Effectiveness of Spatial Overviews for Navigating Software Projects

Woody Morrice
woodymorrice@gmail.com
University of Saskatchewan

Carl Gutwin
gutwin@cs.usask.ca
University of Saskatchewan

Ian Stavness
ian.stavness@usask.ca
University of Saskatchewan

Andy Cockburn
andy@cosc.canterbury.ac.nz
University of Canterbury

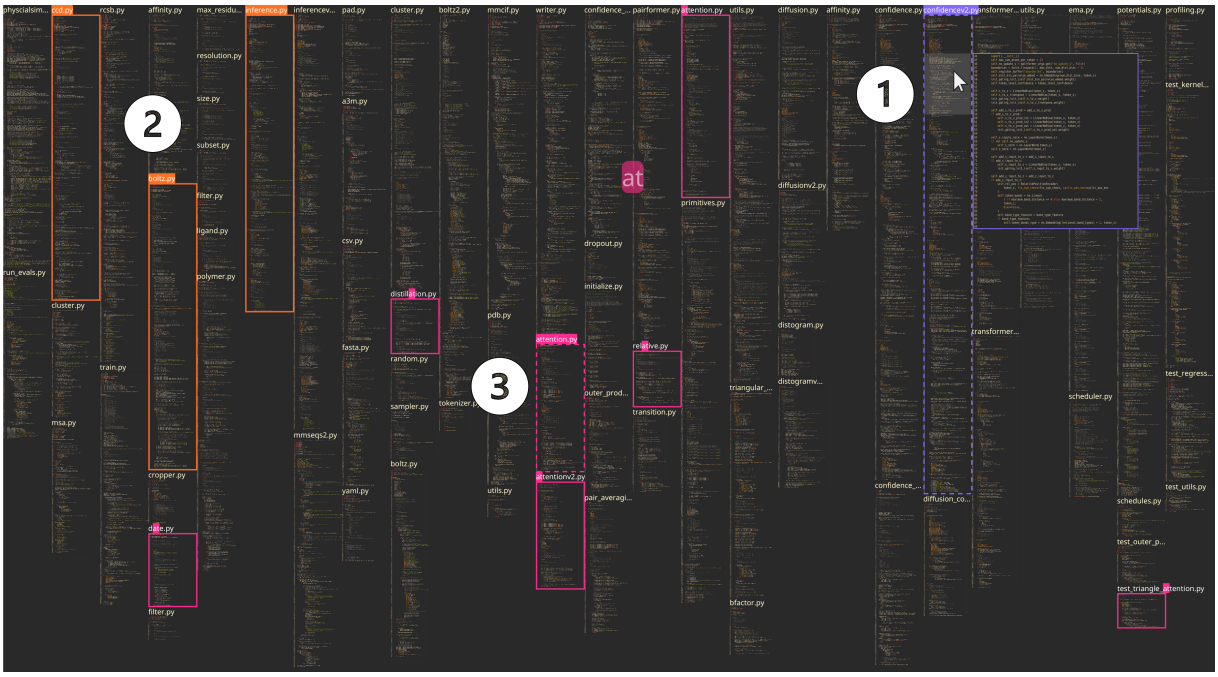

Figure 1: Thumbview interface for a 75-file Python project (github.com/jwohlwend/boltz). File miniatures are shown in columns. 1: Holding shift shows a preview of the code under the cursor. 2: Recently-opened files are highlighted with an orange outline. 3: Search results are highlighted in pink; the file with keyboard focus is indicated in purple with a dashed outline.

ABSTRACT

Spatial overviews allow fast navigation in static documents by allowing users to develop spatial memory of page locations. However, little is known about whether spatial overviews support navigation tasks that involve inter-document navigation, and whether memory-based navigation is compromised when document sets change. We evaluated these two issues in the context of a real-world software project. Our first study looked at inter-document navigation, and compared a spatial overview to standard IDE tools. Across several finding and re-finding tasks, participants were able to revisit files faster with the spatial overview — an advantage that increased as experience grew. A second study explored the effects of project evolution on performance with the spatial overview, and found that spatial memory was resilient to local changes, with participants quickly returning to expert-level performance after a change. Our work reinforces the value of spatial overviews for realistic tasks and usage scenarios in software development environments.

GI '26, June 09-12, 2026, Kitchener-Waterloo, Ontario, Canada

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM
https://doi.org/10.1145/nnnnnnn.nnnnnnn

CCS CONCEPTS

• **Human-centered computing** → **Interaction techniques**.

KEYWORDS

Spatial overviews, spatial navigation, space-filling thumbnails

ACM Reference Format:
Woody Morrice, Carl Gutwin, Ian Stavness, and Andy Cockburn. 2026. Effectiveness of Spatial Overviews for Navigating Software Projects. In *Graphics Interface (GI '26), June 09-12, 2026, Kitchener-Waterloo, Ontario, Canada.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Spatial overviews show all items in a stable layout that allows users to develop spatial memory of item locations [50]. Once locations are learned, users can navigate to any item with a single click – and previous studies have shown that spatial navigation is faster than traditional techniques such as scrolling [13], filtering [30], or hierarchies [51], particularly when tasks involve frequent revisitation. One well-studied implementation of spatial overviews is Space-Filling Thumbnails (SFT) [13, 29], which show a document as a grid of page thumbnails. Users navigate to different pages by clicking on them in the overview, rather than by scrolling. Previous work has shown SFTs to be faster than scrolling, with the advantage increasing with document size and the number of visits [13].

SFTs have primarily been considered for within-document (i.e. *intra-document*) navigation, where the main advantage of the SFT technique is in reducing time-consuming scrolling actions. However, many task scenarios also involve navigation *across* documents (i.e., *inter-document* navigation) where scrolling is not the main method of navigation. Spatial overviews may offer an advantage in tasks that require navigation between and within files since both navigation actions can be performed with the same interface and set of navigation actions. Although other studies have tested the spatial-overview approach in multi-document contexts (e.g., the Data Mountain experiments with web bookmarks [48]), previous research has not looked at spatial navigation in tasks that involve substantial inter-document navigation.

One common scenario that provides this context is software development in an integrated development environment (IDE). Software projects have many files that programmers open and close frequently over the course of a work session, with many revisitations to the same files; therefore, software development tasks could potentially benefit from the improved navigation efficiency that is promised by the SFT approach. Modern IDEs, however, already provide many different navigation techniques (including editor tabs, project browsers, and search tools), and these may already support the user's needs for finding and re-finding files in a project. In addition, the advantages of SFTs depend on the stability of the spatial layout – and projects change over time, which could compromise the user's spatial memory and reduce the technique's effectiveness.

To determine whether spatial overviews can improve navigation in realistic software development datasets and tasks, we developed and evaluated a tool called the Thumbview that is based on Space-Filling Thumbnails, for a simulated IDE based on Microsoft's Visual Studio Code (VSCode). We carried out two studies to investigate two questions: first, how spatial overviews compare to standard IDE navigation techniques (including project browsers, search tools, and editor tabs) for finding and re-finding files in a software project; and second, how performance is affected when the layout of the overview changes (due to changes in the software project). The dataset was a 75-file open-source project from GitHub.

The first study compared three common navigation tasks with the Thumbview against standard IDE tools. Thumbview navigation was found to be faster in all three tasks, and the advantage increased as participants gained experience with the interface and were able to take advantage of their increasing spatial memory. The second study investigated how local file movements in the Thumbview would affect participants' revisitation ability. We observed a performance drop in only two of five movement types, and participants were able to return to their original level of performance with the interface after one or two revisitations.

Our studies make three contributions. First, we add to evidence showing the performance advantages of spatial overviews and spatial navigation. Second, we demonstrate the Thumbview tool and show that spatial overviews work well for inter-file navigation in a realistic integrated development environment. Third, we show that spatial overviews do not have to be completely stable to be useful – user performance with a spatial overview can be resilient to local changes in the spatial layout. Overall, our work shows that the spatial-navigation approach and the Thumbview technique can help to improve navigation efficiency in software projects.

2 RELATED WORK

2.1 Revisitation and Spatial Overviews in UIs

Many tasks commonly performed with interactive devices are highly repetitive. Research has shown high rates of repetition in command use [27, 28], menu selection [14, 24], and web browsing behavior [6, 15, 41]. Studies have found that 58-80% of web page navigations are revisits [41, 60], that people tend to revisit pages that they recently navigated to, and that people browse in very small clusters of related pages [60]. Early research into the organization of a file interface found that storing documents in a stable spatial layout improved document retrieval [34]; similarly, the Data Mountain experiments [48] further explored the advantages of human spatial memory in document retrieval, and showed that a 2-D spatial layout was an effective alternative to the "favorites" mechanisms of a web browser. Flat spatial organizations of commands have been shown to perform better than common interface controls such as lists and space-multiplexed hierarchies [44, 54]. For example, studies of Scarr et al.'s CommandMaps system [51] showed that flattening a spatial hierarchy allowed experienced users to perform menu-based tasks significantly faster than a traditional space-multiplexed ribbon menu, and that CommandMaps were positively received by users in realistic use cases [52].

Spatial overviews provide a birds-eye view of some set of data in two dimensions and can function similarly to a map, allowing users to develop survey knowledge of the spatial layout and use their long-term spatial memory to recognize and differentiate spatial locations [40]. Spatial overviews are useful as revisitation tools, but are also commonly used for other tasks such as information visualization and layout planning. *Spatial stability* is an important aspect of spatial layouts used for revisitation – in a stable layout, elements remain in consistent positions relative to each other, allowing users to rapidly develop expertise with the overview by memorizing element locations. This expertise development can in turn reduce or eliminate the time spent performing visual search.

There is evidence that spatially-stable overviews can improve revisitation in documents and software projects by leveraging the user's spatial memory to learn a mental map of the environment [13, 19], with the added benefit that spatial navigation reduces the need for scrolling. Space-Filling Thumbnails (SFT) [13] is one such technique that has been shown to be an efficient mechanism for document navigation; field studies with the technique showed that users were not hesitant to use the new method, despite being more familiar with standard navigation tools [29]. However, the capabilities of SFTs for inter-document navigation have not been studied, and it is also unclear how well spatial navigation approaches will work when the underlying data changes.

2.2 Code Navigation

Software maintenance studies have shown developers open more files than needed to complete a fix to explore and understand the context of their task [58] or to consider the impact of a change [8]. Developers tend to work with a small number of files per task (i.e., a *working set*), and spend a significant amount of time navigating between working sets [8, 26, 35, 38]. Repeated navigation between working sets has been identified as a limiting factor to developer productivity. Murphy et al. described the repeated task switches as a waste of time [35], and an eye tracking study of bug-fixing behaviour referred to the problem as "thrashing" [56]. A 41-developer study using the Eclipse IDE found that 75% of navigation actions involved the file explorer, and navigation tools like search, the file outline, and bookmarks were used significantly less [42].

