Reflections on Collaborative Software Visualization
in Co-located Environments

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

Abstract—Most software visualization systems and tools for
maintenance are designed from a single-user perspective and are
bound to the desktop and IDEs. These design decisions do not
allow developers to analyze software collaboratively or to easily
interact and navigate visualizations within a co-located environ-
ment. In my PhD thesis I conducted an exploratory study of how
44 pairs of professional software developers use SourceVis, a
collaborative software visualization application, on a large multi-
touch table in a co-located environment. The thesis contributes
to a richer understanding of how pairs of developers collaborate
with shared visualizations on large interactive tabletops to gain
insight into the structure and evolution of software systems.

Keywords—Collaborative Software Visualization; Multi-touch
Tables; Multi-user; Co-located Environments

I. INTRODUCTION

Understanding software for maintenance is often a social
activity and involves pairs of developers working within co-
located environments (same room and same time). In Agile
teams developers typically work in pairs to carry out tasks
including: programming, code reviews, refactoring, and visu-
alization of work flow [1].

Most software development applications and tools that
support these tasks often involve analysis and visualization
features such as code metrics and code coverage tools. These
tools, however, often have limited support for collaboration
when pairs work together in the same room at the same time.

To address these problems my PhD thesis evaluated 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 for software
maintenance purposes [2]. Specifically the thesis made three
contributions.

1) Designing Collaborative Software Visualizations.
I built a large multi-touch table and conducted preliminary user
studies with computer science students using the multi-touch
table and prototypes of my software visualization application.
The user studies led to a protocol which I used for the evaluation
with professional software developers.

2) SourceVis: Collaborative Software Visualization.
I designed and implemented a collaborative software visual-
ization application, called SourceVis, for use on large multi-
touch tables. SourceVis contains analysis and visualization features for exploring the structure and evolution of software.

SourceVis allows multiple users to interact simultaneously or
separately with the table either as a group or as individuals.

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

In this paper I give an overview of my PhD highlighting
the contributions, and discuss the lessons learned focusing on
what worked well and what could have worked better.

II. LARGE MULTI-TOUCH TABLES

In order to explore my topic a large multi-touch table was
required. Due to the lack of available commercial touch tables
at the time I decided to build my own following some existing
guidelines [3]. I built the Blue Multi-Touch Table based upon
the rear-diffused illumination technique (see Figure 1). I now
describe some of the parts of the table.

The steel frame for the table was created and had dimen-
sions of 1200mm wide, 920 mm height, and 780mm deep.
The front side was fixed with two sliding doors on the sides.
The back had an adjustable black cloth material that provided
enough ventilation for the table. The table had four wheels
and a wooden base. The surface diffuser material was rear
projection acrylic called Plexiglas RP 7D006 and is 5mm thick.

The IR lights were four IR LED bars each with 12 LEDs at
850nm. The bars were placed evenly inside the middle of the
table to spread the IR light evenly. The camera was a modified
PS3 with a CS lens mount, 850nm band pass filter, and housed
inside a case. The lens was a Theta 1.67mm, adjustable, and
had a wide angle 113 degree field of view. The data projector
was a short throw Sanyo PLC-WXL46 (1280x800 resolution),
pointed directly at the surface, and mounted to a sliding draw.
The display surface was 1077mm x 673mm (approximately 48
inches display screen). The computer was 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.
III. SourceVis: Software Visualization

SourceVis [4] is a collaborative software visualization application for use on a large multi-touch table. Figure 2 shows some sample visualizations from SourceVis. The aim of SourceVis is to help developers working in co-located teams to explore how a system has been structured by visualizing metrics, stricture, and evolution data. SourceVis contains a suite of 13 software visualization techniques that can help identify what parts of a system are large and likely need to be refactored. The visualizations are grouped into three categories: exploration, structure, and evolution.

SourceVis is designed for multiple users to interact synchronously or asynchronously on a large horizontal interactive multi-touch table. The visualizations can be displayed at full screen, within scalable windows, or at any orientation on the horizontal plane. Multiple visualizations can also be displayed at once in separate windows, or overlaid upon each other.

Figure 2(a) shows a visualization of the dependencies of a class. The class is displayed in the middle of the visualization with dependent classes on the outside. Selecting a class can show metrics about the class. Figure 2(b) shows multiple developers interacting simultaneously with different visualizations. One visualization (left) shows metrics about a system while the other (right) shows the class dependencies of a system.

