Location, Location, Location: Using Spatial Memory in an Integrated Development Environments to Assist Program Code Comprehension? (Work in Progress)

Craig Sutherland, Andrew Luton-Reilly, Beryl Plimmer
Department of Computer Science
University of Auckland
{cj.sutherland|a.luxton-reilly|b.plimmer}@auckland.ac.nz

Abstract
An important task while reading program code is finding the location of relevant sections. When reading non-code documents, readers often rely on spatial memory and indications of the document hierarchy inherent in headings to build up an understanding. However, documents involving code are structured in a non-linear way, without the benefit of headings. Using the results from an observational study, we describe a tool that uses spatial memory for finding previously read sections of code. We propose that this tool will reduce the amount of time spent navigating through code and thus assist comprehension.

1. Introduction
Previous studies indicate that programmers mainly read program code on electronic devices (Sutherland, Luxton-Reilly, & Plimmer, 2015). Electronic devices do not provide the same affordances that paper provides for reading. One such affordance is the ability to find previously read material based on what the reader remembers of the context. The approximate location of the material is an important part of this context.

Reading program code for understanding is a more difficult task than reading other types of documents. Programmers do not read a program from start to finish: instead, they first identify a starting point they think is important and then follow the flow of the program logic (Roehm, Tiarks, Koschke, & Maalej, 2012). This requires the programmer to jump around the code files and functions as they read. This non-linear reading style increases the cognitive demands on the reader (Crisp & Johnson, 2007).

In a previous study, we observed programmers reading program code on paper. From this study, we argued that programmers use annotations to support their wayfinding (Sutherland et al., 2015). Annotations serve as way-marks, allowing a reader to quickly find material they have previously read (Marshall, 1997). Using external cognition as a model to explain this, annotations serve to spatially constrain the search space, and allow the reader to offload some of the cognitive effort required (Rogers, 2004). The navigational benefits of annotations have been observed for both text-based annotations (Storey et al.,2009), and freeform annotations (Sutherland et al., 2015).

However, annotations were not the only form of navigation support that we observed. During the same study (Sutherland et al., 2015), we observed some programmers placing pages in specific locations on the surface (see Figure 1). These programmers would pick up paper from the location, read, and then return the paper to the same location. When asked, the programmers were able to tell us what was at each location. This suggests that the spatial context plays a role in code navigation.

In this paper, we describe our current work on investigating spatial navigation for program code. First, we review related work on spatial memory during reading, and how this influences comprehension. Then we include some unreported results from the previous study. Finally, we describe our current Integrated Development Environment implementation.

2. Related Work
Early studies compared paper to online reading and found a number of differences (O'Hara & Sellen, 1997). For example, paper provides many affordances not available on a computer screen (O'Hara & Sellen, 1997). One major advantage of reading on paper relates to the reader's ability to remember things spatially. Using paper allows the reader to lay out pages in space which helps provide an

overall sense of the document structure. This allows for visualising large amounts of data, quick referencing between documents and concurrent reading of different pages. Readers are able to take in the whole page at a glance, including both what is on the page and where things are relative to each other. Having fixed layout provides explicit cues about the document. These cues supported search and re-reading activities (O'Hara & Sellen, 1997). A second major advantage is paper allows rapid, effortless navigation through the document. Readers can quickly navigate between sections using one or two hands as needed, allowing navigation to overlap with other activities (O'Hara & Sellen, 1997).

While technology has improved, screen-based reading does not yet provide the same spatial memory or rapid navigation benefits. There have been attempts to use multiple devices (e.g. tablet PCs (Morris, Brush, & Meyers, 2007)) to provide these benefits. These devices allowed some rapid navigation but are still limited. In addition, the bulkiness of the devices constrained their use.

Central to these investigations was the premise that changing the form of the text does not influence comprehension. However, studies on how people understand literature challenges this premise. Recent research suggests that moving from a fixed layout (e.g. a page) to a flowing layout (e.g. scrolling text) may fundamentally change how people interact with documents (Mangen & Kuiken, 2014). The change in format may be causing dislocation for the reader as the way-marks in the document are no longer fixed (Mangen & Kuiken, 2014). There are no studies on whether programmers are similarly dislocated when reading code.

Reading on a screen has been found to result in lower comprehension when reading for understanding (Mangen & Kuiken, 2014). High comprehension readers build a mental model of what they are reading (Cataldo & Oakhill, 2000). This model is linked to where they read things and uses the reader's spatial memory. If the layout of the document changes then the reader is unable to rely on spatial memory to locate previously read materials (Cataldo & Oakhill, 2000). A second related reason is when people read, they encode not just the text but also the surrounding context (Long & Spooner, 2010). With paper, the reader encodes the text, its location, and the items around it. With a scrolling reader there is less contextual information to encode.