Research into visualizations to support code navigation has produced several revisitation tools, some focusing on visualizing the working set [11, 20, 32, 36], and others on visual properties of code [19, 22, 25] which has been referred to as the "code-map metaphor" [9]. Studies investigating code comprehension suggest that navigation tools that effectively leverage visuo-spatial working memory (i.e., the ability to process and retain an object's identity and spatial location) have the potential to increase developer efficiency [10].

Previous studies have presented evidence that there are limitations to standard navigation tools for the document-based workflows of an IDE [19, 20, 32]. For example, file-explorer navigation can require several navigation actions, and relies heavily on scrolling, which limits the visual search space [16]. Significant issues have also been identified with modern tab-based document viewers [12, 31, 33, 37, 46, 57]; the term "tab overload" has been used to refer to the inability of tab-based interfaces to support the increasing complexity of user tasks, which has inspired numerous tab management tools [3, 4, 12]. Search tools also present limitations – they require memorization of item names, which has been shown to be more difficult than remembering images [43, 45] or locations [55], and they typically do not support improvement over time, since the process of finding an item with search does not change as the user gains experience.

3 DESIGN OF THE THUMBVIEW DISPLAY

In order to conduct empirical studies examining the effectiveness of spatial overviews in software development, we designed a navigation tool called the Thumbview; the tool is based on the Space-Filling Thumbnails system [13], but extends the navigational scope from the pages in a single document to the files in a software project. The Thumbview was designed to address the lack of revisitation support in current IDEs, and provide means for users to take advantage of their long-term spatial memory. It uses a spatially-stable overview that allows users to memorize locations of files and code in a software project, so that they can return directly to those locations to reduce time spent performing visual search. Following the approach of Space-Filling Thumbnails, the system lets users switch between a detailed view of a single source code file and an overview of the entire project (see Figure 1). The following sections outline four important design considerations for a spatial overview of software projects: how large software projects are (in terms of the number of files, and the size of changes), how files will be represented visually, how files are laid out in the overview, and what user interactions are possible in the overview.

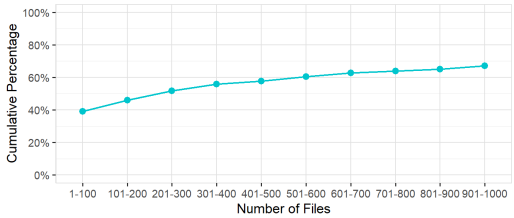

Figure 2: Cumulative percentage of number of files in the top 1000 most-forked projects on GitHub.

3.1 Size of Projects and Size of Changes

Our first consideration was how many items a spatial overview for a software project would need to display. Previous research has shown that project size has a strongly skewed distribution, with many projects having few files and a small number of projects having many files [18, 21, 49, 61]. For example, an analysis of 57,914 GitHub projects showed that the mean number of files per project was 156, but the median number was only nine [61]. To provide additional data on this issue, we downloaded the 1000 most-forked projects on GitHub and counted the number of files in each project. Figure 2 shows the cumulative percentage of the number of files per project in this sample – 39% of projects have fewer than 100 files, and 50% of projects have fewer than 250 files. Overall, these results indicate that many software projects are in the size range that will be suitable for the spatial-overview approach.

Second, the amount that a software project changes over time is another important issue for the design of spatial overviews. Previous work has shown that most projects also have a skewed distribution in terms of the size of changes (i.e., from one code 'commit' action to the next), with a large majority of commits making small changes. For example, Arafat and Riehle's analysis of 8.3 million commits from 9,363 open source projects revealed that 84% of commits changed fewer than 100 lines of code (counting code added, removed, or changed), and that 27% changed three or fewer lines [7]. Similarly, Ferreira et al. reported that for 70.8% of commits to a set of 24 well-known open source Java projects, the median number of files changed in the commit was one [23]. These results suggest

that most projects evolve slowly – meaning that overviews for most projects will be stable enough for users to develop spatial memory.

3.2 File and Directory Representation

To represent files in the overview, we use thumbnail images of the actual source code, so that users can see visual features that are provided by syntax highlighting and code structure; these can act as visual landmarks to help users differentiate between code blocks. This representation also allows people to use their familiarity with the visual features of a file gained from working with the source code – and this familiarity can be valuable despite the change in scale from a full-size view to a thumbnail, as shown in previous studies of the robustness of spatial memory across changes in scale [53]. Thumbnails may provide additional advantages over symbolic representations (e.g., file names) because the human visual system processes images faster than text [17], and studies have shown that searching for images is faster than for names [45].

For directories, Thumbview follows the approach of previous spatial-overview techniques and *flattens* the hierarchy (e.g., [51, 52]) in order to reduce the number of navigation actions required to reach any file. However, Thumbview can represent directories visually (rather than structurally) by colour-coding files by directory and by grouping a directory's files together in the overview.

3.3 Overview Layout

We chose a column-order layout to match the vertically-stretched rectangular shapes of source code files as shown in most IDEs.

Our layout approach is designed to minimize files wrapping to the next column, which creates additional space at the bottom of each column, termed *slack*. This slack space provides a useful buffer for files to grow without causing any change in the overall layout. File names are included above each thumbnail to provide visual landmarks and clearly delineate the end of a file.

The number of files, and the source code's line length, are used to determine the width of the columns. Our current implementation fixes the line length at 120 characters, but future versions of the tool will allow users to set a desired line length. Any line longer than the limit is truncated (since current coding practices encourage line lengths shorter than 120 characters, these truncations do not dramatically affect the visual appearance of the source code).

As described below, we used outline highlighting to indicate various notable properties of files and interaction results in the Thumbview (outlines were used rather than fill colour to avoid obscuring the code content of the thumbnails). Highlight colours were chosen from the IBM Design Library Color-Blind Safe palette.

3.4 Interaction

There are several ways in which a user can interact with the Thumbview tool, including opening the overview, previewing files, selecting a file, searching, and seeing recent history.

3.4.1 Opening the Thumbview and opening files from the overview. In our simulated IDE, the Thumbview can be opened with a right-click action or Control-O (although these actions can easily be changed in the case of overlaps with existing IDE command mappings). We provided Control-O as a second trigger for users who wish to operate their IDE primarily with keyboard shortcuts. All operations with the Thumbview have both mouse and keyboard versions: for example, when using the built-in search function (described below), users can cycle between files with the arrow keys and select files by pressing the Enter key.

Users open a source-code file from the overview with a left-click on the file's thumbnail; the file is automatically scrolled to the location of the click (e.g., if the user clicks in the middle of the file's thumbnail, the file view will show the middle of the file after opening). Any file can also be opened by pressing Enter when it is highlighted by they keyboard focus indicator, which can be moved using the arrow keys.

3.4.2 Previewing files. The Thumbview allows users to interactively inspect files by pressing Shift and mousing over any thumbnail. This produces a larger view of the file (Fig. 1), and shows the exact part of the file that would be visible if the user opened the file from the preview. The purpose of the interactive inspection function is to allow users to quickly examine the details of a code segment without committing to opening that file, and also to open a file to a particular region in order to avoid scrolling after opening.

3.4.3 Search. The Thumbview has an "always live" search capability, meaning that users can simply begin typing to search for a file. The search string is displayed in the middle of the overview (in transparent white text with a coloured background). Search results are indicated by highlighting the outlines of each matching file (in the same colour used for the search-string background, as shown in Fig. 1). Previous research has shown that indicating filtered search results using highlighting in a spatially-stable layout worked well with large data sets, and was faster for revisitation than typical search systems that generate a new search result list with each search filter [30]. In the Thumbview, the first search result is indicated with a dashed outline to signify that it has keyboard focus; pressing Enter will open the focused file. The user can move the focus through the search results with the up and down arrows keys.

3.4.4 History. Since the Thumbview is designed to support revisitation, we include a history mechanism that highlights recently-opened files in the overview. Research has shown that recency is a strong predictor of behaviour for both developers and general users [39, 47]; we implemented recency highlighting to support the user's current working set. Files in the recency set are all given a matching coloured outline, as seen in Fig.1. The history mechanism could also use other strategies (e.g., frequency or frecency).

4 STUDY 1: INTER-FILE NAVIGATION

Previous studies of SFTs have only considered within-document navigation, but a main part of navigation in software development is finding and re-finding files in the project. Therefore, our first study compared the Thumbview display to the navigation mechanisms that are available in standard IDEs. We conducted an in-person within-participants study of experienced programmers performing navigation tasks in a simulated IDE. The goals of the first study were to test the performance of the Thumbview as a navigation tool for realistic software tasks, and compare that performance to the standard navigation tools available in modern IDEs.

4.1 Participants, Apparatus, and Dataset

We recruited 24 current and former university students through announcements on the university website and word of mouth. The participants' ages ranged from 19 to 34 years (M = 22.9, SD = 3.5). Five participants identified as female, 16 as male, and two as non-binary (one non-response). Participants had an average of 5.3 years of coding experience (SD = 2.3), and were all familiar with modern IDEs such as Visual Studio Code or the Jetbrains suite. On average, participants stated that they spent 37% of their coding time working on unfamiliar projects. 20 participants stated that they primarily used mouse-based navigation when using GUI applications, and three participants used keyboard navigation whenever possible.

We built a simulated IDE in which participants could browse through the files in a software project using the navigation tools provided by the interface, and open files in the page view. The system was built with the JavaFX library and CSS, and was deployed locally through Windows Subsystem for Linux (WSL) on a Windows PC. The system was presented to participants through a maximized window on a 27-inch 4K monitor. Participants interacted with the system with an optical mouse and a standard QWERTY keyboard.

The data used in the study was taken from a real-world software project called Boltz, a collection of deep learning models for predicting biomolecular interaction (github.com/jwohlwend/boltz). We used a subset of the project that included 75 Python source code files, making it a representative medium-sized software system (in terms of number of files, and also average lines of code per file), based on our investigation of the 1000 most-forked projects on GitHub as of June 2025 (see Section 3.1).