IV. Evaluation

To evaluate the design of collaborative software visualization I followed a qualitative grounded approach which is common within information visualization [12]. I conducted a user study to evaluate how pairs of software developers performed with SourceVis and a uniform set of tasks to gain insight into existing software systems. The aim of the study was to explore how pairs make use of shared visualizations, perform joint group work, adopt collaborative coupling strategies, and physically arrange themselves around a large multi-touch table. The design of the study used a within-subjects test. Some of the research questions I wanted to answer were as follows:

RQ1 Which study condition combination do pairs prefer (joint group work or parallel individual work)?
RQ2 Which coupling styles do participants use?
RQ3 Which arrangement styles do participants use?

A. Study Condition Combinations (RQ1)

As part of this user study I wanted to explore pairs working on the multi-touch table completing the user tasks in two condition types: as a group or as individuals. I wanted to find out which condition was most effective and which condition participants preferred. I used a within-subjects design and all participants got the opportunity to explore both conditions.

- 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.

B. Collaborative Coupling Style (RQ2)

I wanted to observe what collaborative coupling styles pairs utilized based on the model of Tang et. al [5] (see Table I). The coupling styles are grouped 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). I subdivided the different problem (DP) code by adding in two new styles. Participants working on different problems and the same area of the table (DPDA) and working on different problems and different areas of the table (DPDA) (highlighted grey in Table I).
Table I. Collaborative Coupling Styles, with modifications highlighted in grey [5].

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

C. Physical Arrangement Style (RQ3)

I was interested in observing where participants stood in relation to the multi-touch table. Only three sides of the table were available for use as one side of the table had equipment set up for video recording purposes. Figure 3 shows five of the physical arrangement styles similar to Tang et al. [5] including: (A) Together, (B) Kitty Corner, (C) Side by Side, (D) End Side, and (E) Opposite Ends. Tang et al. [5] had two other positions as their table was accessible from all sides. I also came up with another position, (F) Apart, for when one or both participants were standing away from the table.

D. Participants

44 developers volunteered to participate in the study who all came from local companies (18 in total). I was strict on the criteria for who was recruited for the user study: each pair had worked for the same organization, within the same team, and had known each other for at least 12 months. The participants were all familiar with object-oriented programming languages like Java and C#.

Fig. 4 User study room setup.

E. User Study Room Setup

Figure 4 shows the setup of the room. Figure 4(a) shows the multi-touch table where three sides 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. I video recorded each study from two angles. Figure 4(b) shows the two cameras (A) and (B), and a microphone. Camera (A) was mounted above the table and pointed downwards to record the hands of the participants interacting on the screen. The microphone hung from this camera mount. Camera (B) recorded how the participants worked around the multi-touch table.

F. Procedure

The procedure of the study involved three sections: pre-study, user tasks, and post-study. To conduct the user study I obtained approval from my university human ethics committee. Up to 120 minutes were allowed to complete the user study.

1) Pre-Study: Participants were first given an information sheet, a consent form to sign, and a pre-survey questionnaire to complete. The multi-touch table was introduced and participants were given practice with some example multi-touch applications from MT4j [6]. Participants were then given a demonstration of SourceVis to understand the basics of the application. To become familiar with SourceVis participants were given time to explore visualizing some example software systems. The example systems were different to the ones used in the user tasks and ranged in size from very large to small including: Weka, ArgoUML, GanttProject, and SquirrelSQL. Up to 40 minutes were allowed to complete the pre-study.

2) 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. Up to 60 minutes were allowed for pairs to complete the user tasks. 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?” [7].

Participants were given a scenario where they are working on a co-located software development team to maintain Junit and JHotDraw. They were asked to improve the systems so that they are more easily maintainable by identifying code in the visualizations which could be refactored (e.g. identifying large classes, classes with many dependencies). Below shows some example questions starting with an overview and drilling...
down to explore details on demand. The questions required the participants to search for the size of packages and classes, number of dependencies, and the methods of a class.

- What is the largest package in JHotDraw 7.5.1?
- How many classes are there in the package org.jhotdraw.geom?
- What are the two largest classes in this package?
- Which classes does the largest class depend upon the most?
- In the largest class how many accessor methods are called by the public method repaintHandles()?

Pairs answered questions in three sections where each section was five (Group condition) or ten questions (Individual condition, five per participant) depending on the condition (see Table II). I alternated the condition each pair began in. For Section 1 pairs started either in the Group or Individual conditions. For Section 2 a pair that started in the Group condition then answered the Individual questions and vice versa. For Section 3 pairs got a choice as to what condition they wanted to do, either Group or Individual.

