, N. Churcher, and G. Albertson. Virtual worlds for web site visualisation. In *Proceedings of the Asia-Pacific Software Engineering Conference (APSEC)*, pages 448–455. IEEE Press, 2000.

[121] A. Hatch, M. Smith, C. Taylor, and M. Munro. No silver bullet for software visualisation evaluation. In *Proceedings of The International Conference on Imaging Science, Systems, and Technology (CISST)*, 2001.

[122] R. Hendley, N. Drew, A. Wood, and R. Beale. Case study: Narcissus: visualising information. In *Proceedings of the IEEE Symposium on Information Visualization (INFOVIS)*, pages 90–96. IEEE Press, 1995.

[123] J. Highsmith. *Agile Project Management: Creating Innovative Products*. Addison Wesley, 2009.

[124] O. Hilliges, D. Baur, and A. Butz. Photohelix: Browsing, sorting and sharing digital photo collections. In *Proceedings of the IEEE International Workshop on Horizontal Interactive Human Computer Systems (TABLETOP)*, pages 87–94. IEEE Press, 2007.

[125] O. Hilliges, A. Butz, S. Izadi, and A. Wilson. *Tabletops - Horizontal Interactive Displays*, chapter Interaction on the tabletop: bringing the physical to the digital. Springer Verlag, 2010.

[126] J. Hochenbaum and O. Vallis. Bricktable: A musical tangible multi-touch interface. In *Proceedings of Berlin Open Conference*, 2009.

[127] J. Hochenbaum, O. Vallis, D. Diakopoulos, M. Akten, and A. Kapur. Musical applications for multi-touch surfaces. In *Workshop on Media Arts, Science, and Technology*, 2009.

[128] S. Hodges, S. Izadi, A. Butler, A. Rrustemi, and B. Buxton. Thinsight: versatile multi-touch sensing for thin form-factor displays. In *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST)*, pages 259–268. ACM Press, 2007.

[129] C. Holz and P. Baudisch. Fiberio: A touchscreen that senses fingerprints. In *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST)*, pages 41–50. ACM Press, 2013.

[130] J. Hopkins and P. Fishwick. The rube framework for personalized 3-D software visualization. In *Revised Lectures on Software Visualization, International Seminar*, pages 368–380. Springer Verlag, 2002.

[131] P. Hutterer, B. Close, and B. Thomas. Supporting mixed presence groupware in tabletop applications. In *Proceedings of the IEEE International Workshop on Horizontal Interactive Human Computer Systems (TABLETOP)*, pages 63–70. IEEE Press, 2006.

[132] P. Hutterer and B. Thomas. Groupware support in the windowing system. In *Proceedings of the Australasian Conference on User Interfaces (AUIC)*, pages 39–46. Australian Computer Society, Inc, 2007.

[133] P. Irani, M. Tingley, and C. Ware. Using perceptual syntax to enhance semantic content in diagrams. *IEEE Computer Graphics Applications*, 21(5):76–85, 2001.

[134] P. Irani and C. Ware. Diagrams based on structural object perception. In *Proceedings of the Working Conference on Advanced Visual Interfaces (AVI)*, pages 61–67. ACM Press, 2000.

[135] P. Irani and C. Ware. Diagramming information structures using 3D perceptual primitives. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 10(1):1–19, 2003.

[136] W. Irwin and N. Churcher. Object oriented metrics: Precision tools and configurable visualisations. In *Proceedings of the International Symposium on Software Metrics (METRICS)*, pages 112–123. IEEE Press, 2003.

[137] P. Isenberg. *Collaborative Information Visualization in Co-located Environments*. PhD thesis, University of Calgary, 2009.

[138] P. Isenberg and S. Carpendale. Interactive tree comparison for co-located collaborative information visualization. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1232–1239, Nov. 2007.

[139] P. Isenberg, N. Elmqvist, J. Scholtz, D. Cernea, K.-L. Ma, and H. Hagen. Collaborative visualization: definition, challenges, and research agenda. *Information Visualization*, 10(4):310–326, Oct. 2011.

[140] P. Isenberg and D. Fisher. Collaborative brushing and linking for co-located visual analytics of document collections. In *Proceedings of the EuroGraphics/IEEE Symposium on Visualisation (EuroVis)*, pages 1031–1038. IEEE Press, 2009.

[141] P. Isenberg, D. Fisher, M. Morris, K. Inkpen, and M. Czerwinski. An exploratory study of co-located collaborative visual analytics around a tabletop display. In *Proceedings of the IEEE Symposium on Visual Analytics Science and Technology (VAST)*. IEEE Press, 2010.

[142] P. Isenberg, D. Fisher, S. Paul, M. Morris, K. Inkpen, and M. Czerwinski. Co-located collaborative visual analytics around a tabletop display. *IEEE Transactions on Visualization and Computer Graphics*, 18(5):689–702, May 2012.

[143] P. Isenberg, A. Tang, and S. Carpendale. An exploratory study of visual information analysis. In *Proceedings of the ACM International Conference on Human Factors in Computing Systems (CHI)*, pages 1217–1226. ACM Press, 2008.

[144] P. Isenberg, T. Zuk, C. Collins, and S. Carpendale. Grounded evaluation of information visualizations. In *Proceedings of the AVI Workshop on BEyond time and errors: novel evaLuation methods for Information Visualization (BELIV)*. ACM Press, 2008.

[145] H. Ishii and B. Ullmer. Tangible bits: towards seamless interfaces between people, bits and atoms. In *Proceedings of the ACM International Conference on Human Factors in Computing Systems (CHI)*, pages 234–241. ACM Press, 1997.

[146] S. Izadi, S. Hodges, A. Butler, D. West, A. Rrustemi, M. Molloy, and B. Buxton. Thinsight: a thin form-factor interactive surface technology. *Communications of the ACM*, 52(12):90–98, 2009.

[147] R. Jeffries, A. Anderson, and C. Hendrickson. *Extreme Programming Installed*. Addison Wesley, 2000.

[148] A. Jermakovics, R. Moser, A. Sillitti, and G. Succi. Visualizing software evolution with Lagrein. In *Companion to the ACM Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA)*, pages 749–750. ACM Press, 2008.

[149] A. Jermakovics, M. Scotto, A. Sillitti, and G. Succi. Lagrein: Visualizing user requirements and development effort. In *Proceedings of the IEEE International Conference on Program Comprehension (ICPC)*, pages 293–296. IEEE Press, 2007.

[150] A. Jermakovics, M. Scotto, and G. Succi. Lagrein: tracking the software development process. In *Companion to the ACM Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA)*, pages 882–883. ACM Press, 2007.

[151] B. Johnson and B. Shneiderman. Tree-maps: a space-filling approach to the visualization of hierarchical information structures. In *Proceedings of the IEEE Conference on Visualization (VIS)*, pages 284–291. IEEE Press, 1991.

[152] S. Jordà, G. Geiger, M. Alonso, and M. Kaltenbrunner. The reactable: exploring the synergy between live music performance and tabletop tangible interfaces. In *Proceedings of the International Conference on Tangible and Embedded Interaction (TEI)*, pages 139–146. ACM Press, 2007.