4.2 Procedure and Study Conditions

Before beginning the study, participants completed a consent form and a demographics questionnaire. The study consisted of three types of navigation tasks with each interface, each followed by a short questionnaire (the NASA TLX survey, plus two additional questions about overall ease of use and support for memorization). In each task type, participants carried out several finding and re-finding trials organized into blocks. Before each task, the participant was given practice on the task (using targets that were not used in the test trials). Participants were briefly shown the two interfaces before beginning the tasks, and were introduced to the navigation tools provided with each interface before beginning the study tasks. After all three tasks were completed with one interface, participants repeated the process with the other interface (order counterbalanced). At the end of the study, participants completed a questionnaire comparing the two interfaces. Participants were paid $15 and the session took approximately 60 minutes.

4.2.1 Condition 1: Standard interface. The standard interface was designed to look similar to Microsoft Visual Studio Code, due to its popularity among developers (e.g., VSCode was the most popular IDE on the Stack Overflow Developer Survey for the last three years [1, 2, 5]). The standard interface included a simple page view with non-interactive menu bars at the top and left side of the interface, as well as several navigation components: a Search Bar, a File Explorer, a Tab Browser, and a Recency Cache (see Fig. 3). All navigation components could be used with the mouse, and the Search Bar could be selected using Ctrl+P (the shortcut used for the Command Palette in VSCode). The Search Bar was located in the top middle of the screen and was visible at all times, and was designed to look and function similarly to VSCode's Command Palette. The recency cache was accessed by interacting with the search bar, where recently accessed files would be listed in the dropdown menu below the search bar. For both interfaces, we limited the cache size to the 10 most recently visited files.

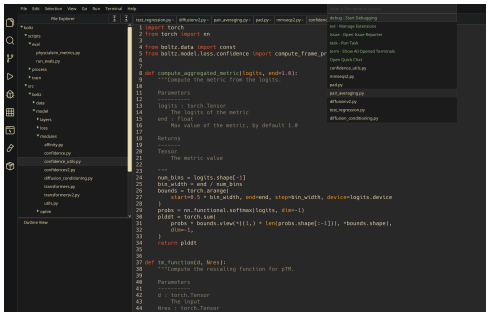

Figure 3: S1: The Standard Interface with a Search Bar, File Explorer, and Tab Browser. See Appendix for a larger version. (Image has been cropped to show interface elements in more detail)

The File Explorer utilized the standard functionality of the TreeView widget in JavaFX with directories (inner nodes) indicated by a green arrow to the left of the directory name, and leaf nodes (source code files) indicated by a file extension and the absence of the green arrow. Clicking the green arrow expanded or collapsed the corresponding directory. Two additional buttons were added to the File Explorer header: one to fully expand the entire tree in the File Explorer, and one to fully collapse the tree. The initial fully-collapsed state of the tree showed the directory of the project, as well as its direct sub-directories. The TreeView widget used half of the window height, with the other half showing the Outline and Timeline views (these were non-interactive).

The Tab browser used the functionality provided by the JavaFX TabPane widget, and included an interactive list of all open tabs above the page view; the user could click a tab to display the corresponding file in the page view, or click the X icon on a tab to close it. The JavaFX widget does not allow rearranging of tabs.

4.2.2 Condition 2: Thumbview interface. The Thumbview interface includes the page view described above, but omits the standard navigation components. Instead, the Thumbview (Fig. 1) is the sole navigation component, which can be opened and closed with right-click or Ctrl+O. The Thumbview displayed all of the relevant source code files as thumbnails in a set of columns, sized to fit a 16:9 full-window rectangular layout. The Thumbview flattened the project hierarchy [51]: files in each project directory were organized together in the overview, but no visual indication of the directory was shown, and all 75 files were laid out at the same level.

As described above, the Thumbview included a search function in which the user could simply begin typing, with the search string shown at the top middle of the screen. Files matching the search query were highlighted with a pink outline, and truncated filenames were expanded. The Thumbview also had a built-in recency cache

in which the 10 most-recently-visited files were indicated with an orange outline. Hovering over file highlighted it with a purple dashed outline and showed the full file name.

4.3 Navigation Tasks

The three types of study tasks were designed to capture common file navigation in IDEs. All three tasks were based on locating a source code file in a software project directory. The tasks were performed by participants in a fixed order with each interface, i.e. Task 1, Task 2, then Task 3 with the first interface, then all three tasks repeated in the same order with the second interface (order counterbalanced). This structure allowed participants to gain experience and familiarity with the project and the interfaces as they progressed through the tasks. In each task, participants carried out several revisitation trials grouped into blocks; the targets for each task were chosen in a random order from a set of targets unique to that task (using different task sets for the two interfaces). In each trial, participants found and opened a single target file. Each task used a different number of blocks (10, 7, 5) based on our observations from early pilots that indicated the point at which the learning curve reached a flattened plateau.

4.3.1 Task 1: Absolute Target (Fig. 4). Participants were given the name of the target file in a cue region in the middle of the screen. They were provided with the Search Bar and the File Explorer to locate the file for the Standard navigation condition, and the Thumbview and its built-in search function for the Thumbview condition. The File Explorer would collapse back to its initial state at the beginning of each block. This task was designed to simulate locating a small working set of files from an initial project state. Participants were given eight blocks of practice with two targets (different from the test targets), and then carried out 10 blocks of three targets with each interface.

4.3.2 Task 2: Relative Target (Fig. 5). Participants were given the name of an anchor file in the cue region, and were told to find the file that was either directly above or directly below the anchor file, in either the File Explorer or the Thumbview depending on the interface being tested. Participants were provided with the Search Bar and the File Explorer again for the Standard navigation condition; however, for this task the File Explorer would not collapse at the beginning of a new block. The relative-target task was designed to simulate the situation where the user remembers the relative position of a file but not the file's name. Participants were given six blocks of practice with two recurring targets, and then carried out seven blocks of three recurring targets with each interface.

4.3.3 Task 3: Working Sets (Fig. 6). Participants alternated in each block between visiting two working sets of five files. Working sets consisted of a mixture of absolute and relative targets in the target cue, and participants would also be asked to visit a file they had not seen before once per working set, while the other four trials were revisitations of targets from the previous block. We varied the target cues so that not every task would be best completed using the same navigation mechanism (in order to include a realistic decision time as the participant determined which strategy to use). Tabs and the Recency Cache were added as additional navigation tools in the Standard navigation condition, and the Recency Cache was

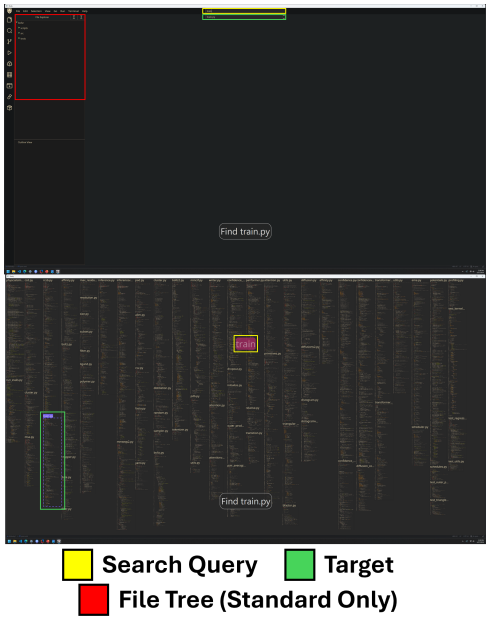

Figure 4: Top: The Standard interface for Task 1 provided with the Search Bar and File Explorer to find the target file. Bottom: The Thumbview interface for Task 1 included the built-in search function.

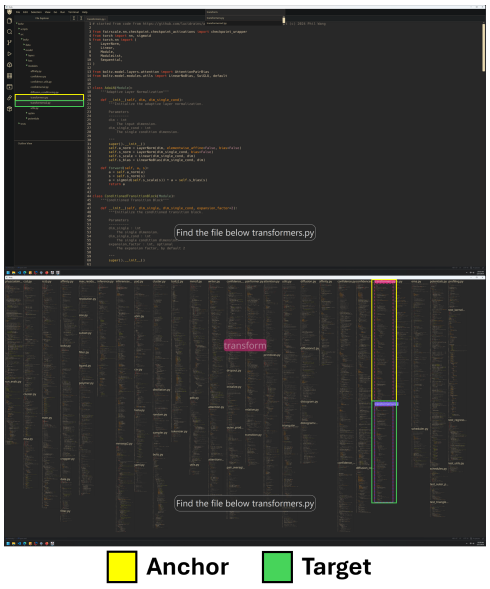

Figure 5: Top: Standard interface for Task 2. Participants found the anchor file in the File Explorer and used that location to find the target file. Bottom: Thumbview interface for Task 2. The spatially-stable overview allowed participants to learn the location of targets relative to the anchor without memorizing the target file name.

also added for the Thumbview condition. Participants were notified that they would occasionally experience a "cache miss", where they would be asked to revisit a file and it would no longer be present in the Recency Cache. Participants were given four blocks of practice with three targets to familiarize themselves with the new navigation tools, and during the task they carried out five blocks of 10 trials (2 working sets of five file targets) with each interface.

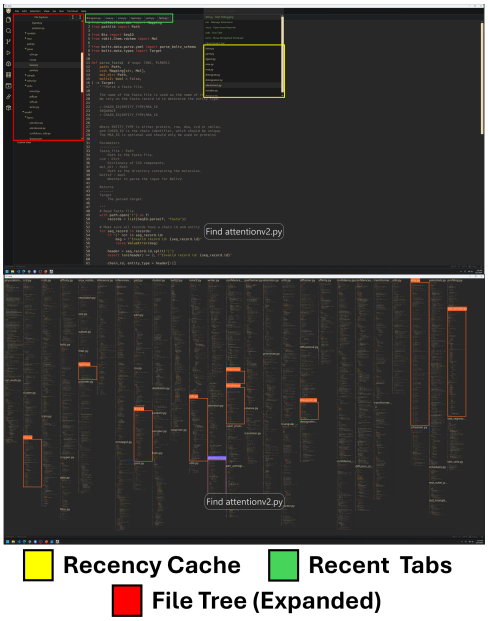

Figure 6: Top: The Standard interface for Task 3 included three navigation tools for participants to use, all of which indicated recent navigation activity. Bottom: The Thumbview interface simply indicated recently visited files with an orange outline.

4.4 Study Design

The study used a within-participants factorial design, with primary factor *Navigation Technique* (Standard or Thumbview, in counterbalanced order) and secondary factors *Task* (Absolute Target, Relative Target, Working Sets) and *Block* (1-10, 1-7, and 1-5 for the three task types). Two predefined target sets were used, and participants were evenly distributed across *Navigation Technique* × target set combinations. Each participant completed 30 test trials for Task 1 (10 blocks × 3 trials), 21 test trials for Task 2 (7 blocks × 3 trials), and 50 test trials for Task 3 (5 blocks × 10 trials), for a total of 101 trials per participant. Our primary dependent measure was completion time, and secondary measures included the number of files visited, the number of characters typed, subjective ratings based on an augmented NASA TLX survey, and preference choices.

