11-30-2015

An Empirical Evaluation Of Predictive Models Of Programmer Navigation

Alka Rani Singh

AN EMPIRICAL EVALUATION OF PREDICTIVE MODELS OF PROGRAMMER NAVIGATION

by

Alka R. Singh

A Thesis

Submitted in Partial Fulfillment of the

Requirements for the Degree of

Master of Science

Major: Computer Science

The University of Memphis

December 2015

ABSTRACT

Singh, Alka R. M.S. The University of Memphis. December 2015. An Empirical Evaluation of Predictive Models of Programmer Navigation. Major Professor: Dr. Scott D. Fleming

The process of software development consists of many activities, such as writing code, debugging, and navigating through code. Navigating through the code to understand or seek information for developing new code is a very time consuming and tedious task. Many tools are developed based on predictive models to help programmers in navigation. These models predict the fragments of code which might be of developer's interest. There have been studies for comparing these models to determine their predictive accuracy. However, the models are often based on crude approximations of where a developer's attention is. For example, prior work has both where the developer's cursor location as well as what is on the center of the screen to approximate where he/she is looking. To address this concern, we conducted an empirical evaluation of these approximations to see how well they agree with a human evaluator's perception of where the developer's attention is. We conducted a replication study on 10 participants and manually coded their navigation pattern. The goals of the study was to evaluate the generalizability of prior work as well as to evaluate the prior operationalizations of navigation. The key findings of this study are: (a) The operationalization based on where the programmer clicks agreed most closely with human evaluator's assessment and, (b) prior navigation results did not generalize well likely due to small sample size and particulars of the task content.

Contents

Chapter 1

Introduction

One of the major challenges for developers, laboratories and software vendors in today's world is enhancing efficiency of software development in terms of time. The IT industry requires that software products not only satisfy user requirements, but also be quickly produced and maintained to keep pace with an increasing demand for quality, functionality and cost-effectiveness. A delay in development can hurt the overall value of a software product, as the value of the product depends not only on its quality, but also on its timely delivery. From a business perspective, software product delays can hinder competitive advantage and reduce the opportunity to generate revenues. Unfortunately, a 1995 survey of over 8000 projects found that only 16% of the projects were finished on schedule and within the estimated budget [1]. The remaining 84% were incomplete or delayed, and the projects completed late were, on average, nearly twice their originally estimated cost. A more recent survey of projects from 2005 and 2007 found similar issues: only approximately 50% of the surveyed projects were able to meet schedule targets [2]. Thus, reducing the time needed for software development could have a considerable impact on overall project success.

A considerable amount of the time spent on development happens during software evolution. The term *software evolution* is used in software engineering to refer to the fact that the software is constantly changing both during initial development as well as during maintenance. Software maintenance is an inevitable activity for successful, long-lived software because such software periodically demands improvements and enhancements for many reasons, such as bug fixes, software upgrades and feature requests. Among all the software development activities, the most time-consuming phase is software maintenance. Studies have shown that 50% – 80% of software costs are in maintenance [3, 4, 5, 6]. Thus, reducing the cost of software evolution can considerably reduce the overall cost.

A key cost during software evolution tasks is in navigating source code. During development or maintenance, the developers are consistently faced with the need to gain an understanding of the code. This understanding of code is important in identifying code dependencies and determining *how* and *where* features were implemented [7]. A study by Corbi et al. shows that 40% of the total software maintenance time goes into understanding and building a mental model of the software program [8]. In order to develop an understanding of the code, programmers spend a substantial amount of time navigating through the code. Navigating code to develop understanding for maintenance purposes is a complex and time-consuming process [9]. For example, one study showed that developers spent 35% of their total maintenance time on the mechanics of navigation alone [9].

Many promising tools have been proposed to make navigation more efficient, and these tools are often implicitly based on predictive models of programmer navigation. That is, each tool uses a model to predict where the programmer wants to go, and based on the prediction tries to get them there quicker. Mostly, the tools are based on a single factor, such as the recency of a visited code fragment. On the other hand, some tools make predictions based on other factors such as hierarchical relationship among classes [10], lexical similarities among methods [11] or structural similarities within the code [12]. Other tools mine logs of developer activity to make predictions about navigation [13, 14, 15, 16, 17, 18]. Even though preliminary evidence has shown that these tools help developers in efficient navigation, most of the tools have not been validated for a wide variety of development contexts and have not received widespread adoption in practice.

The success of these tools is closely tied to the predictive accuracy of their associated models; however, researchers have only recently begun to systematically compare the accuracy of different models. One early study by Parnin and Görg evaluated a set of models and techniques for suggesting relevant context for next navigation to facilitate exploration of source code [14]. They evaluated four predictive models of navigation and they are based on page caching algorithms. In particular, each of these are based on how recently or frequently a given method is visited. Piorkowski et al. expanded the models and included some other models such as adjacency, forward call depth, undirected call depth and source topology [19]. They compared head-to-head a broad range of models to assess their predictive accuracy based on multiple factors under two different operationalizations of navigation (click-based and view-based). The study reported that recency has the best accuracy in click-based operationalization and adjacency has more accuracy over other models in view-based operationalization.

Although these studies have begun to shed light on the relative effectiveness of various models, open questions remain. One such question is the extent to which these prior results generalize. In particular, the most recent study, reported by Piorkowski et al. [20], had only one participant and that participant worked on only two debugging tasks. However, prior studies have shown significant differences in productivity among individuals. For example, a study by Sackman et al. found substantial variations in individual programming productivity [21]. Another study observed that because some people do not make tangible contribution, the data understates the actual variation in productivity [22].