[153] S. Jorda, M. Kaltenbrunner, G. Geiger, and R. Bencina. The reactable*. In *Proceedings of the International Computer Music Conference (ICMC)*, 2005.

[154] S. Jucknath-John and D. Graf. Icon graphs: visualizing the evolution of large class models. In *Proceedings of the ACM Symposium on Software Visualization (SOFTVIS)*, pages 167–168. ACM Press, 2006.

[155] S. Jucknath-John, D. Graf, and G. Taentzer. Evolutionary layout: preserving the mental map during the development of class models. In *Proceedings of the ACM Symposium on Software Visualization (SOFTVIS)*, pages 165–166. ACM Press, 2006.

[156] Y. Jung, J. Keil, J. Behr, S. Webel, M. Zöllner, T. Engelke, H. Wuest, and M. Becker. Adapting X3D for multi-touch environments. In *Proceedings of the ACM International Conference on 3D Web Technology (Web3D)*, pages 27–30. ACM Press, 2008.

[157] M. Kaltenbrunner and R. Bencina. reactivision: a computer-vision framework for table-based tangible interaction. In *Proceedings of the International Conference on Tangible and Embedded Interaction (TEI)*, pages 69–74. ACM Press, 2007.

[158] M. Kaltenbrunner, T. Bovermann, R. Bencina, and E. Costanza. Tuio - a protocol for table based tangible user interfaces. In *Proceedings of the International Workshop on Gesture in Human-Computer Interaction and Simulation*, 2005.

[159] M. Kaltenbrunner, G. Geiger, and S. Jorda. Dynamic patches for live musical performance. In *Proceedings of the International Conference on New Interfaces for Musical Expression (NIME)*, pages 19–22, 2004.

[160] M. Kaltenbrunner, S. Jorda, G. Geiger, and M. Alonso. The reactable*: A collaborative musical instrument. In *Proceedings of the IEEE International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises*, pages 406–411. IEEE Press, 2006.

[161] V. Karavirta, A. Korhonen, and L. Malmi. Taxonomy of algorithm animation languages. In *Proceedings of the ACM Symposium on Software Visualization (SOFTVIS)*, pages 77–85. ACM Press, 2006.

[162] L. Keown. Virtual 3D worlds for enhanced software visualization. Master's thesis, University of Canterbury, 2000.

[163] R. Khaled, P. Barr, H. Johnston, and R. Biddle. Let's clean up this mess: exploring multi-touch collaborative play. In *Proceedings of the ACM International Conference Extended Abstracts on Human Factors in Computing Systems (CHI)*, pages 4441–4446. ACM Press, 2009.

[164] K. Kim, W. Javed, C. Williams, N. Elmqvist, and P. Irani. Hugin: a framework for awareness and coordination in mixed-presence collaborative information visualization. In *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces (ITS)*. ACM Press, 2010.

[165] S. Kim, J. Son, G. Lee, H. Kim, and W. Lee. Tapboard: making a touch screen keyboard more touchable. In *Proceedings of the ACM International Conference on Human Factors in Computing Systems (CHI)*, pages 553–562. ACM Press, 2013.

[166] T. Kim and P. A. Fishwick. A 3D XML-based customized framework for dynamic models. In *Proceedings of the ACM International Conference on 3D Web Technology (Web3D)*, pages 103–109. ACM Press, 2002.

[167] C. Knight and M. Munro. Comprehension with[in] virtual environment visualisations. In *Proceedings of the IEEE International Workshop on Program Comprehension (IWPC)*, pages 4–11. IEEE Press, 1999.

[168] C. Knight and M. Munro. Virtual but visible software. In *Proceedings of the IEEE International Conference on Information Visualization (IV)*, pages 198–205. IEEE Press, 2000.

[169] C. Knight and M. Munro. Visualising Java uncertainty. In *Java/Jini Technologies, in the Multimedia Networks and Management Program of The Convergence of Information Technologies and Communication (ITCOM)*, volume 4521. SPIE, 2001.

[170] A. Ko, R. DeLine, and G. Venolia. Information needs in collocated software development teams. In *Proceedings of the International Conference on Software Engineering (ICSE)*, pages 344–353. IEEE Press, 2007.

[171] H. Koike. An application of three-dimensional visualization to object-oriented programming. In *Proceedings of the Working Conference on Advanced Visual Interfaces (AVI)*, pages 180–192. World Scientific, 1992.

[172] H. Koike. Three-dimensional software visualization: A framework and its applications. In *Proceedings of the International Conference of the Computer Graphics Society on Visual Computing*, pages 151–170. Springer Verlag, 1992.

[173] H. Koike. The role of another spatial dimension in software visualization. *ACM Transactions on Information Systems (TOIS)*, 11(3):266–286, 1993.

[174] H. Koike and H.-C. Chu. Vrcs: Integrating version control and module management using interactive three-dimensional graphics. In *Proceedings of the IEEE Symposium on Visual Languages (VL)*, pages 170–175, 1997.

[175] H. Koike and H.-C. Chu. How does 3-D visualization work in software engineering?: empirical study of a 3-D version/module visualization system. In *Proceedings of the International Conference on Software Engineering (ICSE)*, pages 516–519. IEEE Press, 1998.

[176] R. Koschke. Software visualization for reverse engineering. In *Revised Lectures on Software Visualization, International Seminar*, pages 138–150. Springer Verlag, 2002.

[177] R. Koschke. Software visualization in software maintenance, reverse engineering, and re-engineering: a research survey. *Journal of Software Maintenance*, 15(2):87–109, 2003.

[178] B. Kot, B. Wuensche, J. Grundy, and J. Hosking. Information visualisation utilising 3D computer game engines - case study: A source code comprehension tool. In *Proceedings of the ACM New Zealand Conference on Computer-Human Interaction (CHINZ)*, pages 53–60. ACM Press, 2005.

[179] M. W. Krueger, T. Gionfriddo, and K. Hinrichsen. Videoplace—an artificial reality. *SIGCHI Bull.*, 16(4):35–40, 1985.

[180] R. Kruger, S. Carpendale, S. Scott, and S. Greenberg. How people use orientation on tables: comprehension, coordination and communication. In *Proceedings of the International Conference on Supporting Group Work (GROUP)*, pages 369–378. ACM Press, 2003.

[181] G. Langelier, H. Sahraoui, and P. Poulin. Visualization-based analysis of quality for large-scale software systems. In *Proceedings of the IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 214–223. ACM Press, 2005.

[182] G. Langelier, H. Sahraoui, and P. Poulin. Animation coherence in representing software evolution. In *Proceedings of the ECOOP Workshop on Quantitative Approaches in Object-Oriented Software Engineering (QAOOSE)*, pages 41–50, 2006.

[183] G. Langelier, H. Sahraoui, and P. Poulin. Exploring the evolution of software quality with animated visualization. In *Proceedings of the IEEE Symposium on Visual Languages - Human Centric Computing (VLHCC)*, pages 13–20. IEEE Press, 2008.