5 STUDY 1 RESULTS

Analyses are organized below by dependent variable (completion time, files opened, perceived effort, and preference). The effect size for significant ANOVA results is reported as generalized eta squared

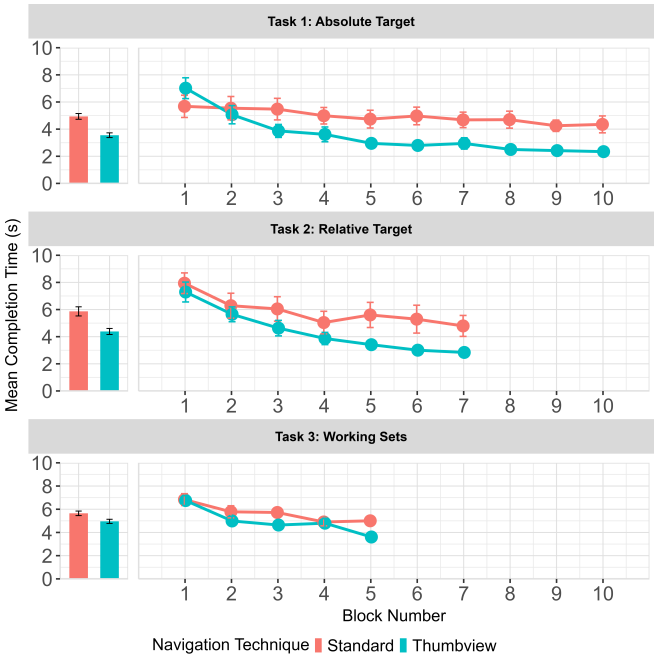

Figure 7: Bar Charts: Mean Completion Time across all Blocks for each Task. Error bars show 95% CI. Line Charts: Mean Task Completion Time by Block per Task.

(η^2). All trials were completed successfully; however, we capped completion times at 20 s based on pilot testing, which determined a reasonable upper limit for the time needed to complete trials for each task type. Five trials for the Thumbview condition were capped, and 29 trials for the Standard interface were capped. In all charts below, error bars show 95% confidence intervals.

5.1 Completion Time

Our completion time measure recorded the time between when the participant clicked to begin the trial to the moment the participant opened the correct target in the page view, including any time the participant spent opening non-target files. All completion times are reported in seconds.

For Task 1, Standard (M = 4.94 s, SD = 2.89) was 1.39 s slower that Thumbview (M = 3.55 s, SD = 2.37). For Task 2, Standard (M =5.86 s, SD = 3.86) was 1.48 s slow than Thumbview (M = 4.38 s, SD = 2.54). For Task 3, Standard (M = 5.65 s, SD = 3.45) was 0.69 s slower than Thumbview (M = 4.96 s, SD = 3.16). Figure 7 summarises completion time results. Condition × Block RM-ANOVAs for each task found significant effects of Condition and Block on completion time, as well as a significant interaction. Table 1 shows these results.

5.2 Secondary Measures

Files visited were recorded as any action that resulted in a file being opened in the page view (target or non-target). Condition × Block RM-ANOVAs found no differences in the number of files visited with each condition in Task 1, but strong differences in tasks 2 and 3. In Task 2, 30% more files were visited per trial with Standard

Task 1: Absolute Targets			
Factor	**F(df1, df2)**	***p***	η^2
Condition	F(1, 22) = 15.71	**< .001**	0.12
Block	F(9, 198) = 22.19	**< .001**	0.21
Condition × Block	F(9, 198) = 7.67	**< .001**	0.08
Task 2: Relative Targets			
Factor	**F(df1, df2)**	***p***	η^2
Condition	F(1, 22) = 13.98	**.001**	0.09
Block	F(6, 132) = 30.16	**< .001**	0.20
Condition × Block	F(6, 132) = 3.24	**.005**	0.02
Task 3: Working Sets			
Factor	**F (df1, df2)**	***p***	η^2
Condition	F(1, 22) = 4.46	**.046**	0.04
Block	F(4, 88) = 41.46	**< .001**	0.20
Condition × Block	F(4, 88) = 5.27	**< .001**	0.03

Table 1: S1: Factorial Analyses of Completion Time.

Task 1: Absolute Targets			
Factor	**F(df1, df2)**	***p***	η^2
Condition	F(1, 22) = 0.14	.710	0.00
Block	F(9, 198) = 1.61	.110	0.03
Condition × Block	F(9, 198) = 2.35	**.015**	0.05
Task 2: Relative Targets			
Factor	**F(df1, df2)**	***p***	η^2
Condition	F(1, 22) = 78.27	**< .001**	0.33
Block	F(6, 132) = 26.73	**< .001**	0.30
Condition × Block	F(6, 132) = 21.67	**< .001**	0.27
Task 1: Working Sets			
Factor	**F(df1, df2)**	***p***	η^2
Condition	F(1, 22) = 132.40	**< .001**	0.45
Block	F(4, 88) = 16.23	**< .001**	0.19
Condition × Block	F(4, 88) = 13.23	**< .001**	0.16

Table 2: S1: Factorial Analyses of Files Visited.

(M = 1.31, SD = 0.52) than Thumbview (M = 1.01, SD = 0.12). In Task 3, 31% more files were visited per trial with Standard (M = 1.38, SD = 0.69) than Thumbview (M = 1.05, SD = 0.22). Table 2 shows the ANOVA results.

We also recorded the number of characters typed with each interface's search function. Condition × Block RM-ANOVAs revealed significant main effects of Block in all three tasks (all $p < .001$), as well as significant interactions (all $p < .001$). Condition had a strong effect in Task 1 and Task 3 ($p < .001$), but not in Task 2 ($p = .18$). Participants used the search function (trials with one or more characters typed) for 64% of trials with the Standard interface and only 32.5% of trials with the Thumbview interface. Participants typed substantially fewer characters on revisitations with the Thumbview interface, which can be seen for Task 1 in Fig. 8.

5.3 Subjective Responses

After completing all three tasks with an interface, participants filled out an augmented TLX questionnaire, described their file-finding strategy with the interface, and listed any difficulties they experienced. Ratings from the augmented TLX are shown in Fig. 9.

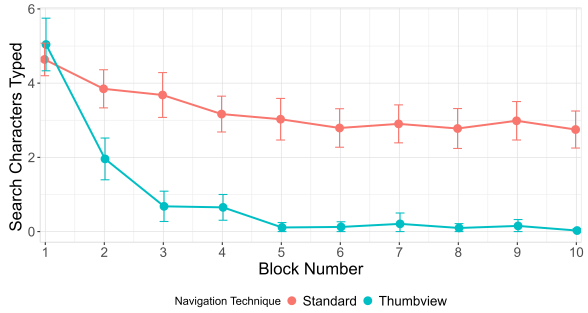

Figure 8: S1: Search Characters Typed per Block by Interface.

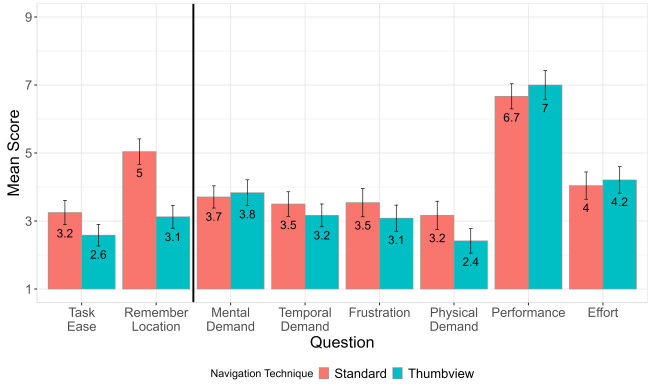

Figure 9: S1: Subjective Responses (9-point scale).

We applied the Aligned Rank Transform and carried out one-way ANOVAs on each question from the augmented TLX using Condition as the factor. Analyses revealed a significant main effect of Condition for "Ability to Remember Location" ($F(1, 23) = 19.97$, $p < .001$), with participants rating the Standard interface as significantly better at helping them remember the location of files than the Thumbview interface (despite the better performance of the Thumbview, which may reflect participants' existing familiarity with the standard navigation tools). We also observed a significant main effect of Condition for Physical Demand, $F(1, 23) = 4.55$, $p = .44$, indicating that participants found the Standard interface more physically demanding than the Thumbview. No other significant differences were observed (all $p \geq .13$).

In terms of strategy, many participants adopted the same approach with the Thumbview: "The main strategy was memory. For a new file, I would use [search] once or twice to locate it and from then on I would remember its approximate location"; "I mostly tried to use spatial memory..." With the Standard interface participants relied heavily on the search function: "My strategy was to use the search as much as possible to avoid having to scroll and find the file manually"; "Mostly relied on search, especially when the file tree felt too large to navigate."

Some participants described encountering difficulties with the Thumbview when targets were small: "...when the target was very small, it was hard to tell what file it was until I hovered over it". The difficulties encountered with the Standard interface often revolved

around scrolling in the File Explorer: "(searching) through the (File Explorer) was difficult because there were too many folders and I would forget where I was"; "When the file tree (was) fully open ... it was easy to lose track of the file I was trying to click".

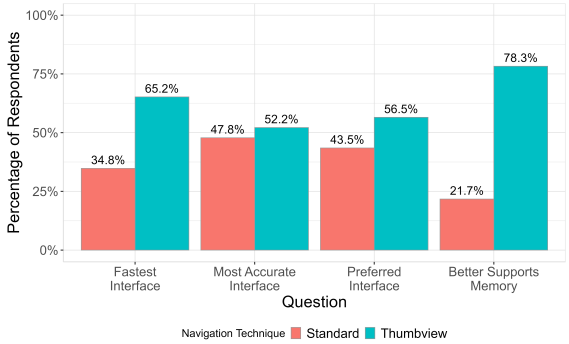

Figure 10: S1: Preference Responses.

In results from a final questionnaire comparing the two interfaces, participants showed a clear preference for the Thumbview interface as the fastest interface (65.2% vs 34.8%) and for supporting memory (78.3% vs 21.7%). Preference data is summarized in Fig. 10.