Another open question is how best to operationalize navigation. The most widely used operationalization of navigation in studies have been *click-based* operationalization of navigation. This click-based operationalization considers a navigation as the change in the cursor position from one fragment of code to another fragment. Along with click-based operationalization, Piorkowski et al. introduced and evaluated another operationalization of navigation called *view-based* [19]. The view-based operationalization considers each navigation as being to the fragment of code which is in middle of the screen. These two different operationalizations of navigation yielded very different results, both in terms of what navigations were counted and of the predictive accuracy produced by the models. The goal of this thesis is to address these open issues of generalizability and navigation operationalization. In particular, we seek to answer the following research questions:

- *RQ1: How well do Piorkowski et al. results generalize with our replication-study results?*
- *RQ2: How well do the operationalizations of navigation agree with human evaluator's perception?*

To answer these research questions, we conducted a qualitative empirical study of programmer's navigation behavior. To address RQ1, we conducted the study on 10 participants to evaluate generalizability of Piorkowski et al.'s lone-participant results. Furthermore, Piorkowski et al.'s participant worked on two debugging tasks, whereas, our participants worked on a variety of evolutionary tasks chosen by them in context of their own project. To address RQ2, we compared the click-based and view-based operationalization of navigation studies by Piorkowski et al. with a new opeartionalization based on a human analyst's assessment of which code fragments a programmer places his/her attention on during tasks. For both RQs, we looked for differences in programmer's navigation profile (e.g., number of methods visited per minute, the revisit pattern to already visited methods and number of different methods visited) and in the accuracy of various predictive models from the literature.

The remainder of this thesis is organized as follows: Chapter 2 discusses the relevant literature in the field. Chapter 3 describes the study method adopted for our study. Chapter 4 reviews the evaluation methodology used in the study. Chapter 5 and 6 report our RQ1 and RQ2 results, respectively. Finally, chapter 7 discusses the results of our research questions in further detail and chapter 8 concludes with a discussion of possible future work.

Chapter 2

Background

The two principle research questions of this thesis center around two concepts from the literature: (1) Operationalizations of navigation and (2) predictive models of navigation. The operationalizations of navigation provide different approximations for recording where a programmer places his/her attention (often trading off on certain strengths/weaknesses, such as automatically vs. ability to detect certain types of navigation). Based on a sequence of recorded navigations, the predictive models each aim to forecast where the programmer will navigate to next using a variety of methods.

2.1 Operationalizations of Navigation

In addressing our research questions, we applied three operationalizations of programmer navigation. In particular, these operationalizations each provide a method for approximating the sequence of code locations (Java methods in our case), where a programmer puts his/her attention. Two of the operationalizations, *click-based* and *view-based*, were previously described in the literature [19]. For this work, we also used a third operationalization, *human-assessment*, against which to compare the first two operationalizations (see RQ2).

2.1.1 Click-Based Operationalization

A click-based navigation is operationalized as the change in the cursor position from one Java method to another. When a programmer clicks in a method other than the current method where the cursor is, it is counted as a click-based navigation. That is, if the current position of a programmer's cursor is in some method *A* and then the programmer clicks in another method *B*, it is counted as a click-based navigation. But if the current position of the programmer's cursor is in method *A* and the programmer clicks somewhere else inside the body of the method *A*, it is not counted as a click-based navigation. For example, in Fig. 1, the current method under the click-based operationalization is `getBalanceNoSign` in which the text cursor is currently present. If the programmer clicks somewhere else inside the `getBalanceNoSign` method, it will not be counted as a click-based navigation. However, if programmer clicks in any other method, for example, in `getLastName`, the click-based operationalization records a navigation to that method.

In particular, the following actions were counted as click-based navigations:

1. clicking into a method other than the current one,
2. clicking on a tab to make the contents of another file visible, and
3. opening a file, for example, by clicking on it in the package explorer.

In the case of a newly opened file, the first method in the opened file is considered to be the next navigation. Although, the click-based operationalization has been widely used (e.g., in [14, 20, 19]) and is automatable, it has a few disadvantages. It will fail to record navigations if a programmer scrolls through a file, but does not actually click in any of the methods that scroll by. Also, just because a programmer clicked on a method does not necessarily mean that programmer has his/her attention on that method.