[184] M. Lanza. Codecrawler - lessons learned in building a software visualization tool. In *Proceedings of the European Conference on Software Maintenance and Reengineering (CSMR)*, pages 409–418. IEEE Press, 2003.

[185] M. Lanza and S. Ducasse. Polymetric views-a lightweight visual approach to reverse engineering. *IEEE Transactions on Software Engineering*, 29(9):782–795, 2003.

[186] M. Lanza and S. Ducasse. *Tools for Software Maintenance and Reengineering*, chapter CodeCrawler - An Extensible and Language Independent 2D and 3D Software Visualization Tool, pages 74–94. RCOST Software Technology Series, 2005.

[187] M. Lanza and R. Marinescu. *Object-Oriented Metrics in Practice*. Springer Verlag, 2006.

[188] U. Laufs, C. Ruff, and J. Zibuschka. MT4j a cross-platform multi-touch development framework. In *Proceedings of the Workshop on Engineering Patterns for Multi-Touch Interfaces at Symposium Engineering Interactive Computing Systems (EICS)*. ACM Press, 2010.

[189] B. Lee, C. Plaisant, C. Parr, J.-D. Fekete, and N. Henry. Task taxonomy for graph visualization. In *Proceedings of the AVI Workshop on BEyond time and errors: novel evaLuation methods for Information Visualization (BELIV)*, pages 1–5. ACM Press, 2006.

[190] Y. Y. Lee, N. Chen, and R. Johnson. Drag-and-drop refactoring: intuitive and efficient program transformation. In *Proceedings of the International Conference on Software Engineering (ICSE)*, pages 23–32. IEEE Press, 2013.

[191] G. J. Lepinski, T. Grossman, and G. Fitzmaurice. The design and evaluation of multitouch marking menus. In *Proceedings of the ACM International Conference on Human Factors in Computing Systems (CHI)*, pages 2233–2242. ACM Press, 2010.

[192] C. Lewerentz and A. Noack. *Graph Drawing Software*, chapter CrocoCosmos - 3D Visualization of Large Object-Oriented Programs, pages 279–297. Springer Verlag, 2003.

[193] C. Lewerentz and F. Simon. Metrics-based 3D visualization of large object-oriented programs. In *Proceedings of the International Workshop on Visualizing Software for Understanding and Analysis (VISSOFT)*, pages 70–77. IEEE Press, 2002.

[194] H. Lieberman. A three-dimensional representation for program execution. In *Proceedings of the IEEE Workshop on Visual Languages (VL)*, pages 111–116. IEEE Press, 1989.

[195] W. Lowe, M. Ericsson, J. Lundberg, T. Panas, and N. Pettersson. Vizzanalyzer - a software comprehension framework. In *Proceedings of the International Conference on Software Engineering Research and Practice (SERPS)*, pages 127–136. CSREA Press, 2003.

[196] M. Lungu and M. Lanza. Softwarenaut: cutting edge visualization. In *Proceedings of the ACM Symposium on Software Visualization (SOFTVIS)*, pages 179–180. ACM Press, 2006.

[197] M. Lungu and M. Lanza. Softwarenaut: Exploring hierarchical system decompositions. In *Proceedings of the Conference on Software Maintenance and Reengineering (CSMR)*, pages 351–354. IEEE Press, 2006.

[198] J. Maletic and A. Marcus. CFB: a call for benchmarks - for software visualization. In *Proceedings of the International Workshop on Visualizing Software for Understanding and Analysis (VISSOFT)*, pages 113–116. IEEE Press, 2003.

[199] J. Maletic, A. Marcus, and M. Collard. A task oriented view of software visualization. In *Proceedings of the International Workshop on Visualizing Software for Understanding and Analysis (VISSOFT)*, pages 32–40. IEEE Press, 2002.

[200] S. Malik and J. Laszlo. Visual touchpad: a two-handed gestural input device. In *Proceedings of the International Conference on Multimodal Interfaces (ICMI)*, pages 289–296. ACM Press, 2004.

[201] A. Marcus, D. Comorski, and A. Sergeyev. Supporting the evolution of a software visualization tool through usability studies. In *Proceedings of the IEEE International Workshop on Program Comprehension (IWPC)*, pages 307–316. IEEE Press, 2005.

[202] A. Marcus, L. Feng, and J. Maletic. 3D representations for software visualization. In *Proceedings of the ACM Symposium on Software Visualization (SOFTVIS)*, pages 27–36. ACM Press, 2003.

[203] A. Marcus, L. Feng, and J. Maletic. Comprehension of software analysis data using 3D visualization. In *Proceedings of the IEEE International Workshop on Program Comprehension (IWPC)*, pages 105–114. IEEE Press, 2003.

[204] D. Markusson. Interface development of a multi-touch photo browser. Master's thesis, Umea University, 2008.

[205] J. Matejka, T. Grossman, J. Lo, and G. Fitzmaurice. The design and evaluation of multi-finger mouse emulation techniques. In *Proceedings of the ACM International Conference on Human Factors in Computing Systems (CHI)*, pages 1073–1082. ACM Press, 2009.

[206] N. Matsushita, Y. Ayatsuka, and J. Rekimoto. Dual touch: a two-handed interface for pen-based pdas. In *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST)*, pages 211–212. ACM Press, 2000.

[207] N. Matsushita and J. Rekimoto. Holowall: designing a finger, hand, body, and object sensitive wall. In *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST)*, pages 209–210. ACM Press, 1997.

[208] R. Mazza and A. Berre. Focus group methodology for evaluating information visualization techniques and tools. In *Proceedings of the IEEE International Conference on Information Visualization (IV)*, pages 74–80. IEEE Press, 2007.

[209] W. McGrath, B. Bowman, D. McCallum, J. D. Hincapié-Ramos, N. Elmqvist, and P. Irani. Branch-explore-merge: facilitating real-time revision control in collaborative visual exploration. In *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces (ITS)*, pages 235–244. ACM Press, 2012.

[210] P. McIntosh. *X3D-UML: User-Centred Design, Implementation and Evaluation of 3D UML Using X3D*. PhD thesis, Royal Melbourne Institute of Technology (RMIT), 2010.

[211] P. McIntosh, M. Hamilton, and R. van Schyndel. X3D-UML: enabling advanced UML visualisation through X3D. In *Proceedings of the ACM International Conference on 3D Web Technology (Web3D)*, pages 135–142. ACM Press, 2005.

[212] C. Mesnage and M. Lanza. White coats: Web-visualization of evolving software in 3D. In *Proceedings of the International Workshop on Visualizing Software for Understanding and Analysis (VISSOFT)*, pages 40–45. IEEE Press, 2005.

[213] M. Meyer, T. Girba, and M. Lungu. Mondrian: An agile information visualization toolkit. In *Proceedings of the ACM Symposium on Software Visualization (SOFTVIS)*, pages 135–144. ACM Press, 2006.

[214] I. Mistrk, J. Grundy, A. van der Hoek, and J. Whitehead, editors. *Collaborative Software Engineering*. Springer Verlag, 2010.

[215] R. Morgan and F. Maurer. Maseplanner: A card-based distributed planning tool for agile teams. In *Proceedings of the International Conference on Global Software Engineering (ICGSE)*, pages 132–138. IEEE Press, 2006.