We asked participants to justify each of their preferences with a short written response. Participants found that the Thumbview was faster due to the spatial layout, and less reliance on search: "Once you had the location of the file memorized, it was very quick and easy to find on Thumbview"; "[the] Standard interface forced me to mainly only use search, whereas with Thumbview I could click around the general area where I remembered the file to be" Participants cited relative targets as the main reason why they perceived the Thumbview to be more accurate: "When searching for files above/below an anchor file ... I could quickly click the file I wanted and I didn't actually have to open (the anchor file) to get the file above/below it"; "Thumbview with the recency cache and the location on the overview was much easier... Standard Interface was a lot more clunky and had me choosing the wrong file first".

6 STUDY 2: EFFECTS OF VISUAL CHANGES

The second study examined the effects of changes in the spatial layout of the Thumbview on participants' ability to revisit targets, using unique local visual movements in file locations to reflect the evolution of a software project over time. In the study, we examined the degree of disruption caused by a move (in terms of revisitation time), the number of additional revisitations to a moved target before participants regained their previous level of performance, and their subjective ratings of effort, frustration, mental demand, and perceived performance with the system. Note that because this evaluation was focused on the specific effects of changes to the Thumbview, we did not include the standard IDE in this study.

6.1 Participants, Apparatus, and Codebase

20 people participated in the study: 10 individuals who had participated in the previous study, and 10 additional people who had no experience with the Thumbview interface. The second group completed the first task of Study 1 in order to familiarize themselves with the interface before beginning the study. Participants were between the ages of 19 and 36 (M = 24.6, SD = 4.99). Four participant identified as female and 16 as male. Participants had an average of 6.8 years of coding experience (SD = 4.57). The second study used a version of the Study 1 system that was modified to remove the standard interface. The system was presented to participants through a maximized window on a 27-inch 4k monitor, and participants interacted with the system using a mouse and standard keyboard. The same codebase from the previous study was used.

6.2 Types of Movement in the Overview

Move Types were chosen to examine how different types of local moves of target locations affected revisitation ability. Diagrams for each move type are included in Fig. 11. The five Move Types were:

- Move Down. This change involved an expansion of the source code file directly above the target file by 144 lines of code, causing the target file to move down vertically within the same column.
- Move Right. This change involved an expansion of a source code file above the target file in the same column by 128 lines, with the growth of the file being enough to shift the target file to the top of the next column (slack in the next column prevented cascading effects – see Section 3.3).
- Move Up. For this change the target file was moved to the parent folder in the directory hierarchy, causing it to shift to the top of the same column.
- Move Left. For this change the target file was moved to an adjacent folder at the same level in the directory hierarchy, resulting in a shift to the column it's left.
- No Move. We designed the Thumbview such that any deletion of code would not cause any files to shift in the overview, it would simply create an empty space (slack) where that code used to be. For this reason, the target file does not move for this Move Type.

6.3 Procedure

After filling out a consent form and demographics questionnaire, participants performed five simple finding and re-finding tasks, each corresponding to a different local move, with the Thumbview interface. For each Move Type described above, participants were given a three-target working set to revisit in each block. The targets were chosen such that only one of the three targets would move per Move Type (see Figure 11).

Participants completed six blocks of training (the *memorization phase*) to familiarize themselves with the locations of the target files in the Thumbview. The system then switched to a new layout of the Thumbview, based on the current Move Type. Participants then revisited the same three target files for six more blocks (the *recovery phase*). Participants were not allowed to use the search function for the first five seconds of each trial, to better explore how participants used their spatial memory to try and find the files.

6.4 Study Design

The second study used a within-participants factorial design with two factors. The primary factor was *Move Type* (Move Down, Move

Right, Move Up, Move Left, No Move) and *Block* (1-12) was a secondary factor. Our primary dependent measure was completion time, and secondary measures included search characters typed and subjective responses from the effort questionnaire (using a subset of the NASA-TLX questions). Move Types were counterbalanced using a 10x5 Latin square. Each participant completed six memorization blocks and six recovery blocks with three targets per block for each of the five Change Types, for a total of 180 trials.

7 STUDY 2: RESULTS

Effect size for significant ANOVA results is reported as generalized eta squared (η^2). Error bars in charts show 95% confidence intervals.

7.1 Effect of Moves on Completion Time

Our primary analysis focuses on the effect of local moves on completion time. Therefore, we use only blocks 6-12 (the last block before the change, and the six blocks following the change). We carried out a 5×7 (Move Type × Block) ANOVA across all levels of Move Type and Blocks 6-12. We found significant main effects of Move Type and Block, as well as a significant interaction (Table 3).

Factor	F(df1, df2)	p	η^2
Block (6-12)	F(1, 19) = 107.56	< **.001**	0.25
Move Type	F(4, 76) = 11.05	< **.001**	0.27
Block (6-12) × Move Type	F(4, 76) = 12.15	< **.001**	0.16

Table 3: Factorial Analysis of Move Type and Block (6-12).

To investigate the degree and length of disruption caused by each change, we compared Block 6 to individual subsequent Blocks for each Move Type, until a significant difference was no longer observed. Results are summarized in Table 5 and in Figure 11. As shown in the table, four of the five tasks led to a significant time increase in Block 7 (immediately after the move); the amount of increase ranged from less than a second (Move Down) to 8.5 seconds (Move Right). However, by Block 8, the disruption was only significant for two tasks, and the amount of increase was only one second or less; by Block 9, there were no longer any differences.

Move Type	B6 CT	B6 & B7	B6 & B8	B6 & B9
Move Down	3.37	+0.90 *	-0.30 *	-0.29
Move Right	3.13	+8.57 *	+1.07 *	+0.24
Move Up	3.13	+5.32	+0.37 *	+0.38
Move Left	2.90	+1.29	+0.34 *	+0.37
No Move	3.65	-0.05	-0.24	-0.04

Table 4: S2: Mean Completion Time (s) for Block 6 and change in Completion Time relative to Block 6 for Blocks 7–9. Positive values indicate an increase in Completion Time. Stars indicate a significant difference in the comparison.

7.2 Effect of Participant Experience

To assess whether amount of experience with the Thumbview affected the degree of disruption caused by a move, we carried out an ANOVA comparing our 10 participants who had taken part in

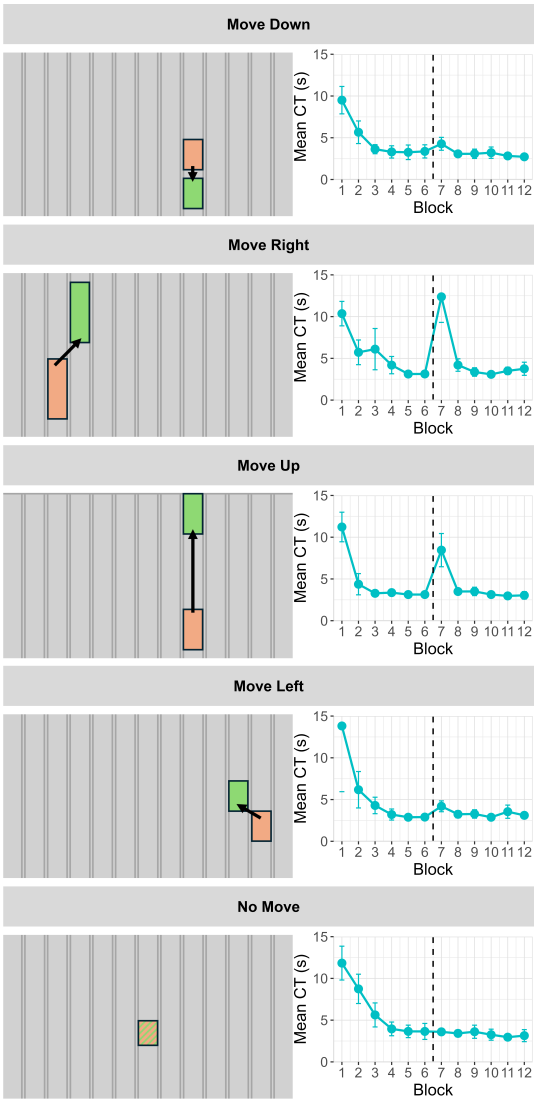

Figure 11: S2: Diagrams visualizing each Move Type in Study 2 paired with charts for Completion Time by Block per Move Type. In the diagrams, the target's original location is shown in orange, and its location after the move is shown in green (note the target did not move for No Move).

the first study (and thus had more experience) with the 10 new participants. Experience Group was therefore a between-subjects factor and Block was a within-subjects factor. There was no main effect of Experience Group on Completion Time (F(1, 18) = 1.49, p = .24); there was an interaction between the factors, shown in Figure 11 as difference in amount of disruption in Blocks 6 and 7.

Initially, we had expected that increased experience would lead to greater disruption after the move – that is, more-experienced users would be more reliant on their spatial memory, and thus more affected by a move. However, the opposite was true: more-experienced users were less affected by the move (an increase of

2.1 s) than less-experienced users (an increase of 4.2 s). This result shows that layout changes did not impair experienced users.

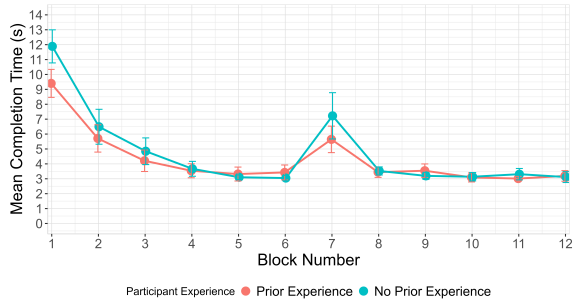

Figure 12: S2 Mean Completion Time per Block by Participant Experience Group. Error bars show 95% CI.

7.3 Subjective Responses

After working with each Move Type, participants answered questions about the difficulty of finding files after the move, the amount of disruption caused by the move, and their perceived performance in dealing with the move. At the end of the study, participants also answered summary questions (drawn from the TLX survey) about overall effort, frustration, mental demand, and perceived performance with the system. Figures 13 and 14 show these results. For all cases, participants rated the difficulty and disruption of the moves as low (3.0 or less on a 9-point scale), and rated their performance as high (7.6 or more). Responses to the end-of-study questions also showed that participants did not feel that the moves had caused a great deal of effort, frustration, or mental demand (all means at or below 2.8 on a 9-point scale), and that perceived performance was high overall (7.7 on a 9-point scale).

Participants could optionally include written comments about their subjective ratings which provided some insight into their reasoning. Participant 11 signalled that the moves in the position of targets were intuitive, noting that "the shifts in the locations of the files made sense to me", while Participants 8 and 20 mentioned how small shifts were easy to adapt to: "I actually sometimes found it easier after they changed ... because files were close to the original location"; "I find learning the new location of the file still relatively easy because the file is located close to its previous location".