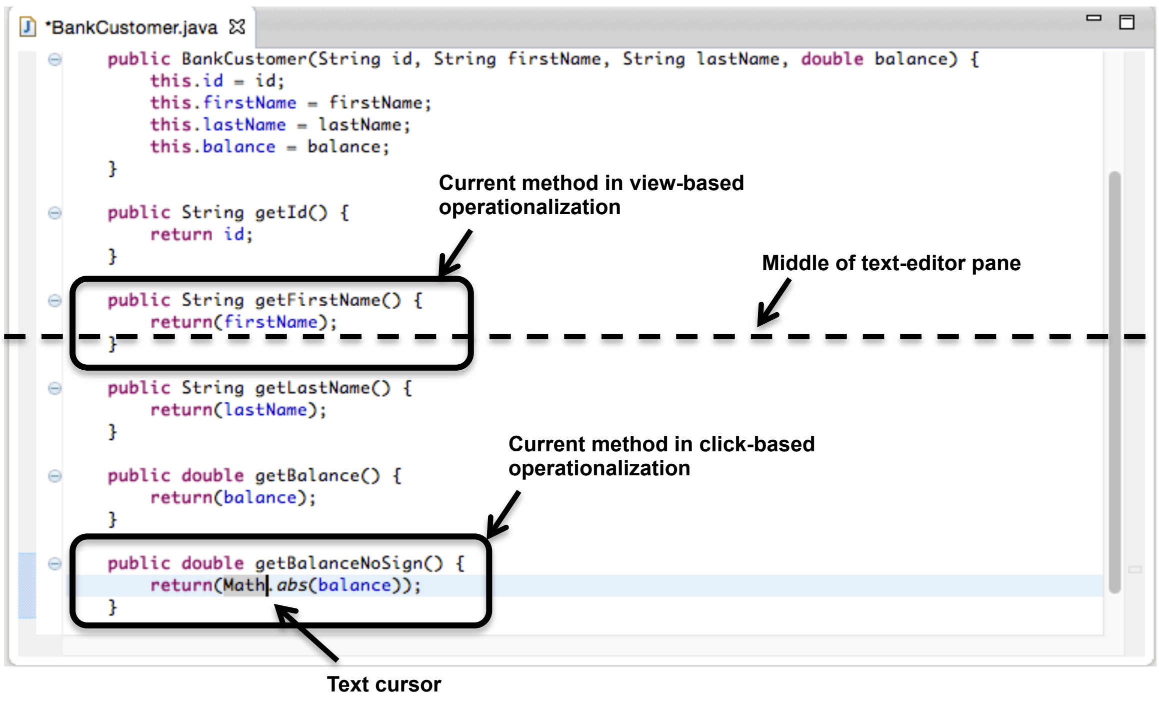
2.1.2 View-Based Operationalization

Unlike the click-based operationalization of navigation, the view-based operationalization does not take the position of the text cursor into account to operationalize a navigation; instead, it defines the current method as the one in the middle of the text-editor pane. For example, in Fig. 1, the current method (view-based) is *getFirstName*, which is present in the middle of the text-editor pane. Furthermore, we adhered to the following two additional rules to record view-based navigations. If when switching or opening editor tabs, a method *A*'s definition becomes completely visible in the text editor and it is present above the middle of the screen while method *B* is in the middle of the screen, then navigations are recorded in the order of first a navigation to method *A* and then to method *B*. Also, if a programmer scrolls through a file, navigations to each method are recorded in the sequence they come to the middle of the screen. The disadvantage of view-based operationalization is that it is not automated and hence it is difficult to record navigations, which has to be done manually. Also, if the programmer quickly scrolls through a file, it records all the methods which passes through the middle of the screen as navigations even if the programmer did not pay attention to them.

2.1.3 Human-Assessment-Based Operationalization

The human-assessment-based operationalization leverages the opinion of a human observer. It stands to reason that this operationalization will be more accurate than the click-based and view-based operationalizations because a human observer is able to take more information into account in deciding where the programmer's attention is. For example, a human-analyst can recognize navigations to methods based on:

- the programmer talking about the methods,
- the programmer creating/editing the methods, and/or
- the programmer copying/selecting/highlighting the contents of the methods.


*BankCustomer.java
Current method in view-based operationalization
Middle of text-editor pane
Current method in click-based operationalization
Text cursor

```java
public BankCustomer(String id, String firstName, String lastName, double balance) {
    this.id = id;
    this.firstName = firstName;
    this.lastName = lastName;
    this.balance = balance;
}

public String getId() {
    return id;
}

public String getFirstName() {
    return(firstName);
}

public String getLastName() {
    return(lastName);
}

public double getBalance() {
    return(balance);
}

public double getBalanceNoSign() {
    return(Math.abs(balance));
}
```

Figure 1: Example of navigations in click-based versus view-based operationalizations. The editor depicted is the Eclipse Java editor(the same kind used by our study participants).

In a few special situations, the human analyst ignored certain actions that might have otherwise been considered navigations:

- if the programmer visited a file just to run it, because the programmer was using an IDE feature to run the program and not to view the content of the file.
- if the programmer clicked on a tab by mistake and did not look at the content of the file.
- if the programmer closed a tab, causing another file to be automatically opened, but the programmer did not look at the content of that file.

We used standard inter-rater reliability methods [23] to ensure our human-assessment-based navigations would be consistent across different observers. Following the method, two researchers analyze 20% of the total data independently and then check the level of agreement. If their results agree 80% or more using the Jaccard index, they can divide the remainder of the data and analyze it separately. In our study, two analysts independently analyzed the same 4 hours of video data (which was 20% of our total video data), and their level of agreement was more than 86%. Because this exceeded the 80% threshold, they divided the remaining 16 hours of video to be separately coded.

2.2 Predictive Models of Programmer Navigation

To address our research questions, we applied the same predictive models of navigation used by Piorkowski et al. [19]. The models assessed in Piorkowski study can be classified in three major categories: working set approximation models, structural similarity models and lexical similarity models. We have included all the models assessed in the prior work except the lexical similarity model *Bug Report Similarity* due to the lack of bug reports in the projects on which the participants were working on.

Following from Piorkowski et al. [19], each of the models takes as input a sequence of method-to-method navigations from a programmer's programming session. Such a navigation history is represented by H, which is a sequence of method-to-method navigations $(m_1, m_2, \ldots, m_n)$ such that for every m_i in H, $m_i \neq m_{i+1}$. If the navigation history H_j for a programming session is given up to a particular point where $H_j = (m_1, m_2, \ldots, m_j)$, each model tries to predict the next method m_{j+1} which is most likely to be visited. At this particular point, the developer has opened multiple source files and the set of methods now known to the programmer is M_j. M_j contains all the methods defined and referenced in the previously opened files irrespective of the fact that they might be closed after opening it. H_j contains a subset of the methods from M_j because the programmer has to open a file before navigating to one of its methods.