[216] R. Morgan, J. Walny, H. Kolenda, E. Ginez, and F. Maurer. Using horizontal displays for distributed and collocated agile planning. In *Proceedings of the International Conference on Agile Processes in Software Engineering and Extreme Programming (XP)*, pages 38–45. Springer Verlag, 2007.

[217] L. Muller. Multi-touch displays: design, application and performance evaluation. Master's thesis, University of Amsterdam, 2008.

[218] S. Muller, M. Wursch, T. Fritz, and H. Gall. An approach for collaborative code reviews using multi-touch technology. In *Proeedings of the Workshop on Cooperative and Human Aspects of Software Engineering (CHASE)*, pages 93 –99, 2012.

[219] S. Muller, M. Wursch, P. Schoni, G. Ghezzi, E. Giger, and H. Gall. Tangible software modeling with multi-touch technology. In *Proeedings of the Workshop on Cooperative and Human Aspects of Software Engineering (CHASE)*, pages 100 –104, 2012.

[220] M. Najork. Web-based algorithm animation. In *Proceedings of the Conference on Design Automation (DAC)*, pages 506–511. ACM Press, 2001.

[221] M. Najork and M. Brown. Three-dimensional web-based algorithm animations. Technical Report SRC-RR-170, Compaq Systems Research Centre, 2001.

[222] M. Nebeling and M. Norrie. jQMultiTouch: lightweight toolkit and development framework for multi-touch/multi-device web interfaces. In *Proceedings of the ACM Symposium on Engineering Interactive Computing Systems (EICS)*. ACM Press, 2012.

[223] K. Nesbitt. Using guidelines to assist in the visualisation design process. In *Proceedings of the Asia-Pacific Symposium on Information Visualisation (APVIS)*, pages 115–123. Australian Computer Society, Inc, 2005.

[224] J. Nielsen. *Usability Engineering*. Morgan Kaufmann, 1994.

[225] J. Noble. *Abstract Program Visualisation*. PhD thesis, Victoria University of Wellington, 1996.

[226] J. Noble and L. Groves. Tarraingím - a program animation environment. In *Proceedings of the New Zealand Computer Science Conference*, 1991.

[227] J. Noble, L. Groves, and R. Biddle. Object oriented program visualisation in tarraingím. *Australian Computing Journal*, 27(4), 1995.

[228] C. North, T. Dwyer, B. Lee, D. Fisher, P. Isenberg, G. Robertson, and K. Inkpen. Understanding multi-touch manipulation for surface computing. In *Proceedings of the IFIP TC International Conference on Human-Computer Interaction (INTERACT)*, pages 236–249. Springer Verlag, 2009.

[229] M. Ogawa and K.-L. Ma. code_swarm: A design study in organic software visualization. In *Proceedings of the IEEE Symposium on Information Visualization (INFOVIS)*. ACM Press, 2009.

[230] M. Oudshoorn, H. Widjaja, and S. Ellershaw. Aspects and taxonomy of program visualisation. In *Software Visualisation*, pages 3–26. World Scientific, 1996.

[231] T. Panas. Signature visualization of software binaries. In *Proceedings of the ACM Symposium on Software Visualization (SOFTVIS)*, pages 185–188. ACM Press, 2008.

[232] T. Panas, R. Berrigan, and J. Grundy. A 3D metaphor for software production visualization. In *Proceedings of the IEEE Symposium on Information Visualization (INFOVIS)*, pages 314–319. IEEE Press, 2003.

[233] T. Panas, R. Lincke, and W. Lowe. Online-configuration of software visualizations with Vizz3D. In *Proceedings of the ACM Symposium on Software Visualization (SOFTVIS)*, pages 173–182. ACM Press, 2005.

[234] G. Parker, G. Franck, and C. Ware. Visualization of large nested graphs in 3D: navigation and interaction. *Journal of Visual Languages and Computing*, 9(3):299–317, 1998.

[235] C. Parnin and C. Görg. Lightweight visualizations for inspecting code smells. In *Proceedings of the ACM Symposium on Software Visualization (SOFTVIS)*, pages 171–172. ACM Press, 2006.

[236] C. Parnin, C. Görg, and O. Nnadi. A catalogue of lightweight visualizations to support code smell inspection. In *Proceedings of the ACM Symposium on Software Visualization (SOFTVIS)*, pages 77–86. ACM Press, 2008.

[237] C. Parnin, C. Görg, and S. Rugaber. Codepad: interactive spaces for maintaining concentration in programming environments. In *Proceedings of the ACM Symposium on Software Visualization (SOFTVIS)*. ACM Press, 2010.

[238] W. D. Pauw, H. Andrade, and L. Amini. Streamsight: a visualization tool for large-scale streaming applications. In *Proceedings of the ACM Symposium on Software Visualization (SOFTVIS)*, pages 125–134. ACM Press, 2008.

[239] W. D. Pauw, R. Helm, D. Kimelman, and J. Vlissides. Visualizing the behavior of object-oriented systems. In *Proceedings of ACM Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA)*, pages 326–337. ACM Press, 1993.

[240] W. D. Pauw, E. Jensen, N. Mitchell, G. Sevitsky, J. Vlissides, and J. Yang. Visualizing the execution of Java programs. In *Revised Lectures on Software Visualization, International Seminar*, pages 151–162. Springer Verlag, 2002.

[241] W. D. Pauw, D. Kimelman, and J. Vlissides. Modeling object-oriented program execution. In *Proceedings of the European Conference on Object-Oriented Programming (ECOOP)*, pages 163–182. Springer Verlag, 1994.

[242] W. D. Pauw, D. Kimelman, and J. Vlissides. *Software Visualization*, chapter Visualizing Object-Oriented Software Execution, pages 329–346. MIT Press, 1998.

[243] W. D. Pauw, S. Krasikov, and J. Morar. Execution patterns for visualizing web services. In *Proceedings of the ACM Symposium on Software Visualization (SOFTVIS)*, pages 37–45. ACM Press, 2006.

[244] W. D. Pauw, N. Mitchell, M. Robillard, G. Sevitsky, and H. Srinivasan. Drive-by analysis of running programs. In *Proceedings of the ICSE Workshop on Software Visualization*, pages 27–32, 2001.

[245] W. D. Pauw and G. Sevitski. Visualizing reference patterns for solving memory leaks in Java. In *Proceedings of the European Conference on Object-Oriented Programming (ECOOP)*, pages 116–134. Springer Verlag, 1999.

[246] P. Peltonen, E. Kurvinen, A. Salovaara, G. Jacucci, T. Ilmonen, J. Evans, A. Oulasvirta, and P. Saarikko. It's mine, don't touch!: interactions at a large multi-touch display in a city centre. In *Proceedings of the ACM International Conference on Human Factors in Computing Systems (CHI)*, pages 1285–1294. ACM Press, 2008.

[247] T. Perl and S. Kögl. Adaptation and evaluation of numpty physics for multi-touch multi-player interaction. Bachelors Thesis, Technical University of Vienna, 2009.