3) Post Study and Data Collection: Each participant completed a post-study questionnaire to rate their perceptions on the effectiveness of SourceVis. A post-interview was conducted to explore anything interesting observed during the study. Up to 20 minutes were allowed to complete this post-study. I collected video for all pairs, then analyzed each video and coded the actions (collaborative coupling and physical arrangement styles) I observed the participants performing using a video analysis tool called ELAN1. Each video was approximately 2GB and took up to eight hours to code as it required very precise analysis for each action with up to 250 codes.

V. FINDINGS

I discuss the key findings from the user study, and the implications for collaborative group work using large interactive surfaces and the design of multi-touch tables. More details can be found in the PhD thesis [2].

A. Collaborative Group Work

I wanted to find out if participants preferred group work or parallel individual work, what coupling styles they used, and what physical arrangement styles they used.

1) Design for joint group work over parallel individual work: I found that pairs preferred to work in groups rather than individuals to complete the user tasks. Groups worked jointly together and individuals worked separately in parallel. Pairs experienced working in groups and as individuals. Participants preferred to work as a group as evidenced by all 22 pairs selecting to work together as a group in Section 3.

I 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. I 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 stated in the post-interviews that working as a group was easier and more effective. I observed pairs completing questions faster in groups than as individuals.

Some participants appreciated that the multi-touch table allowed them to work as individuals at the same time, but they preferred group work. I observed that most participants struggled performing parallel individual work. Some pairs even completed the Individual condition as a group.

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. [8].

2) Support a flexible variety of coupling styles: I found that pairs spent more time in 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.

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.

Collaborative software visualization systems should support a flexible variety of coupling styles. This reinforces the findings from Tang et al. [5].

3) Support fluid transitions between coupling and arrangement styles: I found the most frequently used arrangement styles by pairs were standing Side by Side, standing at Opposite Ends, and standing Apart. 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 Side by Side and Opposite Ends which were the dominant styles in the Group and Individual conditions respectively.

I compared coupling styles to arrangement styles. The 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

---

1http://tla.mpi.nl/tools/tla-tools/elan/

<table>
<thead>
<tr>
<th>Section</th>
<th>GIG</th>
<th>GI</th>
<th>IG</th>
<th>IIG</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Group</td>
<td>Group</td>
<td>Individual</td>
<td>Individual</td>
</tr>
<tr>
<td>2</td>
<td>Individual</td>
<td>Individual</td>
<td>Group</td>
<td>Group</td>
</tr>
<tr>
<td>3 (choice)</td>
<td>Group</td>
<td>Individual</td>
<td>Group</td>
<td>Individual</td>
</tr>
</tbody>
</table>
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.

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. [5] and Isenberg et al. [8].

4) Design visualizations for closely coupled arrangements with rotation features: I found the preferred coupling style was viewing engaged (VE) and arranged 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 (Bottom (A) in Figure 4(b)). When pairs were working on parallel individual work, participants displayed visualizations in windows and manually rotated them to face their direction. This meant participants could stand in different arrangements.

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 options for rotating the visualizations. There should also be automatic options to perform the rotation operations. This reinforces the findings from Kruger et al. [9].

B. Designing Multi-touch Tables

I wanted to find out how the multi-touch table helped pairs with the user tasks for software maintenance.

1) Provide high resolution display: I found that the resolution of the table used in the study (1280x800 pixels) 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. Participants stated that they found text was readable when visualizations were displayed at full screen, but reading text when displayed in visualization windows was difficult especially when windows were reduced in size.

Multi-touch tables should provide a high resolution display. The resolution should be equal or greater than contemporary desktop computers. Providing a high resolution display would reduce windows overlapping and make text easier to read. This reinforces the findings from Tang et al. [5].

2) Provide appropriate table space: I 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. I found that when pairs were performing parallel individual work the size and resolution of the table was not large enough. This forced pairs to mainly be arranged at opposite ends of the table.

Multi-touch tables should provide appropriate table space. Depending on the task and number of users 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 [10]. This reinforces the findings from Scott et al. [11].

3) Distinguish between simultaneous user interactions: I found that most of the time one participant was interacting with the table even though the multi-touch features allowed multiple users to interact at once. When pairs were interacting simultaneously the system could not distinguish 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. Some pairs stated this was the reason why they took turns when interacting with the table. These issues were partly due to the multi-touch table sometimes suffering from inconsistent touch detection.

Multi-touch tables should distinguish between simultaneous user interactions to detect different users interacting with the system. Distinguishing between user interactions will likely offer a more effective collaborative user experience.

VI. Discussion

I now discuss the lessons learned from the PhD process focusing on what worked well and what could have worked better. In the anticipation the advice will help future students.