3. Study Design

In our study, we observed 13 experienced programmers as they read a short C# program (six classes) printed on paper. The participants were instructed to read the program so they could explain it to another programmer. The participants were allowed to manipulate the paper in any way they desired. The interested reader is referred to Sutherland, et al. (2015) for the full details of the study design.

4. Findings on Spatial Navigation

Nine of thirteen participants arranged stacks of paper around the desk in specific locations. By specific location, the participants could tell us what was in each location just by glancing at the top page in the pile. While reading, these participants would pick up a stack of paper and either move it to the bottom centre location or hold it in the air while they were reading. When the participant finished reading they would return the stack to its previous location.

Figure 1 shows an example of a participant's layout. The pages around the left, top and right are the individual code files. In this example, the participant has highlighted the name of each code file. The bottom centre shows the participant current working files. The file they are currently reading is centre right; the page centre left is for working notes. Notice the space at the top: this was the location of the current file they are reading.

As this behaviour was not the focus of the study we did not query the participants about it. We only have comments from seven of the participants on their use of the surface for laying out the pages. These comments show a consistent rationale behind why they did this.

Four of the participants started distributing the pages based on a conscious choice: they deliberately arranged the stacks of paper to allow quick access. One participant stated, "I knew I needed to flick between files so I wanted to do so quickly" and "by arranging like this I know the location of each file". Another participant stated, "I placed the files this way because I know they are related and I would need to move between them". Thus the arrangement of the stacks allows quick navigation.

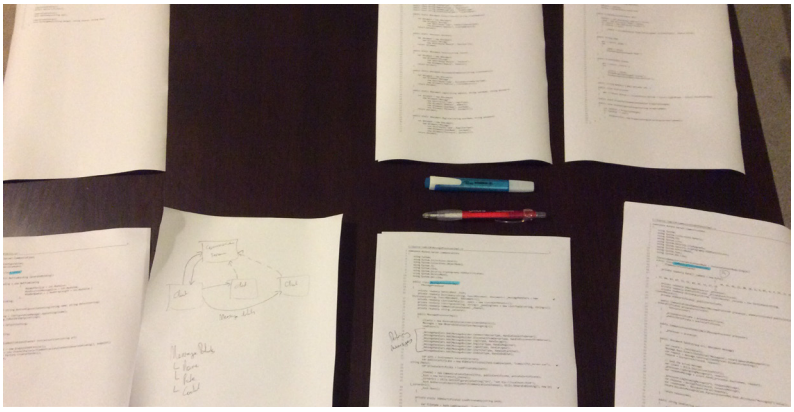

Figure 1 – Example layout of all the files by a single participant in the study.

The remaining three participants did not originally layout the pages. But by the end of the study they had changed their approach. One participant stated, "Originally I kept losing my place, I knew what I needed but I had to keep searching for it. Then I started placing the pages back in same place and I could find them much faster." Another participant stated "I didn't care where the files were but then I figured which ones needed to be where." Both comments show the participants found they could reduce time by having the stacks in specific locations.

We also considered the approaches taken to locate files during the task. While there were a variety of responses, we group them into three broad categories. The first group of participants were those who "knew" where each file was. These participants had chosen or memorised the location of each file and made sure they returned it to the same location when finished (two participants). The second group added annotations to the top of the first page to identify the class (four participants), either by highlighting the class name or writing the class name at the top of the page. They would then quickly glance at these annotations to identify the pile. The final group located pages by what the front page looked like (two participants). By the end of the reading task, they had a rough mental image of each front page and used these images. One of these participants stated, "I just know what it looks like now." We did not ask the final participant how they identified each pile[1].

One final related observation is the space needed for reading. Five participants left a space for reading (three deliberately, one by accident and one unknown as to why). One participant stated the reason for this space was "so I can put things and it won't mess up the other piles." Another participant stated "I like to rest the paper but I didn't want to accidentally pick up other pieces of paper." The other four participants did not leave a space (three by accident and one unknown). One participant stated "why would I need a space for reading, I'm holding the paper as I read." The other two participants said they did not make a choice to do this but just moved paper as needed.