8 DISCUSSION

The two studies provide several main results for our investigations into spatial overviews for software projects. For Study 1 (performance of spatial overviews for inter-file navigation):

- The Thumbview significantly reduced navigation times (by 11.5% to 26.6%) across all three tasks compared to the Standard interface;
- Participants opened fewer non-target files and used search less often with the Thumbview, suggesting less exploratory effort and fewer mechanical actions;
- the Thumbview was rated higher in terms of speed and ability to help participants remember the location of files.

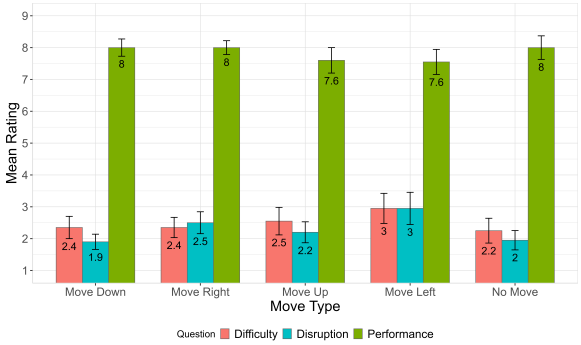

Figure 13: S2: Subjective responses after each move type. Each question asked about the participant's experience with that move type.

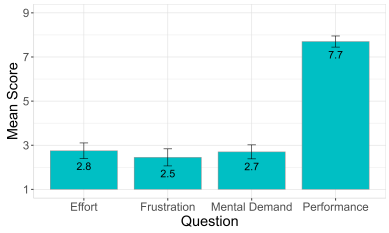

Figure 14: S2: TLX responses at end of study. Questions asked about participants' overall experience after seeing all move types.

For Study 2 (effects of target movement on revisitation):

- Four of five types of move did cause a short-term reduction in performance, ranging from an additional second to more than eight seconds;
- Participants quickly recovered to an expert level of performance after only one or two revisitations;
- The effects of target movement had less of an effect on users who had more experience with the layout than on those with less experience.

Overall, the Thumbview enabled faster navigation and fewer navigation actions compared to standard IDE navigation tools. Additionally, the benefits of spatial navigation were resilient to local movements in target files, with participants quickly returning to an expert level of performance. In the following sections, we provide explanations for these results, discuss their implications for real-world IDE use, and outline limitations and directions for future work.

8.1 Explanations for main results

Despite participants being more familiar with the tools available in the Standard IDE navigation interface, Thumbview navigation was faster across all tasks in the first study. We see three possible reasons why the spatial overview was faster. First, the Thumbview allowed participants to navigate with fewer actions than the Standard condition, and each reduction in the number of actions can save time in the overall navigation task. Second, the spatial overview better enables increased efficiency as the user gains experience with the

targets: as participants learned the file locations, they were able to select them more quickly, whereas in the Standard interface, the navigation tools do not enable faster operation as the user gains experience with a particular file. Third, the multiplicity of navigation methods in the Standard interface may have also slowed users who spent time making a decision about which method to use for a given task; this decision time did not occur in the Thumbview because all navigation styles (search-based, exploration-based, and memory-based) occur through the same interface.

In the second study, we showed that participants' spatial memory was disrupted immediately following the movement of a target file. Different move types in this study led to different degrees of disruption across the different move types, and we believe that the size of the layout shift caused by the move is the reason behind these differences: the Move Right change led to the target shifting to the next column (which also led to it moving up in the layout), and this meant that the target changed both in its vertical and horizontal position; and the Move Up change was the largest shift of any move (from the bottom to the top of the layout). In contrast, the Move Left and Move Down changes kept the target close to its original position. Future studies will explore the effects of different sizes of layout shift, and will also investigate visual annotations that can help users find files that have moved a substantial amount from their previous location. For example, we are developing "change visualization" methods for the Thumbview that can visually guide users from old locations to new ones, for files that have moved since the overview was last seen.

Despite the temporary reductions in performance, Study 2 also clearly showed that users were not permanently disrupted by moves to target files: in all cases, performance returned to the previous level of expertise after the first or second revisitation after the move. We believe that this strong recovery in performance arises from the natural robustness of spatial memory – for example, previous work has shown that once users learn the locations of a set of objects, the layout of those objects can be scaled, skewed, or rotated with very little negative effect on retrieval time [53]. Second, even when a target is not exactly where the user remembered it to be, people have a natural ability to carry out quick localized visual search in a general area – and as long as spatial memory can direct the user's attention to the correct region, localized search can usually quickly find an item that has moved by a small amount. Our study findings also show that users quickly adapt to the new location, possibly by updating their mental map of where targets are located.

8.2 Generalizing the findings, and future work

Our studies have made progress in generalizing the spatial approach to real-world application, although there are several issues to consider. First, we believe that our study results have a good chance of holding in real-world software-development scenarios. The dataset used in our studies was a taken from a real open-source project, and the study tasks were representations of what developers actually do (e.g., based on the strong revisitation patterns seen in previous work [11, 32, 35, 42]). In addition, the types of moves used for Study 2 were representative of real-world change metrics showing that most commits involve a small number of lines [23], and our users were all experienced programmers, with more than four years of coding experience. We did not consider the effect of participant familiarity with the codebase (all participants were unfamiliar with the codebase used in the study), however we believe that increasing familiarity with a project would only increase the effectiveness of the Thumbview as a navigation tool.

We plan further work to investigate how spatial overviews will work within the broad range of projects that exist. We believe that the current implementation of the Thumbview is best suited to small-to-medium-sized projects (e.g., fewer than 200 files); larger projects would lead to very small thumbnails that are tightly packed in the overview (and we are unsure how an extremely dense view will affect spatial memory). However, there are other potential ways of organizing the overview that could accommodate larger sizes: for example, the layout approach could switch to using tiles for each file rather than thumbnails, could use a spatially-stable layout algorithm such as Hilbert-Moore Treemaps (e.g., [59]), or could move to a hierarchy of overviews (e.g., one at the package level, and one for each package at the file level).

Our studies also did not test within-file targets (i.e., specific functions or code segments), and these tasks are an obvious part of software navigation. However, we believe that the advantages of the Thumbview seen in our studies will naturally extend to intra-file navigation tasks: previous work with SFTs has already shown that spatial overviews are effective for finding targets within files, and that scrolling is the main bottleneck in standard interfaces – a limitation that is likely to be true for IDEs as well. The capability of the Thumbview that allows navigation to specific parts of a file (through its preview feature, as described in Section 3.4.2) will allow users to remember and quickly move to specific file locations.

We must also consider whether the improvement in navigation times using the Thumbview are important in a broader view of software development activities. Although the individual time differences between Thumbview and Standard navigation were relatively small (a second or two), we believe that techniques that enable user automaticity can make a substantial difference in usability. In many real-world tasks, navigation is not the activity that the user is trying to focus on – instead, they are carrying out a domain task that involves remembering data and planning a solution. In these contexts, a few seconds may feel like a big difference when the user is trying to complete a task that requires navigation, and any problems in navigation could cause a loss of focus on that activity. Providing users with a way to navigate that feels automatic and does not require cognitive attention could substantially reduce friction and frustration in the use of an IDE.

A real-world version of the Thumbview tool will need to provide greater support to developers in terms of what types of project files they include in the overview. In our studies, we included only source files in the Thumbview, and did not show others such as git files, images, and documentation; a full implementation would have to include a way for users to specify what type of files they would like to be displayed. A simple solution would be to have configuration options with a default set of allowed extensions. This also raises the issue of how the user accesses any files that are not in the Thumbview; we assume that low-frequency files can be found through other means (e.g., the user could switch back to the standard mechanisms, although further studies are needed to

determine whether there will be interference between the different navigation techniques).

Finally, the current implementation of the Thumbview groups files into directories, but does not visually highlight the hierarchical structure in the project (although this is possible). Several participants in the first study noted the lack of directory structure (e.g., one participants stated "Having a general folder structure aids me a lot in remembering where files are"; another stated "it didn't help me understand the directory structure as well as (the Standard interface)". Visualizing directory structure is therefore an important area of future work. We have explored several options for visually displaying directory structure in the Thumbview, including highlight borders grouping files from the same directory on mouse-over (and indicating parent directories), and multiple levels of hierarchy, i.e. a spatial overview of the top level project directory where clicking a child directory would open a spatial overview of its contents. However, this type of solution depends heavily on the arrangement of the files in the software project: for example, deeply nested directories would have to be treated specially.

9 CONCLUSION

Previous research has shown that spatially-stable overviews can allow users to take advantage of their increasing familiarity with a document by building up spatial memory of locations. However, little is known about whether the spatial approach can be extended to inter-file navigation, and whether changes to the spatial layout of the overview will negatively affect performance. Software development scenarios provide a context for studying both of these questions. We developed the Thumbview interface as an adaptation of Space-Filling Thumbnails that is appropriate for use in an IDE. We then carried out two studies to investigate inter-file navigation performance and the effects of layout changes in the overview. The first study showed that the Thumbview allowed participants to find and revisit files significantly faster than standard IDE navigation mechanisms; participants also strongly preferred the Thumbview interface. The second study tested performance before and after a target file moved as a result of changes in the project, and the study showed that although there were reductions in performance immediately after the move, users quickly recovered and regained their expert level of performance. Overall, our work suggests that spatial overviews and spatial navigation can be a valuable addition to modern IDEs, with the potential to reduce the time and effort developers spend searching for and revisiting files – even as projects change over time. Future work will explore extending this approach to finer-grained navigation (such as within-file navigation), scaling to larger projects, and integrating the Thumbview into real-world development environments.