[248] C. Plaisant. The challenge of information visualization evaluation. In *Proceedings of the Working Conference on Advanced Visual Interfaces (AVI)*, pages 109–116. ACM Press, 2004.

[249] B. Price, R. Baecker, and I. Small. A principled taxonomy of software visualization. *Journal of Visual Languages and Computing*, 4(2):211–266, 1993.

[250] B. Price, R. Baecker, and I. Small. *Software Visualization*, chapter An Introduction to Software Visualization, pages 3–27. MIT Press, 1998.

[251] B. Price, I. Small, and R. Baecker. A taxonomy of software visualisation. In *Proceedings of the International Conference on System Sciences*, pages 597–606, 1992.

[252] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2013.

[253] O. Radfelder and M. Gogolla. On better understanding UML diagrams through interactive three-dimensional visualization and animation. In *Proceedings of the Working Conference on Advanced Visual Interfaces (AVI)*, pages 292–295. ACM Press, 2000.

[254] C. Reas and B. Fry. *Processing: A Programming Handbook for Visual Designers and Artists*. MIT Press, 2007.

[255] S. Reiss. A framework for abstract 3D visualization. In *Proceedings of the IEEE Symposium on Visual Languages (VL)*, pages 108–115. IEEE Press, 1993.

[256] S. Reiss. 3-D visualization of program information. In *Proceedings of Graph Drawing*, pages 12–24. Springer Verlag, 1994.

[257] S. Reiss. An engine for the 3D visualization of program information. *Journal of Visual Languages and Computing*, 6(3):299–323, 1995.

[258] S. Reiss. Bee/Hive: A software visualization back end. In *Proceedings of the ICSE Workshop on Software Visualization*, pages 44–48. IEEE Press, 2001.

[259] S. Reiss. An overview of BLOOM. In *Proceedings of the ACM Workshop on Program Analysis for Software Tools and Engineering (PASTE)*, pages 2–5. ACM Press, 2001.

[260] S. Reiss. A visual query language for software visualization. In *Proceedings of the IEEE Symposium on Human Centric Computing Languages and Environments (HCC)*, pages 80–82. IEEE Press, 2002.

[261] S. Reiss. Event-based performance analysis. In *Proceedings of the IEEE International Workshop on Program Comprehension (IWPC)*, pages 74–83. IEEE Press, 2003.

[262] S. Reiss. Jive: visualizing Java in action demonstration description. In *Proceedings of the International Conference on Software Engineering (ICSE)*, pages 820–821. IEEE Press, 2003.

[263] S. Reiss. Visualizing Java in action. In *Proceedings of the ACM Symposium on Software Visualization (SOFTVIS)*, pages 57–65. ACM Press, 2003.

[264] S. Reiss. Visualizing program execution using user abstractions. In *Proceedings of the ACM Symposium on Software Visualization (SOFTVIS)*, pages 125–134. ACM Press, 2006.

[265] S. Reiss. Controlled dynamic performance analysis. In *Proceedings of the International Workshop on Software and Performance (WOSP)*, pages 43–54. ACM Press, 2008.

[266] S. Reiss. Dynamic detection of event handlers. In *Proceedings of the ICSE Workshop on Dynamic Analysis (WODA)*, pages 1–7. ACM Press, 2008.

[267] S. Reiss. Dyvise: Performance analysis of production systems. In *Proceedings of the International Conference on Software Engineering (ICSE)*. IEEE Press, 2009.

[268] S. Reiss. Visualizing the Java heap. In *Proceedings of the IEEE International Conference on Software Maintenance (ICSM)*. IEEE Press, 2009.

[269] S. Reiss. Visualizing the Java heap to detect memory problems. In *Proceedings of the International Workshop on Visualizing Software for Understanding and Analysis (VISSOFT)*. IEEE Press, 2009.

[270] S. Reiss and M. Renieris. Generating Java trace data. In *Proceedings of the ACM Conference on Java Grande*, pages 71–77. ACM Press, 2000.

[271] S. Reiss and M. Renieris. Languages for dynamic instrumentation. In *Proceedings of the ICSE Workshop on Dynamic Analysis (WODA)*, pages 41–45, 2003.

[272] S. Reiss and M. Renieris. *Software Visualization: From Theory to Practice*, chapter The Bloom Software Visualization System, pages 311–357. Kluwer Academic Publishers, 2003.

[273] S. Reiss and M. Renieris. Jove: Java as it happens. In *Proceedings of the ACM Symposium on Software Visualization (SOFTVIS)*, pages 115–124. ACM Press, 2005.

[274] J. Rekimoto. Smartskin: an infrastructure for freehand manipulation on interactive surfaces. In *Proceedings of the ACM International Conference on Human Factors in Computing Systems (CHI)*, pages 113–120. ACM Press, 2002.

[275] J. Rekimoto and N. Matsushita. Perceptual surfaces: Towards a human and object sensitive interactive display. In *Proceedings of Workshop on Perceptural User Interfaces (PUI)*, 1997.

[276] S. Richter, C. Holz, and P. Baudisch. Bootstrapper: Recognizing tabletop users by their shoes. In *Proceedings of the ACM International Conference on Human Factors in Computing Systems (CHI)*, pages 1249–1252. ACM Press, 2012.

[277] J. Rilling and S. Mudur. On the use of metaballs to visually map source code structures and analysis results onto 3D space. In *Proceedings of the Working Conference on Reverse Engineering (WCRE)*, pages 299–308. IEEE Press, 2002.

[278] J. Rilling and S. Mudur. 3D visualization techniques to support slicing-based program comprehension. *Computers & Graphics*, 29(3):311–329, 2005.

[279] J. Rilling, J. Wang, and S. Mudur. Metaviz - issues in software visualizing beyond 3D. In *Proceedings of the International Workshop on Visualizing Software for Understanding and Analysis (VISSOFT)*, pages 92–97. IEEE Press, 2003.

[280] G. Robertson, M. Czerwinski, P. Baudisch, B. Meyers, D. Robbins, G. Smith, and D. Tan. The large-display user experience. *IEEE Computer Graphics Applications*, 25(4):44–51, 2005.

[281] G.-C. Roman and K. Cox. A taxonomy of program visualization systems. *IEEE Computer*, 26(12):11–24, 1993.

[282] G.-C. Roman, K. Cox, D. Wilcox, and J. Plun. Pavane: A system for declarative visualization of concurrent computations. *Journal of Visual Languages and Computing*, 3(2):161–193, 1992.

[283] H. Ruan, C. Anslow, S. Marshall, and J. Noble. Exploring the inventor's paradox: applying Jigsaw to software visualization. In *Proceedings of the ACM Symposium on Software Visualization (SOFTVIS)*, pages 83–92. ACM Press, 2010.

[284] K. Ryall, C. Forlines, C. Shen, M. R. Morris, and K. Everitt. Experiences with and observations of direct-touch tabletops. In *Proceedings of the IEEE International Workshop on Horizontal Interactive Human Computer Systems (TABLETOP)*, pages 89–96. IEEE Press, 2006.