5. Current Prototype

Spatial layout is not currently available in Integrated Development Environments (IDEs), the workbench for most programmers. However there has been some research on using spatial layout for debugging (DeLine, Bragdon, Rowan, Jacobsen, & Reiss, 2012). Also, IDEs provide a significant amount of support for code navigation with search functionality and hierarchical code displays

In order to investigate whether this functionality would be useful in an electronic environment, we have designed and are currently developing a prototype. We have deliberately embedded spatial layout functionality within an IDE so that the IDE support continues to be available. Figure 2 shows the current prototype we are developing. The user can open multiple code editors on the display surface, close them and move them around. The current prototype embeds these editors as scrolling windows but the intention is to change them into a paging display, with an indicator as to the number of pages in the code file.

[1] This was the first participant and we were not aware of this behaviour yet

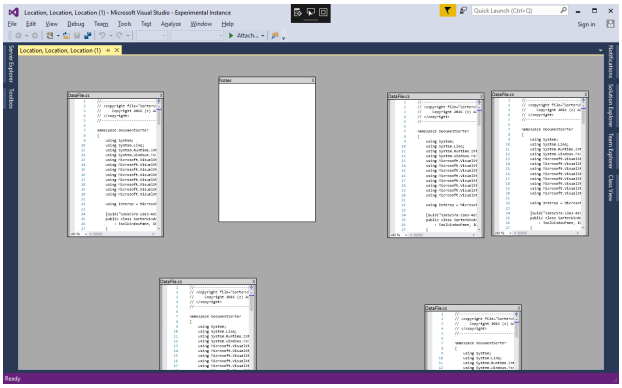

Figure 2 – Current prototype of the navigation layout.

To evaluate the prototype we will perform a between subject evaluation. Each participant will be asked to read and explain a small program in either the new prototype or the standard version of Visual Studio. Planned comparison variables are the length of time needed and accuracy of comprehension. We will also observe the participants' behaviours as they read.

6. References

Cataldo, M. G., & Oakhill, J. (2000). Why are poor comprehenders inefficient searchers? An investigation into the effects of text representation and spatial memory on the ability to locate information in text. *Journal of Educational Psychology*, 92(4), 791.

Crisp, V., & Johnson, M. (2007). The use of annotations in examination marking: opening a window into markers' minds. *British Educational Research Journal*, 33(6), 943–961.

DeLine, R., Bragdon, A., Rowan, K., Jacobsen, J., & Reiss, S. P. (2012). Debugger canvas: industrial experience with the code bubbles paradigm. In *Proceedings of the 34th International Conference on Software Engineering* (pp. 1064-1073). IEEE Press.

Long, D. L., & Spooner, A. (2010). Placing a text in context. *Psychonomic Bulletin & Review*, 17(2), 237–242. doi: 10.3758/PBR.17.2.237

Mangen, A., & Kuiken, D. (2014). Lost in an iPad: Narrative engagement on paper and tablet. *Scientific Study of Literature*, 4(2), 150–177.

Marshall, C. C. (1997). Annotation: From paper books to the digital library. In *Proceedings of the 2nd ACM international conference on digital libraries* (pp. 131–140). New York, NY, USA: ACM. doi:10.1145/263690.263806

Morris, M. R., Brush, A. J. B., & Meyers, B. R. (2007). Reading revisited: Evaluating the usability of digital display surfaces for active reading tasks. In *2nd annual IEEE international workshop on horizontal interactive human-computer systems* (pp. 79–86).

O'Hara, K., & Sellen, A. (1997). A comparison of reading paper and on-line documents. In *Proceedings of the SIGCHI conference on human factors in computer systems* (pp. 335–342). New York, NY, USA: ACM. doi: 10.1145/258549.258787

Roehm, T., Tiarks, R., Koschke, R., & Maalej, W. (2012). How do professional developers comprehend software? In *Proceedings of the 34th international conference on software engineering* (pp. 255–265). Piscataway, NJ, USA: IEEE Press.

Rogers, Y. (2004). New theoretical approaches for human-computer interaction. *Annual Review of Information Science and Technology*, 38(1), 87–143. doi: 10.1002/aris.1440380103

Storey, M., Ryall, J., Singer, J., Myers, D., Cheng, L., & Muller, M. (2009, July). How software developers use tagging to support reminding and refinding. *IEEE Transactions on Software Engineering*,35(4), 470-483. doi: 10.1109/TSE.2009.15

Sutherland, C., Luxton-Reilly, A., & Plimmer, B. (2015). An observational study of how experienced programmers annotate program code. In J. Abascal, S. Barbosa, M. Fetter, T. Gross, P. Palanque,& M. Winckler (Eds.), *Human-computer interaction – INTERACT* (Vol. 9297, pp. 177–194). Springer International Publishing. doi: 10.1007/978-3-319-22668-2_15