In order for a model to predict the next method m_{j+1}, the model ranks the methods from the set of known existing methods M_j-$\{m_j\}$, from least likely to most likely. For a rank of a method to be calculated, the model creates a mapping A_j from each method in M_j-$\{m_j\}$ to an activation value. According to model's prediction, the methods with higher activation value are more likely to be visited. Then an activation function is used by models to create a ranking function R_j in such a way that R_j is A_j with rank-transformed activation values. The higher the activation value of the method, the lower the rank number the method will get (with lower rank numbers, such as rank 1, corresponds to higher ranks). Remember that according to model's prediction, methods with higher rank are more likely to be visited than methods with lower rank. For example, a method with rank 1 is more likely to be visited next by a programmer than a method with rank 3 according to a model. Also, if there are *n* methods with the same activation value, their ranks would be averaged by *n*. Every model uses a different approach for calculating the activation value for methods.

Some models incorporate a notion of cost of navigation between methods, such that higher activation is assigned to methods that cost less to navigate to from the current method than to the methods that cost more. The models maintain a graph G_j such that every method in M_j corresponds to a different vertex in G_j. For a particular model, all m in $M_j - \{m_j\}$, $A_j(m)$ for that model subtracts from $|M_j|$ the length of the shortest path from m to the current method m_j.

2.2.1 Recency and Working Set Models

The *recency* model ranks more-recently visited methods higher than less-recently visited methods. Formally, for every method m in the set of visited method M_j, if programmer already visited m, the activation function $A_j()$ will assign activation value to the method m such that $A_j(m) =$ the max sequence number for m in the programmer's navigation history H_j; otherwise, if the method m was not visited previously, the activation $A_j(m) = 0$.

The *working Set* model is similar to the *recency* model, but the difference is that only a fixed number of recently visited methods are ranked, while all other methods are ranked zero. Formally,*working Set* assumes a window-size W. If a method *m* is in last W visited methods, the activation $A_j(m) = 1$ otherwise $A_j(m) = 0$.

2.2.2 Frequency Model

The *frequency* model assigns lower ranks to methods visited more frequently than those visited less frequently. Formally, for every method *m* in the list of visited method M_j, the activation $A_j(m) =$ number of occurrences of *m* in the programmer's navigation history H_j.

2.2.3 Within-File Distance Model

The Within-File Distance model assigns higher ranks to methods closer to the current method in the file. The ranking function of this model is based on the adjacency factor—that is, this model assumes that the methods closer to the current method are more likely to be visited next. This model creates a graph G_j, where there are links between method nodes, which are textually adjacent in a file. Formally, for every method *m* in M_j, there is a undirected edge from *m* to the methods adjacent to *m*. The adjacent methods could appear before or after the method *m*.

2.2.4 Forward Call Depth and Undirected Call Depth Models

The Forward Call Depth model ranks methods based on a call graph with unidirectional links—that is, the methods being called from the current method are ranked higher than the other methods. Similar to the Within File Distance model, the Forward-Call Depth model also creates a graph G_j, where there is an edge from each method *m* to every method called by *m*. Formally, in the constructed graph G_j, there is a directed edge from method m_a to m_b, if method m_a calls method m_b.

The Undirected Call Depth model is similar to the Forward Call Depth model, with the only difference being that the links from both directions are considered in ranking. That is, this model ranks both the methods: (1) which are being called from the current method and, (2) methods which call the current method. Formally, in the constructed graph G_j, there is an edge between method M_a to M_b, if method M_a calls method M_b or vice versa.

2.2.5 Source Topology Model

The Source Topology model ranks methods higher which share one or more of several structural relationships with the method where programmer's current position is. The Source Topology model constructs a source topology graph which is similar to the graphs constructed by other models, except that in addition to method nodes, this graph contains nodes for classes, interfaces, variables, packages and projects. If there is a *calls-a*, *has-a* or *within-file adjacency* relationship among these nodes, then there is an edge between these elements. Formally, for every element v in $M_j \bigcup C_j \bigcup V_j \bigcup P_j$ where C_j is the set of classes or interfaces referenced in the files the programmer has opened so far, V_j is set of variables and P_j is the set of packages, which includes every package and variable referenced in a file), there is an vertex that maps to v. The source topology graph has an edge between elements v_a and v_b if v_a calls v_b, if v_a has v_b or if v_a and v_a are adjacent methods in a file.

Chapter 3

Study Method

To address our two research questions, we conducted a laboratory study of developers engaged in software evolution tasks. Our data analysis focused on a panel of descriptive statistics regarding each programmer's navigation behavior (e.g., rate of navigation and revisits) and how accurately the various models from Chapter 2.2 predicted those navigations. To address the question of generalizability (RQ1), we compared our analysis results with those reported by Piorkowski et al. [19] to see how well Piorkowski et al.'s results generalize. To address the accuracy of prior operationalizations of navigation (RQ2), we compared each of the prior operationalizations of navigation (i.e., click-based and view-based) with our human-assessment-based operationalization (as defined in Chapter 2.1) to see which was more consistent with the perception of a human.

3.1 Participants

Our participants consisted of 10 graduate students enrolled in a graduate-level software engineering course at the University of Memphis. Table 1 lists their background information. Each participant had a unique identifier of the form $P\langle$ *identifier number*$\rangle$ (e.g., P1, P2, etc). Participants had an average of 6.75 years of programming experience (standard deviation = 2.51). Seven of the ten participants also had experience programming professionally (mean = 2.93 years, standard deviation = 1.54).