[285] A. Savidis, P. Papadakos, and G. Zargianakis. Rapid visual design with semantics encoding through 3D CRC cards. In *Proceedings of the ACM Symposium on Software Visualization (SOFTVIS)*, pages 193–196. ACM Press, 2008.

[286] T. Schäfer and M. Mezini. Towards more flexibility in software visualization tools. In *Proceedings of the International Workshop on Visualizing Software for Understanding and Analysis (VISSOFT)*, pages 1–6. IEEE Press, 2005.

[287] J. Schöning, P. Brandl, F. Daiber, F. Echtler, O. Hilliges, J. Hook, M. Lchtefeld, N. Motamedi, L. Muller, P. Olivier, T. Roth, and U. von Zadow. Multi-touch surfaces: A technical guide. Technical Report TUM-I0833, University of Munster, 2008.

[288] J. Schöning, F. Daiber, A. Krüger, and M. Rohs. Using hands and feet to navigate and manipulate spatial data. In *Proceedings of the ACM International Conference Extended Abstracts on Human Factors in Computing Systems (CHI)*, pages 4663–4668. ACM Press, 2009.

[289] S. Scott. *Territoriality in Collaborative Tabletop Workspaces*. PhD thesis, University of Calgary, 2005.

[290] S. Scott, S. Carpendale, and S. Habelski. Storage bins: Mobile storage for collaborative tabletop displays. *IEEE Computer Graphics Applications*, 25(4):58–65, 2005.

[291] S. Scott, S. Carpendale, and K. Inkpen. Territoriality in collaborative tabletop workspaces. In *Proceedings of the ACM Conference on Computer Supported Cooperative work (CSCW)*, pages 294–303. ACM Press, 2004.

[292] S. Scott, K. Grant, and R. Mandryk. System guidelines for co-located, collaborative work on a tabletop display. In *Proceedings of the European Conference on Computer Supported Cooperative Work (ECSCW)*, pages 159–178. Kluwer Academic Publishers, 2003.

[293] M. Sensalire, P. Ogao, and A. Telea. Visualizing object-oriented software: Towards a point of reference for developing tools for industry. In *Proceedings of the International Workshop on Visualizing Software for Understanding and Analysis (VISSOFT)*, pages 26–29. IEEE Press, 2007.

[294] M. Sensalire, P. Ogao, and A. Telea. Classifying desirable features of software visualization tools for corrective maintenance. In *Proceedings of the ACM Symposium on Software Visualization (SOFTVIS)*, pages 87–90. ACM Press, 2008.

[295] M. Sensalire, P. Ogao, and A. Telea. Evaluation of software visualization tools: Lessons learned. In *Proceedings of the International Workshop on Visualizing Software for Understanding and Analysis (VISSOFT)*, pages 19–26. IEEE Press, 2009.

[296] G. Sevitsky, W. D. Pauw, and R. Konuru. An information exploration tool for performance analysis of Java programs. In *Proceedings of the IEEE International Conference on Technology of Object-Oriented Languages and Systems (TOOLS)*, pages 85–101, March 2001.

[297] H. Shah, C. Görg, and M. J. Harrold. Visualization of exception handling constructs to support program understanding. In *Proceedings of the ACM Symposium on Software Visualization (SOFTVIS)*, pages 19–28. ACM Press, 2008.

[298] B. Sharif and J. Maletic. The effect of layout on the comprehension of UML class diagrams: A controlled experiment. In *Proceedings of the International Workshop on Visualizing Software for Understanding and Analysis (VISSOFT)*, pages 11–18. IEEE Press, 2009.

[299] C. Shen, F. Vernier, C. Forlines, and M. Morris. Diamondspin: An extensible toolkit for around-the-table interaction. In *Proceedings of the ACM International Conference on Human Factors in Computing Systems (CHI)*, pages 167–174. ACM Press, 2004.

[300] B. Shneiderman. Tree visualization with tree-maps: 2-D space-filling approach. *ACM Transactions on Graphics (TOG)*, 11(1):92–99, 1992.

[301] B. Shneiderman. The eyes have it: A task by data type taxonomy for information visualizations. In *Proceedings of the IEEE Symposium on Visual Languages (VL)*, pages 336–343. IEEE Press, 1996.

[302] J. Sillito, G. Murphy, and K. D. Volder. Questions programmers ask during software evolution tasks. In *Proceedings of the ACM International Conference on Foundations of Software Engineering (FSE)*, pages 23–34. ACM Press, 2006.

[303] J. Singer and C. Kirkham. Visualized adaptive runtime subsystems. In *Proceedings of the ACM Symposium on Software Visualization (SOFTVIS)*, pages 195–196. ACM Press, 2006.

[304] D. Smith, N. Graham, D. Holman, and J. Borchers. Low-cost malleable surfaces with multi-touch pressure sensitivity. In *Proceedings of the IEEE International Workshop on Horizontal Interactive Human Computer Systems (TABLETOP)*, pages 205–208. IEEE Press, 2007.

[305] E. Smith, R. Loftin, E. Murphy-Hill, C. Bird, and T. Zimmermann. Improving developer participation rates in surveys. In *Proceedings of the Workshop on Cooperative and Human Aspects of Software Engineering (CHASE)*, pages 89–92, 2013.

[306] R. Smith and W. Piekarski. Public and private workspaces on tabletop displays. In *Proceedings of the Australasian Conference on User Interfaces (AUIC)*, pages 51–54. Australian Computer Society, Inc, 2008.

[307] I. Sommerville. *Software Engineering*. Addison Wesley, 9 edition, 2010.

[308] A. Soro, S. Iacolina, R. Scateni, and S. Uras. Evaluation of user gestures in multi-touch interaction: a case study in pair-programming. In *Proceedings of the International Conference on Multimodal Interfaces (ICMI)*, pages 161–168. ACM Press, 2011.

[309] R. Spence. *Information Visualization: Design For Interaction*. Addison Wesley, 2007.

[310] J. Stasko. *Tango: A Framework and System for Algorithm Animation*. PhD thesis, Brown University, May 1989.

[311] J. Stasko. Tango: A framework and system for algorithm animation. *IEEE Computer*, 23(9):27–39, 1990.

[312] J. Stasko, M. Brown, and B. Price. *Software Visualization*. MIT Press, 1997.

[313] J. Stasko and E. Kraemer. A methodology for building application-specific visualizations of parallel programs. *Journal of Parallel and Distributed Computing*, 18(2):258–264, 1993.

[314] J. Stasko and J. Wehrli. Three-dimensional computation visualization. In *Proceedings of the IEEE Symposium on Visual Languages (VL)*, pages 100–107. IEEE Press, 1993.

[315] T. Storer and I. Duncan. 3D animation of Java program execution for teaching object oriented concepts. In *Proceedings of the International Conference on Visualisation, Imaging and Image Processing (VIIP)*, pages 76–81. ACTA Press, 2007.

[316] M.-A. Storey, C. Bennett, I. Bull, and D. German. Remixing visualization to support collaboration in software maintenance. In *Proceedings of the Frontiers of Software Maintenance (FoSM)*, pages 139–148. IEEE Press, 2008.