After my PhD candidacy I had three major steps: build multi-touch hardware, develop a software prototype, and conduct user evaluations. The steps needed to be completed in order as they impacted each other. I found that each of these steps took a long time to perform and it was not clear when each step was complete. Getting feedback early from my supervisors and peers on each step of the PhD process was important for me as it helped me move from one step to the next. I did not have detailed plans for each step but had a general goal in mind. I found writing the main ideas on a whiteboard and ticking them off as I went was good motivation for showing what progress had been made. Perhaps planning each step into smaller more detailed monthly segments may have been more effective to help me keep on track more.

In order to conduct the research I needed access to a large multi-touch table. At the time of the project multi-touch tables were very expensive for the budget I had and weren’t available for purchase in my country. Hence this led me to build my own hardware. Unfortunately building the tables was a very time consuming process as it required a lot of labour, effort, and very precise construction and collaboration. These steps impacted the time left I had to complete the PhD. In retrospect I could have probably used a smaller touch screen with less touches but it was a tradeoff in terms of what I could build versus what was available, and want the goals of the project were. As collaborative devices like large multi-touch tables become ubiquitous there will be less need to build your own.
Developing PhD prototype software is very different than building a commercial product. Identifying the key features of the design and implementing these was more valuable than trying to implement everything from the requirements and making the system perfect. Knowing when your prototype software is complete enough and when to implement the next feature is challenging. Adopting an Agile like process for developing the software by developing features one at a time and obtaining regular feedback proved to be more effective rather than developing a large overly complex application and then obtaining feedback at the end. Sometimes quick hacks were required as opposed to elegantly designed features. I found that when developing my prototypes I was more effective using programming languages I was familiar with (e.g. Java, Perl) rather than some of the other languages I experimented with at the beginning of the project which supported multi-touch features (e.g. Processing, C#, C++, Smalltalk).

As part of the process of developing the software visualization prototypes I continually conducted various user studies with students to establish a protocol following a qualitative grounded approach [12]. Evaluating with a convenience sample of computer science students was not as realistic as they were not the intended target users. Once the user study protocol was mature enough I was confident at evaluating my prototype with professional software developers. Evaluating with professional software developers was time consuming to organize but critical to the research as they were the target users of my system and offered subjective opinions based on industry experience. As part of the evaluation process I found it was more efficient to obtain human ethics approval for the entire PhD thesis as opposed to each user study I conducted.

Conducting research outside of North America and Europe is challenging. Academic supervisors outside of North America and Europe tend to have a broader scope of research projects. This leads to selecting supervisors at your university that may not be inherently familiar with your project. Selecting supervisors with a track record of supervising PhD students and access to research funding is vital. Without funding it is difficult to travel and present your research since the majority of the top tier conferences are predominantly hosted in North America and Europe. When attending conferences it is important to network with other students in your areas to share ideas and build relationships. Submitting papers to regional conferences outside of North America and Europe is the least effective for getting exposure of your research. Hence the goal should be to aim for high quality publication venues (e.g. ICSE, FSE, AVE, ICSME) as opposed to lower quality regional ones.

In the future, I would like to conduct a comparative user experiment between collaborative software visualization tools and existing developer tools to see if pairs of developers outperform with the collaborative tools. I would envisage the experiment to be similar to Wettel et al. [13], but between SourceVis and state of the art software exploration tools (e.g. Eclipse and spreadsheet of metrics data). An issue with user studies and experiments is most of the time they are conducted in controlled environments. Based on the results of this comparative user experiment I would like to conduct field studies of developers using collaborative visualizations in their daily work routines to see how effective collaborative tools could be for developing and maintaining software.

VII. CONCLUSION

Most software visualization systems and tools for maintenance are designed from a single-user perspective and are bound to the desktop, IDEs, and the web. These design decisions do not allow developers to analyze software collaboratively or to easily interact and navigate visualizations within a co-located environment. This paper gave an overview of my PhD thesis where I described the three contributions: building multi-touch tables, developing SourceVis, and conducting the first rigorous user study with professional software developers for collaborative software visualization.

The key findings of the research were as follows. For collaborative group work I recommend designing for joint group work over parallel individual work, supporting a flexible variety of coupling styles, supporting fluid transitions between coupling and arrangement styles, and designing visualizations for closely coupled arrangements with rotation features. For the design of multi-touch tables I recommend providing a high resolution display, providing appropriate table space, and distinguishing between simultaneous user interactions. Our findings reinforced some findings from other researchers.

Acknowledgments: This work was supported by the New Zealand Ministry of Science and Innovation, TelstraClear, Victoria University PhD Scholarship, and the Canadian NSERC SurfNet network.

REFERENCES