Table 1: PARTICIPANT BACKGROUND INFORMATION

ID	Sex	Age	Major	Years of Programming Experience	
				Total	As Professional
P1	M	30s	CS	12	5
P2	M	20s	CS	5	0
P3	F	20s	CS	6	0
P4	M	20s	CS	6	1
P5	F	20s	CS	6	2.5
P6	M	20s	CS	4	0
P7	M	20s	CS	8	4
P8	M	20s	CS	5.5	1
P9	M	20s	CS	5	4
P10	M	20s	EE	10	3

3.2 Tasks and Environment

For this study, participants worked on their projects from the software engineering course. Their projects were Java EE based web applications for helping students and advisors track student progress through the degree program with features such as registration of courses and grading by faculty. The participants were members of teams in the course, and the team worked collaboratively on the project.

For each study session, the participant worked on tasks of his/her choice. The tasks observed could be loosely categorized as (1) implementing different features for their project or (2) debugging if there was a bug present in the source code. However, the exact features and bugs worked on varied widely. Most of the participants worked on four or more tasks during their sessions.

To perform their tasks, participants worked on a workstation with a 24-inch monitor. Their programming environment consisted of the Eclipse integrated development environment (IDE), the MySQL Workbench (relational database management system) and a web browser with full internet access. They were also allowed to use any software pre-installed on a Windows PC. For example, two participants used Notepad to make notes and Microsoft Paint to sketch their ideas while working on project.

The development environment was outfitted with several technologies for our collection of data. Camtasia[1] recorded screen-capture video throughout each participant's session. Participants wore a headset with microphone to collect audio of their utterances. An HD webcam recorded video of the participant.

3.3 Procedure

In the beginning of the session, participants were familiarized with the intent of the study, and read and signed an informed-consent form. Participants were then asked to fill out a questionnaire about their backgrounds. In that questionnaire, they were asked questions about their age, gender, major, mother tongue, programming experience and professional experience (if any).

To better understand where participants placed their attention, we employed the *think-aloud method* [24]. The think-aloud method is a well-validated method by which participants externalize their inner dialogue by continuously uttering their thoughts as they perform a task. It is a well-established research method in Psychology and Human-Computer Interaction, which is used to gain insight into the goals and intentions of the person thinking aloud. Following the method, we asked our participants to think-aloud while working on their task. If a participant stopped talking for three minutes, an attending researcher asked the participant to "Please keep talking". To familiarize them with the concept of thinking-aloud, they were shown a demonstration of thinking aloud while

[1]http://discover.techsmith.com/try-camtasia/

answering a question. After the demonstration, participants were asked to think-aloud while answering a similar question in order to practice thinking aloud. Following the exercise, they also practiced thinking-aloud using a computer. After the initial training and practicing, the participant worked on his/her project tasks for 120 minutes.

3.4 Analysis Method

We analyzed the videos as per the methods in Chapter 2.1 to produce analysis data for click-based, view-based, and human-assessment-based navigations. Then, these navigation data was used as an input to the predictive models from Chapter 2.2. Finally, we assess the predictive accuracy of the models.

Predictive accuracy of a model is defined by its ability to accurately predict a programmer's next navigation. A model is said to get a *hit* if the programmer's actual navigation is among the model's top-W predictions, where W is the window size. For example, if a model predicts five methods, m_1, m_2, m_3, m_4 and m_5, as its top-five predictions, where m_1 is the top most prediction with rank 1 and m_5 is ranked 5 as its lowest prediction, and if the actual method which the programmer navigated to is m_4, then it is considered as a hit for $W = 4$ but is a miss for $W = 3$. Some of the model's rankings were of partial ordering. To address the issue of ties, we refined the definition of a hit as follows: if R is the rank of a method, T is number of ties and A is the actual navigation of programmer, then a *hit* is considered when $\lfloor T/2 \rfloor + \lfloor R(m) \rfloor < W$.

Chapter 4

RQ1 Results: Generalizability of Piorkowski Study

To assess the generalizability of Piorkowski et al.'s results [19] on programmers' navigation behaviors and models thereof (RQ1), we directly compared those results with our replication-study results. Given the well-known individual differences among people in general and programmers in particular [25], there was reason for concern about the representativeness of Piorkowski et al.'s single participant. Our comparison focused on two aspects of Piorkowski et al.'s results. The first aspect was programmer *navigation profile*—that is, descriptive qualities of navigation behavior, such as number of different places (i.e., Java methods) visited and rate of navigation. The second aspect was the predictive accuracy of a slate of models (detailed in Chapter 2.2) of programmer navigation—that is, models that aim to predict where a developer will navigate next at a given point in time. In the remainder of this chapter, we report our comparison results for each of these aspects in turn.

For each aspect we compared, we considered the two different operationalizations of programmer navigation that Piorkowski used: click-based and view-based. Recall from Chapter 2.1, the click-based operationalization records a navigation each time the programmer clicks in a Java method in the code editor, whereas, the view-based operationalization records a navigation each time a method passes through the center of the programmer's code editor. Below, we report the results for each of these operationalizations, for each aspect of comparison (i.e., participant navigation profile and predictive-model hit rate).