[317] M.-A. Storey, D. Cubranic, and D. German. On the use of visualization to support awareness of human activities in software development: a survey and a framework. In *Proceedings of the ACM Symposium on Software Visualization (SOFTVIS)*, pages 193–202. ACM Press, 2005.

[318] J. Sundararaman and G. Back. HDPV: interactive, faithful, in-vivo runtime state visualization for C/C++ and Java. In *Proceedings of the ACM Symposium on Software Visualization (SOFTVIS)*, pages 47–56. ACM Press, 2008.

[319] D. Tan, D. Gergle, P. Scupelli, and R. Pausch. Physically large displays improve path integration in 3D virtual navigation tasks. In *Proceedings of the ACM International Conference on Human Factors in Computing Systems (CHI)*, pages 439–446. ACM Press, 2004.

[320] A. Tang, M. Tory, B. Po, P. Neumann, and S. Carpendale. Collaborative coupling over tabletop displays. In *Proceedings of the ACM International Conference on Human Factors in Computing Systems (CHI)*, pages 1181–1190. ACM Press, 2006.

[321] A. Teiche, A. Rai, C. Yanc, C. Moore, D. Solms, G. Cetin, J. Riggio, N. Ramseyer, P. D'Intino, L. Muller, R. Khoshabeh, R. Bedi, M. T. Bintahir, T. Hansen, T. Roth, and S. Sandler. Multitouch technologies. http://nuigroup.com/, 2009.

[322] A. Telea, H. Hoogendorp, O. Ersoy, and D. Reniers. Extraction and visualization of call dependencies for large C/C++ code bases: A comparative study. In *Proceedings of the International Workshop on Visualizing Software for Understanding and Analysis (VISSOFT)*, pages 81–88. IEEE Press, 2009.

[323] A. Telea, A. Maccari, and C. Riva. An open toolkit for prototyping reverse engineering visualizations. In *Proceedings of the Symposium on Data Visualisation (VISSYM)*, pages 241–249. Eurographics Association, 2002.

[324] A. Telea and L. Voinea. An interactive reverse engineering environment for large-scale C++ code. In *Proceedings of the ACM Symposium on Software Visualization (SOFTVIS)*, pages 67–76. ACM Press, 2008.

[325] E. Tempero, C. Anslow, J. Dietrich, T. Han, J. Li, M. Lumpe, H. Melton, and J. Noble. Qualitas corpus: A curated collection of Java code for empirical studies. In *Proceedings of the Asia Pacific Software Engineering Conference (APSEC)*, pages 336–345, 2010.

[326] A. Teyseyre and M. Campo. An overview of 3D software visualization. *IEEE Transactions on Visualization and Computer Graphics*, 15(1):87–105, 2009.

[327] R. Theron, A. Gonzalez, and F. Garcia. Supporting the understanding of the evolution of software items. In *Proceedings of the ACM Symposium on Software Visualization (SOFTVIS)*, pages 189–192. ACM Press, 2008.

[328] B. Thompson, D. Pearce, C. Anslow, and G. Haggard. Visualizing the computation tree of the tutte polynomial. In *Proceedings of the ACM Symposium on Software Visualization (SOFTVIS)*, pages 211–212. ACM Press, 2008.

[329] M. Thörnlund. Gesture analyzing for multi-touch screen interfaces. Bachelor's Thesis Luleå University of Technology, 2007.

[330] M. Tobiasz, P. Isenberg, and S. Carpendale. Lark: Coordinating co-located collaboration with information visualization. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):10651072, 2009.

[331] A. Toney and B. Thomas. Considering reach in tangible and table top design. In *Proceedings of the IEEE International Workshop on Horizontal Interactive Human Computer Systems (TABLETOP)*, pages 57–58. IEEE Press, 2006.

[332] M. Tory and T. Moller. Evaluating visualizations: Do expert reviews work? *IEEE Computer*, 25(5):8–11, 2005.

[333] T. L. Toza, D. Garlan, J. Herbsleb, and B. Myers. Program comprehension as fact finding. In *Proceedings of the ACM International Conference on Foundations of Software Engineering (FSE)*, pages 361–370. ACM Press, 2007.

[334] T. L. Toza and B. Myers. Hard to answer questions about code. In *Proceedings of the Workshop on Evaluation and Usability of Programming Languages and Tools (PLATEAU) at SPLASH*. ACM Press, 2010.

[335] C. Treude. *The Role of Social Media Artifacts in Collaborative Software Development*. PhD thesis, University of Victoria, 2012.

[336] E. Tse, C. Shen, S. Greenberg, and C. Forlines. Enabling interaction with single user applications through speech and gestures on a multi-user tabletop. In *Proceedings of the Working Conference on Advanced Visual Interfaces (AVI)*, pages 336–343. ACM Press, 2006.

[337] E. R. Tufte. *The visual display of quantitative information*. Cheshire: Graphics Press, 1983.

[338] E. R. Tufte. *Envisioning Information*. Cheshire: Graphics Press, 1990.

[339] J. W. v. Gudenberg, A. Niederle, M. Ebner, and H. Eichelberger. Evolutionary layout of UML class diagrams. In *Proceedings of the ACM Symposium on Software Visualization (SOFTVIS)*, pages 163–164. ACM Press, 2006.

[340] E. Valiati, M. Pimenta, and C. Freitas. A taxonomy of tasks for guiding the evaluation of multidimensional visualizations. In *Proceedings of the AVI Workshop on BEyond time and errors: novel evaLuation methods for Information Visualization (BELIV)*, pages 1–6. ACM Press, 2006.

[341] F. Viegas, M. Wattenberg, and J. Feinberg. Participatory visualization with wordle. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):1137–1144, Nov. 2009.

[342] F. Viegas, M. Wattenberg, M. McKeon, F. van Ham, and J. Kriss. Harry potter and the meat-filled freezer: A case study of spontaneous usage of visualization tools. In *HICSS '08: Proceedings of the Proceedings of the 41st Annual Hawaii International Conference on System Sciences*, page 159. IEEE Press, 2008.

[343] F. Viegas, M. Wattenberg, F. van Ham, J. Kriss, and M. McKeon. ManyEyes: a site for visualization at internet scale. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1121–1128, 2007.

[344] S. Voigt, J. Bohnet, and J. Döllner. Enhancing structural views of software systems by dynamic information. In *Proceedings of the International Workshop on Visualizing Software for Understanding and Analysis (VISSOFT)*, pages 47–50. IEEE Press, 2009.

[345] L. Voinea and A. Telea. Cvsgrab: Mining the histiry of large software projects. In *Proceedings of the EuroGraphics/IEEE Symposium on Visualisation (EuroVis)*, pages 187–194. IEEE Press, 2006.

[346] L. Voinea and A. Telea. How do changes in buggy mozilla files propogate. In *Proceedings of the ACM Symposium on Software Visualization (SOFTVIS)*, pages 147–148. ACM Press, 2006.

[347] L. Voinea and A. Telea. Multiscale and multivariate visualizations of software evolution. In *Proceedings of the ACM Symposium on Software Visualization (SOFTVIS)*, pages 115–124. ACM Press, 2006.