REFERENCES

[1] 2023. *Stack Overflow 2023 Developer Survey.* https://survey.stackoverflow.co/2023/

[2] 2024. *Stack Overflow 2024 Developer Survey.* https://survey.stackoverflow.co/2024/

[3] 2025. *OneTab.* https://www.one-tab.com/

[4] 2025. *SessionBuddy.* https://sessionbuddy.com/

[5] 2025. *Stack Overflow 2025 Developer Survey.* https://survey.stackoverflow.co/2025/

[6] Eytan Adar, Jaime Teevan, and Susan T. Dumais. 2008. Large scale analysis of web revisitation patterns *(CHI '08)*. Association for Computing Machinery, New York, NY, USA, 1197–1206. https://doi.org/10.1145/1357054.1357241

[7] O. Arafat and D. Riehle. 2009. The Commit Size Distribution of Open Source Software. In *2009 42nd Hawaii International Conference on System Sciences*. 1–8. https://doi.org/10.1109/HICSS.2009.421

[8] Saulius Astromskis, Gabriele Bavota, Andrea Janes, Barbara Russo, and Massimiliano Di Penta. 2017. Patterns of developers behaviour: A 1000-hour industrial study. *Journal of Systems and Software* 132 (2017), 85–97. https://doi.org/10.1016/j.jss.2017.06.072

[9] Ivan Bacher, Brian Mac Namee, and John Kelleher. 2017. The Code-Map Metaphor - A Review of Its Use Within Software Visualisations. In *Proceedings of the 12th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP 2017) - IVAPP*. INSTICC, SciTePress, 17–28. https://doi.org/10.5220/0006072300170028

[10] Abir Bouraffa, Gian-Luca Fuhrmann, and Walid Maalej. 2023. Developers' Visuo-Spatial Mental Model and Program Comprehension. In *Proceedings of the 45th International Conference on Software Engineering* (Melbourne, Victoria, Australia) *(ICSE '23)*. IEEE Press, 1920–1932. https://doi.org/10.1109/ICSE48619.2023.00163

[11] Andrew Bragdon, Robert Zeleznik, Steven P. Reiss, Suman Karumuri, William Cheung, Joshua Kaplan, Christopher Coleman, Ferdi Adeputra, and Joseph J. LaViola. 2010. Code bubbles: a working set-based interface for code understanding and maintenance. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Atlanta, Georgia, USA) *(CHI '10)*. Association for Computing Machinery, New York, NY, USA, 2503–2512. https://doi.org/10.1145/1753326.1753706

[12] Joseph Chee Chang, Yongsung Kim, Victor Miller, Michael Xieyang Liu, Brad A Myers, and Aniket Kittur. 2021. Tabs.do: Task-Centric Browser Tab Management. In *The 34th Annual ACM Symposium on User Interface Software and Technology* (Virtual Event, USA) *(UIST '21)*. Association for Computing Machinery, New York, NY, USA, 663–676. https://doi.org/10.1145/3472749.3474777

[13] Andy Cockburn, Carl Gutwin, and Jason Alexander. 2006. Faster document navigation with space-filling thumbnails. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Montréal, Québec, Canada) *(CHI '06)*. Association for Computing Machinery, New York, NY, USA, 1–10. https://doi.org/10.1145/1124772.1124774

[14] Andy Cockburn, Carl Gutwin, and Saul Greenberg. 2007. A predictive model of menu performance *(CHI '07)*. Association for Computing Machinery, New York, NY, USA, 627–636. https://doi.org/10.1145/1240624.1240723

[15] Andy Cockburn and Bruce McKenzie. 2001. What do web users do? An empirical analysis of web use. *International Journal of Human-Computer Studies* 54, 6 (2001), 903–922. https://doi.org/10.1006/ijhc.2001.0459

[16] Andy Cockburn, Joshua Savage, and Andrew Wallace. 2005. Tuning and testing scrolling interfaces that automatically zoom. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Portland, Oregon, USA) *(CHI '05)*. Association for Computing Machinery, New York, NY, USA, 71–80. https://doi.org/10.1145/1054972.1054983

[17] Veronika Coltheart. 1999. *Fleeting memories: Cognition of brief visual stimuli.* Mit Press. https://doi.org/10.5860/CHOICE.37-2453

[18] Ozren Dabic, Emad Aghajani, and Gabriele Bavota. 2021. Sampling Projects in GitHub for MSR Studies. In *2021 IEEE/ACM 18th International Conference on Mining Software Repositories (MSR)*. 560–564. https://doi.org/10.1109/MSR52588.2021.00074

[19] R. DeLine, M. Czerwinski, B. Meyers, G. Venolia, S. Drucker, and G. Robertson. 2006. Code Thumbnails: Using Spatial Memory to Navigate Source Code. In *Visual Languages and Human-Centric Computing (VL/HCC'06)*. 11–18. https://doi.org/10.1109/VLHCC.2006.14

[20] Robert DeLine and Kael Rowan. 2010. Code canvas: zooming towards better development environments. In *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering - Volume 2* (Cape Town, South Africa) *(ICSE '10)*. Association for Computing Machinery, New York, NY, USA, 207–210. https://doi.org/10.1145/1810295.1810331

[21] Michael Dorner, Maximilian Capraro, Ann Barcomb, and Krzysztof Wnuk. 2020. A replication study on measuring the growth of open source. *arXiv preprint arXiv:2008.07753* (2020).

[22] S.C. Eick, J.L. Steffen, and E.E. Sumner. 1992. Seesoft-a tool for visualizing line oriented software statistics. *IEEE Transactions on Software Engineering* 18, 11 (1992), 957–968. https://doi.org/10.1109/32.177365

[23] Mívian Ferreira, Diego Gonçalves, Mariza Bigonha, and Kecia Ferreira. 2023. Characterizing Commits in Open-Source Software. In *Proceedings of the XXI Brazilian Symposium on Software Quality* (Curitiba, Brazil) *(SBQS '22)*. Association for Computing Machinery, New York, NY, USA, Article 7, 10 pages. https://doi.org/10.1145/3571473.3571508

[24] Leah Findlater and Joanna McGrenere. 2004. A comparison of static, adaptive, and adaptable menus. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Vienna, Austria) *(CHI '04)*. Association for Computing Machinery, New York, NY, USA, 89–96. https://doi.org/10.1145/985692.985704

[25] J. Froehlich and P. Dourish. 2004. Unifying artifacts and activities in a visual tool for distributed software development teams. In *Proceedings. 26th International Conference on Software Engineering*. 387–396. https://doi.org/10.1109/ICSE.2004.1317461

[26] Victor M. González and Gloria Mark. 2004. "Constant, constant, multi-tasking craziness": managing multiple working spheres. In *Proceedings of the SIGCHI*

Conference on Human Factors in Computing Systems (Vienna, Austria) *(CHI '04)*. Association for Computing Machinery, New York, NY, USA, 113–120. https://doi.org/10.1145/985692.985707

[27] Saul Greenberg and Ian H. Witten. 1993. Supporting command reuse: mechanisms for reuse. *International Journal of Man-Machine Studies* 39, 3 (1993), 391–425. https://doi.org/10.1006/imms.1993.1066

[28] Saul Greenberg and Ian H. Witten. 1993. Supporting command reuse: mechanisms for reuse. *International Journal of Man-Machine Studies* 39, 3 (1993), 391–425. https://doi.org/10.1006/imms.1993.1066

[29] Carl Gutwin, Andy Cockburn, and Nickolas Gough. 2017. A Field Experiment of Spatially-Stable Overviews for Document Navigation. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems* (Denver, Colorado, USA) *(CHI '17)*. Association for Computing Machinery, New York, NY, USA, 5905–5916. https://doi.org/10.1145/3025453.3025905

[30] Carl Gutwin, Michael van der Kamp, Jeremy Storring, Andy Cockburn, and Cody Phillips. 2020. Testing the Limits of the Spatial Approach: Comparing Retrieval and Revisitation Performance of Spatial and Paged Data Organizations for Large Item Sets. In *Graphics Interface 2020*. https://openreview.net/forum?id=fCBIfLCpD1

[31] Nathan Hahn, Joseph Chee Chang, and Aniket Kittur. 2018. Bento Browser: Complex Mobile Search Without Tabs. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems* (Montreal QC, Canada) *(CHI '18)*. Association for Computing Machinery, New York, NY, USA, 1–12. https://doi.org/10.1145/3173574.3173825

[32] Austin Z. Henley and Scott D. Fleming. 2014. The patchworks code editor: toward faster navigation with less code arranging and fewer navigation mistakes. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Toronto, Ontario, Canada) *(CHI '14)*. Association for Computing Machinery, New York, NY, USA, 2511–2520. https://doi.org/10.1145/2556288.2557073

[33] Austin Z. Henley, Scott D. Fleming, and Maria V. Luong. 2017. Toward Principles for the Design of Navigation Affordances in Code Editors: An Empirical Investigation. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems* (Denver, Colorado, USA) *(CHI '17)*. Association for Computing Machinery, New York, NY, USA, 5690–5702. https://doi.org/10.1145/3025453.3025645

[34] William P. Jones and Susan T. Dumais. 1986. The spatial metaphor for user interfaces: experimental tests of reference by location versus name. *ACM Trans. Inf. Syst.* 4, 1 (Jan. 1986), 42–63. https://doi.org/10.1145/5401.5405

[35] Mik Kersten and Gail C. Murphy. 2006. Using task context to improve programmer productivity. In *Proceedings of the 14th ACM SIGSOFT International Symposium on Foundations of Software Engineering* (Portland, Oregon, USA) *(SIGSOFT '06/FSE-14)*. Association for Computing Machinery, New York, NY, USA, 1–11. https://doi.org/10.1145/1181775.1181777

[36] Benjamin P. Klein and Austin Z. Henley. 2021. CodeRibbon: More Efficient Workspace Management and Navigation for Mainstream Development Environments. In *2021 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. 604–608. https://doi.org/10.1109/ICSME52107.2021.00065

[37] A.J. Ko, Htet Htet Aung, and B.A. Myers. 2005. Eliciting design requirements for maintenance-oriented IDEs: a detailed study of corrective and perfective maintenance tasks. In *Proceedings. 27th International Conference on Software Engineering, 2005. ICSE 2005.* 126–135. https://doi.org/10.1109/ICSE.2005.1553555

[38] Amy J. Ko, Brad A. Myers, Michael J. Coblenz, and Htet Htet Aung. 2006. An Exploratory Study of How Developers Seek, Relate, and Collect Relevant Information during Software Maintenance Tasks. *IEEE Transactions on Software Engineering* 32, 12 (2006), 971–987. https://doi.org/10.1109/TSE.2006.116

[39] Joseph Lawrance, Christopher Bogart, Margaret Burnett, Rachel Bellamy, Kyle Rector, and Scott D. Fleming. 2013. How Programmers Debug, Revisited: An Information Foraging Theory Perspective. *IEEE Transactions on Software Engineering* 39, 2 (2013), 197–215. https://doi.org/10.1109/TSE.2010.111

[40] Kevin Lynch. 1964. *The image of the city*. MIT press.

[41] B. McKenzie and A. Cockburn. 2001. An empirical analysis of web page revisitation. (2001), 9 pp.–. https://doi.org/10.1109/HICSS.2001.926533

[42] G.C. Murphy, M. Kersten, and L. Findlater. 2006. How are Java software developers using the Eclipse IDE? *IEEE Software* 23, 4 (2006), 76–83. https://doi.org/10.1109/MS.2006.105

[43] Douglas L Nelson, Valerie S Reed, and John R Walling. 1976. Pictorial superiority effect. *Journal of experimental psychology: Human learning and memory* 2, 5 (1976), 523. https://doi.org/10.1037/0278-7393.2.5.523

[44] Daniel L Odell, Richard C Davis, Andrew Smith, and Paul K Wright. 2004. Toolglasses, marking menus, and hotkeys: a comparison of one and two-handed command selection techniques. (2004).