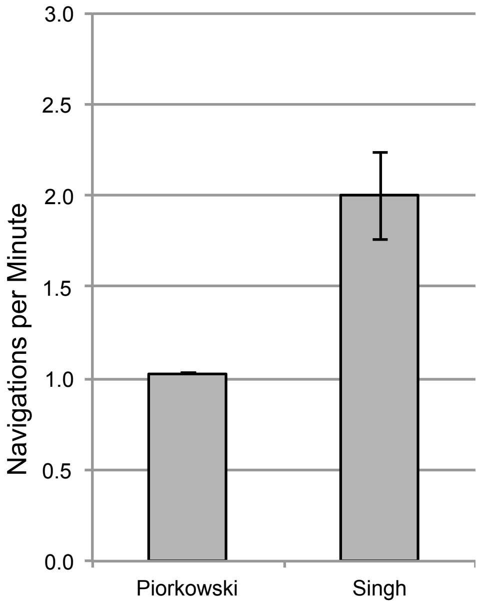
4.1 Navigation-Profile Comparison

We evaluated, the generalizability of the navigation behavior exhibited by the participant in the Piorkowski study by comparing our results for both click-based and view-based operationalizations. We report our results for each operationalization in turn.

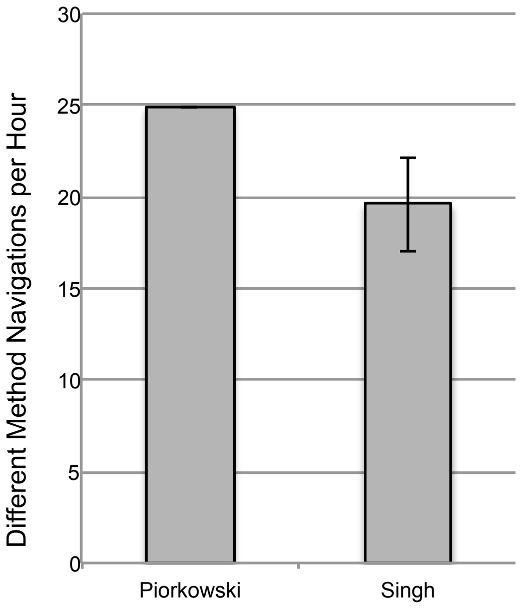
Fig. 2 shows our navigation-profile results alongside those from the Piorkowski study for click-based operationalization. Fig. 2a–c each focuses on a different aspect of navigation behavior: rate of navigation, number of different methods visited and the percentage of revisits, respectively. Fig. 2d summarizes the relative differences in our data and the Piorkowski data for each of the three aspects. The relative difference between the two quantities x and y is expressed as a percentage, indicating how much the two quantities differed. For example, a 50% relative difference indicates that the value of either x or y is double than that of the other. Formally, we compute relative difference as follows:

$$\text{Relative difference}(x,y) = \frac{|x-y|}{\max(x,y)}$$

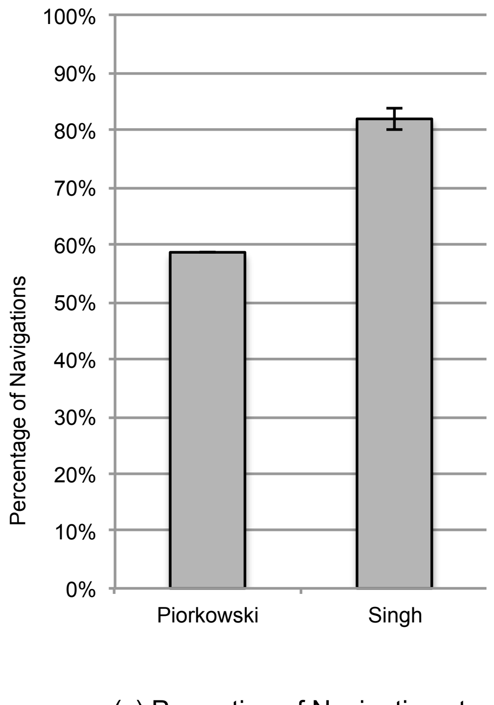
For the click-based operationalization of navigations, Piorkowski et al.'s navigation-profile results were substantially different from those of our participants'. For example, the relative difference in the rates of navigation was considerable at 49%—our participants navigated twice as much as the Piorkowski participant; Furthermore, the average relative difference across all aspects of the navigation profile was 33%—i.e., on average the result from one study was 1.5 times the result from the other.



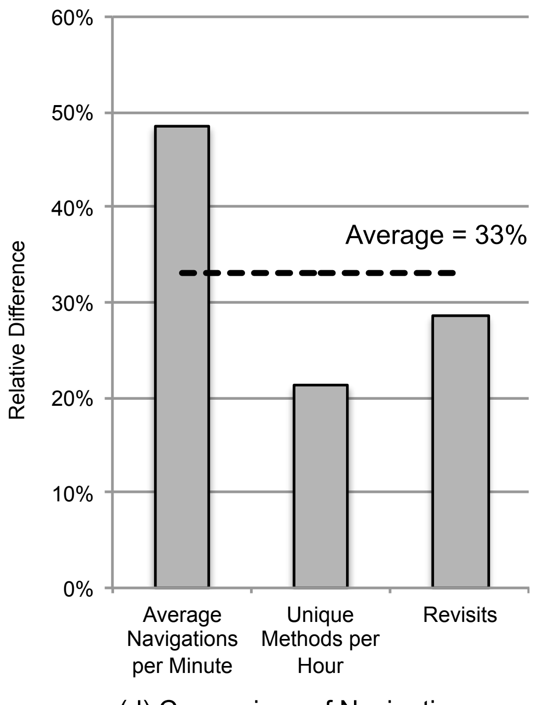
(a) Rate of Navigations



(b) Number of Different Methods Visited



(c) Proportion of Navigations to Previously Visited Methods



(d) Comparison of Navigation Profiles with the Piorkowski Study

Figure 2: Navigation profiles for click-based operationalization. Charts a–c compare the Piorkowski results with our results for three aspects of navigation behavior. The values of our study are averaged among 10 participants hence there is a standard error bar. Piorkowski et al. had only one participant; thus, there is no standard error bar for their results. Chart d shows the relative differences across all aspects of the navigation profile.