[348] L. Voinea, A. Telea, and J. van Wijk. Cvsscan: Visualization of code evolution. In *Proceedings of the ACM Symposium on Software Visualization (SOFTVIS)*, pages 47–56. ACM Press, 2005.

[349] J. von Pilgrim and K. Duske. Gef3D: a framework for two-, two-and-a-half-, and three-dimensional graphical editors. In *Proceedings of the ACM Symposium on Software Visualization (SOFTVIS)*, pages 95–104. ACM Press, 2008.

[350] X. Wang, Y. Ghanam, and F. Maurer. From desktop to tabletop: Migrating the user interface of agileplanner. In *Proceedings of the Conference on Human-Centered Software Engineering (HCSE) and International Workshop on Task Models and Diagrams*, pages 263–270. Springer Verlag, 2008.

[351] X. Wang and F. Maurer. Tabletop agileplanner: A tabletop-based project planning tool for agile software development teams. In *Proceedings of the IEEE International Workshop on Horizontal Interactive Human Computer Systems (TABLETOP)*. IEEE Press, 2008.

[352] C. Ware. *Information Visualization: Perception for Design*. Morgan Kaufmann, 2004.

[353] C. Ware, G. Franck, M. Parkhi, and T. Dudley. Layout for visualizing large software structures in 3D. In *Proceedings of the International Conference on Visual Information Systems (VISUAL)*, pages 215–223. Springer Verlag, 1997.

[354] C. Ware, D. Hui, and G. Franck. Visualizing object oriented software in three dimensions. In *Proceedings of the Conference of the Centre for Advanced Studies on Collaborative Research (CASCON)*, pages 612–620. IBM Press, 1993.

[355] M. Weiss, R. Jennings, R. Khoshabeh, J. Borchers, J. Wagner, Y. Jansen, and J. D. Hollan. Slap widgets: bridging the gap between virtual and physical controls on tabletops. In *Proceedings of the ACM International Conference Extended Abstracts on Human Factors in Computing Systems (CHI)*, pages 3229–3234. ACM Press, 2009.

[356] P. Wellner. The digitaldesk calculator: tangible manipulation on a desk top display. In *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST)*, pages 27–33. ACM Press, 1991.

[357] P. Wellner. Interacting with paper on the digitaldesk. *Communications of the ACM*, 36(7):87–96, 1993.

[358] D. Wendlandt, M. Casado, P. Tarjan, and N. McKeown. The clack graphicsl router: Visualizing network structure. In *Proceedings of the ACM Symposium on Software Visualization (SOFTVIS)*, pages 7–15. ACM Press, 2006.

[359] W. Westerman. *Hand Tracking, Finger Identification and Chrodic Manipulation on a Multi-touch Surface*. PhD thesis, University of Delaware, 1999.

[360] R. Wettel and M. Lanza. Visualizing software systems as cities. In *Proceedings of the International Workshop on Visualizing Software for Understanding and Analysis (VISSOFT)*, pages 92–99. IEEE Press, 2007.

[361] R. Wettel and M. Lanza. Visually localizing design problems with disharmony maps. In *Proceedings of the ACM Symposium on Software Visualization (SOFTVIS)*, pages 155–164. ACM Press, 2008.

[362] R. Wettel, M. Lanza, and R. Robbes. Software systems as cities: A controlled experiment. In *Proceedings of the International Conference on Software Engineering (ICSE)*, pages 551–560. ACM Press, 2011.

[363] L. Williams and R. Kessler. *Pair Programming Illuminated*. Addison Wesley, 2002.

[364] A. Wilson. Touchlight: an imaging touch screen and display for gesture-based interaction. In *Proceedings of the International Conference on Multimodal Interfaces (ICMI)*, pages 69–76. ACM Press, 2004.

[365] A. Wilson. Playanywhere: a compact interactive tabletop projection-vision system. In *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST)*, pages 83–92. ACM Press, 2005.

[366] A. Wilson, S. Izadi, O. Hilliges, A. Garcia-Mendoza, and D. Kirk. Bringing physics to the surface. In *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST)*, pages 67–76. ACM Press, 2008.

[367] A. Wilson and R. Sarin. Bluetable: connecting wireless mobile devices on interactive surfaces using vision-based handshaking. In *Proceedings of Graphics Interface*, pages 119–125. ACM Press, 2007.

[368] P. Wilson. *Computer supported cooperative work: an introduction*. Springer Science and Business, 1991.

[369] J. Wobbrock, M. Morris, and A. Wilson. User-defined gestures for surface computing. In *Proceedings of the ACM International Conference on Human Factors in Computing Systems (CHI)*, pages 1083–1092. ACM Press, 2009.

[370] C. Wolfe, D. Smith, and N. Graham. A low-cost infrastructure for tabletop games. In *Proceedings of the Conference on Future Play*, pages 145–151. ACM Press, 2008.

[371] M. Wu and R. Balakrishnan. Multi-finger and whole hand gestural interaction techniques for multi-user tabletop displays. In *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST)*, pages 193–202. ACM Press, 2003.

[372] M. Wu, C. Shen, K. Ryall, C. Forlines, and R. Balakrishnan. Gesture registration, relaxation, and reuse for multi-point direct-touch surfaces. In *Proceedings of the IEEE International Workshop on Horizontal Interactive Human Computer Systems (TABLETOP)*, pages 185–192. IEEE Press, 2006.

[373] C. Wysseier. Interactive 3D visualization of feature-traces. Master's thesis, University of Berne, 2005.

[374] B. Yost, Y. Haciahmetoglu, and C. North. Beyond visual acuity: the perceptual scalability of information visualizations for large displays. In *Proceedings of the ACM International Conference on Human Factors in Computing Systems (CHI)*, pages 101–110. ACM Press, 2007.

[375] P. Young and M. Munro. A new view of call graphs for visualising code structures. Technical Report 03/97, University of Durham, 1997.

[376] D. Zeckzer, R. Kalcklösch, L. Schröder, H. Hagen, and T. Klein. Analyzing the reliability of communication between software entities using a 3D visualization of clustered graphs. In *Proceedings of the ACM Symposium on Software Visualization (SOFTVIS)*, pages 37–46. ACM Press, 2008.

[377] H. Zhang, X.-D. Yang, B. Ens, H.-N. Liang, P. Boulanger, and P. Irani. See me, see you: a lightweight method for discriminating user touches on tabletop displays. In *Proceedings of the ACM International Conference on Human Factors in Computing Systems (CHI)*, pages 2327–2336. ACM Press, 2012.

[378] K. Zhang. *Software Visualization: From Theory to Practice*. Kluwer Academic Publishers, 2003.

[379] T. Zuk and S. Carpendale. Theoretical analysis of uncertainty visualizations. In *Proceedings of the Visualization and Data Analysis Conference at Electronic Imaging*, volume 6060, page 606007. SPIE, 2006.

[380] T. Zuk, L. Schlesier, P. Neumann, M. S. Hancock, and S. Carpendale. Heuristics for information visualization evaluation. In *Proceedings of the AVI Workshop on BEyond time and errors: novel evaLuation methods for Information Visualization (BELIV)*, pages 1–6, 2006.