[45] Allan Paivio and Ian Begg. 1974. Pictures and words in visual search. *Memory & Cognition* 2, 3 (1974), 515–521. https://doi.org/10.3758/BF03196914

[46] C. Parnin and C. Gorg. 2006. Building Usage Contexts During Program Comprehension. In *14th IEEE International Conference on Program Comprehension (ICPC'06)*. 13–22. https://doi.org/10.1109/ICPC.2006.14

[47] David Piorkowski, Scott Fleming, Christopher Scaffidi, Christopher Bogart, Margaret Burnett, Bonnie John, Rachel Bellamy, and Calvin Swart. 2012. Reactive information foraging: an empirical investigation of theory-based recommender systems for programmers. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Austin, Texas, USA) *(CHI '12)*. Association for Computing Machinery, New York, NY, USA, 1471–1480. https://doi.org/10.1145/2207676.2208608

[48] George Robertson, Mary Czerwinski, Kevin Larson, Daniel C. Robbins, David Thiel, and Maarten van Dantzich. 1998. Data mountain: using spatial memory for document management. In *Proceedings of the 11th Annual ACM Symposium on User Interface Software and Technology* (San Francisco, California, USA) *(UIST '98)*. Association for Computing Machinery, New York, NY, USA, 153–162. https://doi.org/10.1145/288392.288596

[49] Guzel Safiullina, Aidar Gumerov, Gcinizwe Dlamini, and Giancarlo Succi. 2024. Preliminary study: Exploring github repository metrics. In *Future of Information and Communication Conference*. Springer, 579–591. https://doi.org/10.1007/978-3-031-53960-2_38

[50] Joey Scarr, Andy Cockburn, and Carl Gutwin. 2013. Supporting and Exploiting Spatial Memory in User Interfaces. *Foundations and Trends in Human-Computer Interaction* 6, 1 (12 2013), i–84. https://doi.org/10.1561/1100000046 arXiv:https://www.emerald.com/fthci/article-pdf/6/1/i/10974952/1100000046en.pdf

[51] Joey Scarr, Andy Cockburn, Carl Gutwin, and Andrea Bunt. 2012. Improving command selection with CommandMaps. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Austin, Texas, USA) *(CHI '12)*. Association for Computing Machinery, New York, NY, USA, 257–266. https://doi.org/10.1145/2207676.2207713

[52] Joey Scarr, Andy Cockburn, Carl Gutwin, Andrea Bunt, and Jared E. Cechanowicz. 2014. The usability of CommandMaps in realistic tasks. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Toronto, Ontario, Canada) *(CHI '14)*. Association for Computing Machinery, New York, NY, USA, 2241–2250. https://doi.org/10.1145/2556288.2556976

[53] Joey Scarr, Andy Cockburn, Carl Gutwin, and Sylvain Malacria. 2013. Testing the robustness and performance of spatially consistent interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Paris, France) *(CHI '13)*. Association for Computing Machinery, New York, NY, USA, 3139–3148. https://doi.org/10.1145/2470654.2466430

[54] Joey Scarr, Andy Cockburn, Carl Gutwin, and Philip Quinn. 2011. Dips and ceilings: understanding and supporting transitions to expertise in user interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Vancouver, BC, Canada) *(CHI '11)*. Association for Computing Machinery, New York, NY, USA, 2741–2750. https://doi.org/10.1145/1978942.1979348

[55] Arthur I Schulman. 1973. Recognition memory and the recall of spatial location. *Memory & Cognition* 1, 3 (1973), 256–260. https://doi.org/10.3758/BF03198106

[56] Zohreh Sharafi, Ian Bertram, Michael Flanagan, and Westley Weimer. 2022. Eyes on Code: A Study on Developers' Code Navigation Strategies. *IEEE Transactions on Software Engineering* 48, 5 (2022), 1692–1704. https://doi.org/10.1109/TSE.2020.3032064

[57] J. Singer, R. Elves, and M.-A. Storey. 2005. NavTracks: supporting navigation in software maintenance. In *21st IEEE International Conference on Software Maintenance (ICSM'05)*. 325–334. https://doi.org/10.1109/ICSM.2005.66

[58] Zéphyrin Soh, Foutse Khomh, Yann-Gaël Guéhéneuc, and Giuliano Antoniol. 2013. Towards understanding how developers spend their effort during maintenance activities. In *2013 20th Working Conference on Reverse Engineering (WCRE)*. 152–161. https://doi.org/10.1109/WCRE.2013.6671290

[59] Susanne Tak and Andy Cockburn. 2013. Enhanced Spatial Stability with Hilbert and Moore Treemaps. *IEEE Transactions on Visualization and Computer Graphics* 19, 1 (2013), 141–148. https://doi.org/10.1109/TVCG.2012.108

[60] Linda Tauscher and Saul Greenberg. 1997. Revisitation patterns in World Wide Web navigation. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems* (Atlanta, Georgia, USA) *(CHI '97)*. Association for Computing Machinery, New York, NY, USA, 399–406. https://doi.org/10.1145/258549.258816

[61] Milo Z Trujillo, Laurent Hébert-Dufresne, and James Bagrow. 2022. The penumbra of open source: projects outside of centralized platforms are longer maintained, more academic and more collaborative. *EPJ Data Science* 11, 1 (2022), 31. https://doi.org/10.1140/epjds/s13688-022-00345-7

10 APPENDIX

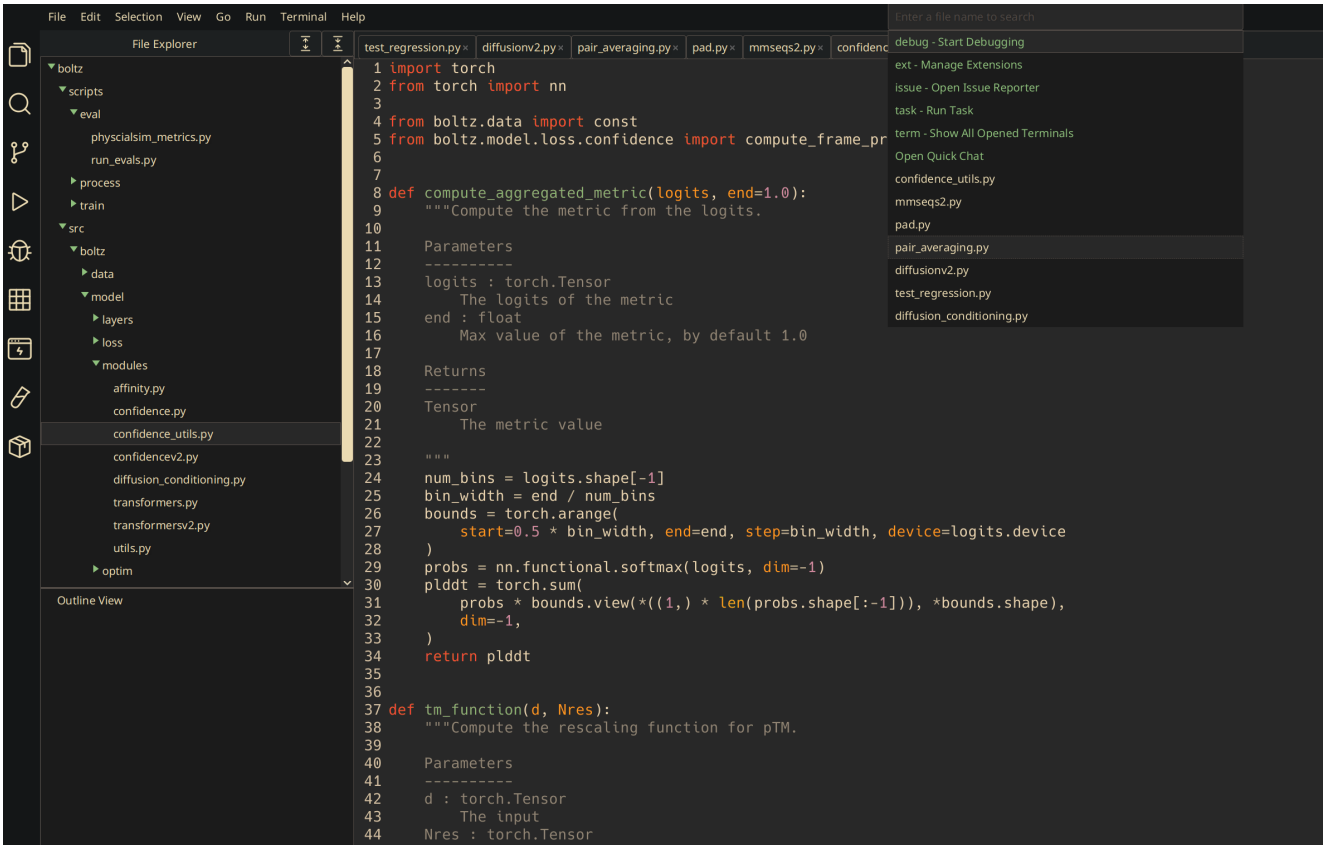

Figure 15: S1: The Standard Interface with a Search Bar, File Explorer, and Tab Browser. (Image has been cropped to show interface elements in more detail)

	B6 & B7			B6 & B8			B6 & B9		
Move Type	**F(df1, df2)**	***p***	η^2	**F(df1, df2)**	***p***	η^2	**F(df1, df2)**	***p***	η^2
Move Down	F(1, 19) = 20.23	**< .001**	0.52	F(1, 19) = 0.72	**.04**	0.04	F(1, 19) = 0.79	.38	0.04
Move Right	F(1, 19) = 36.10	**< .001**	0.66	F(1, 19) = 9.23	**.006**	0.33	F(1, 19) = 1.33	.26	0.07
Move Up	F(1, 19) = 29.00	**< .001**	0.60	F(1, 19) = 3.29	.09	0.15	F(1, 19) = 3.53	.08	0.16
Move Left	F(1, 19) = 15.71	**< .001**	0.45	F(1, 19) = 2.81	.11	0.13	F(1, 19) = 1.94	.18	0.09
No Move	F(1, 19) = 0.01	.92	0.00	F(1, 19) = 0.36	.55	0.02	F(1, 19) = 0.03	.87	0.00

Table 5: S2: Factorial Analyses of Completion Time Before and After the Target Moved.