Fig. 3 presents the navigation-profile results for the view-based operationalization. The organization of this figure is similar to that of Fig. 2. Similar to the click-based results, Piorkowski et al.'s participant navigation profile results for the view-based operationalization were also substantially different from that of our participants. The largest difference was in the number of different methods visited, which had 52% relative difference in both studies, and the average relative difference across all aspects of navigation profile was 30%.

4.2 Model Accuracy Comparison

In addition to navigation profiles, we also evaluated the generalizability of the results that Piorkowski et al. reported for the predictive accuracy of the 7 models of navigation from Chapter 2.2. Similar to the above section, we looked at both the click-based operationalization and the view-based operationalization. To compute the hit rates for the models, we used a window size W=10 (same as Piorkowski et al.). That is, if a developer's actual navigation was to a method in the top-ten predictions made by a model, it was considered a *hit*; otherwise a *miss*. Fig. 4a juxtaposes the model-accuracy results for the Piorkowski study and our study based on the click-based operationalization. Fig. 4b shows the relative difference between ours and the Piorkowski's hit rates for each model.

As the Fig. 4 shows, for the click-based operationalization of navigation, Piorkowski et al.'s models predictive accuracy results were considerably different from ours. Our hit rates for models, such as frequency and undirected call depth, were considerably different from the Piorkowski study at 49% and 63%, respectively. The Piorkowski study generalized well for a couple factors: within-file distance and forward call depth both had a relative difference less than 15%. However, the mean relative difference in model results across all models was substantial at 31%.

Turning to the view-based operationalization, as the Fig. 5 shows, there was even less agreement between our model-accuracy results and Piorkowski et al.'s than for the

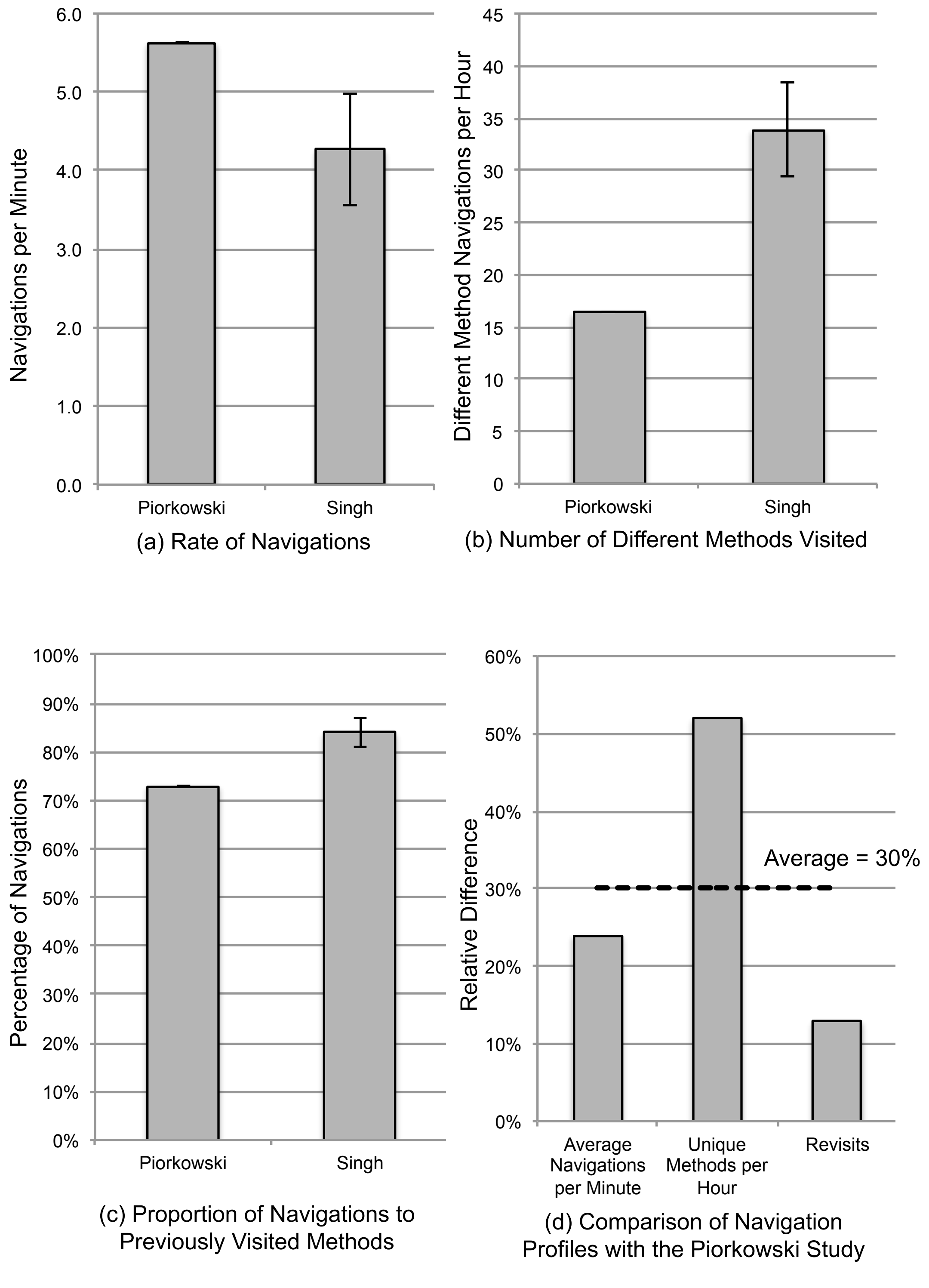
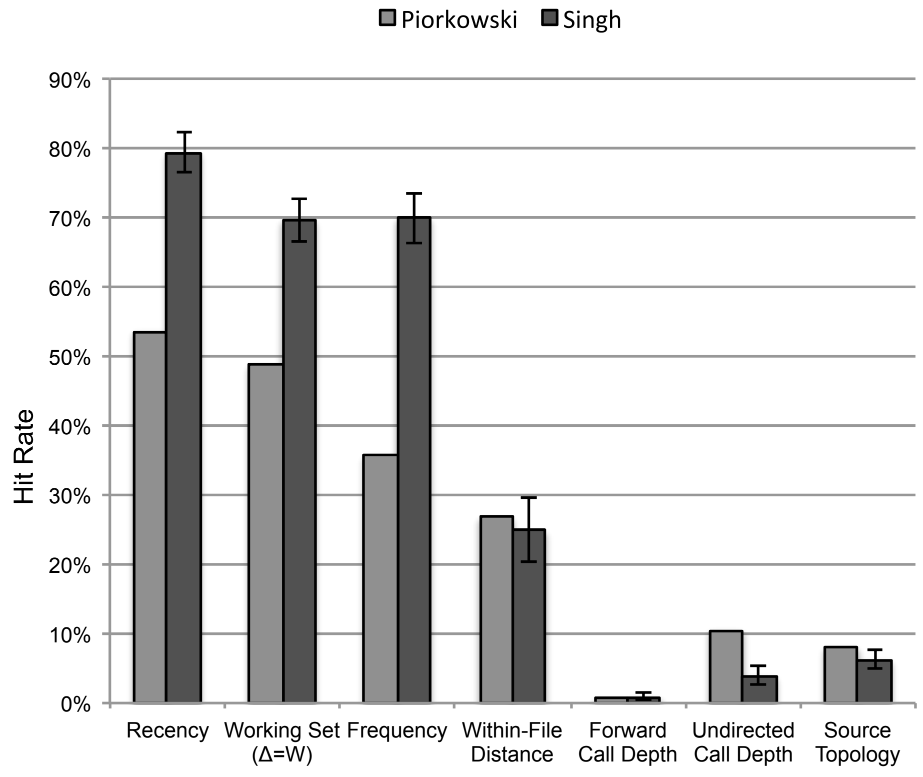
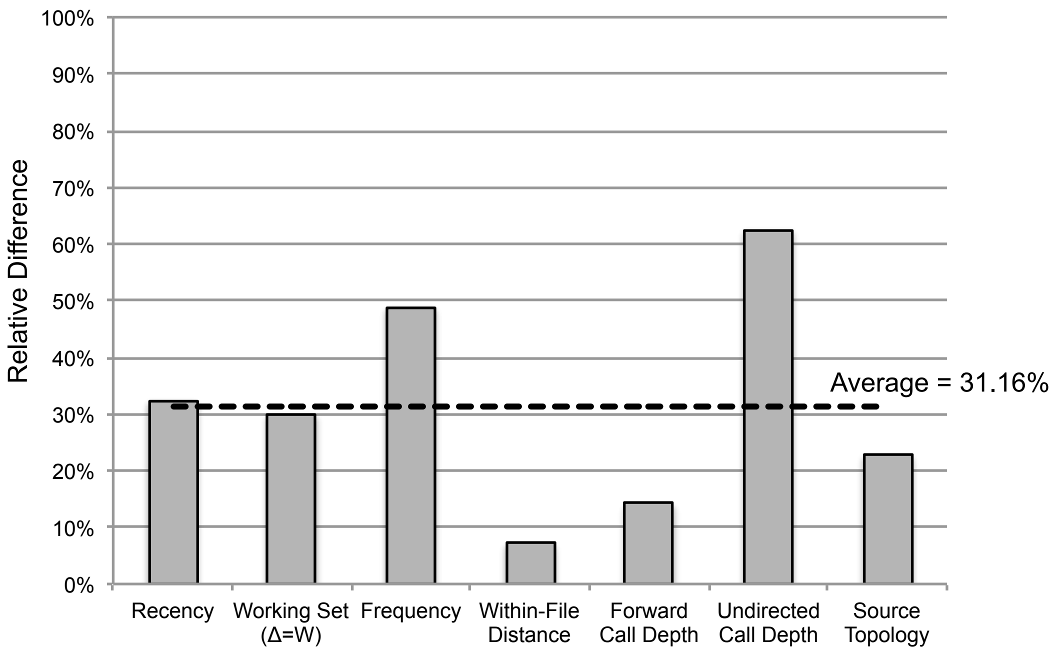


Figure 3: Navigation profiles for view-based operationalization. Charts a–c compare the Piorkowski results with our results for three aspects of navigation behavior. The values of our study are averaged among 10 participants hence there is a standard error bar. Piorkowski et al. had only one participant; thus, there is no standard error bar for their results. Chart d shows the relative differences across all aspects of the navigation profile.



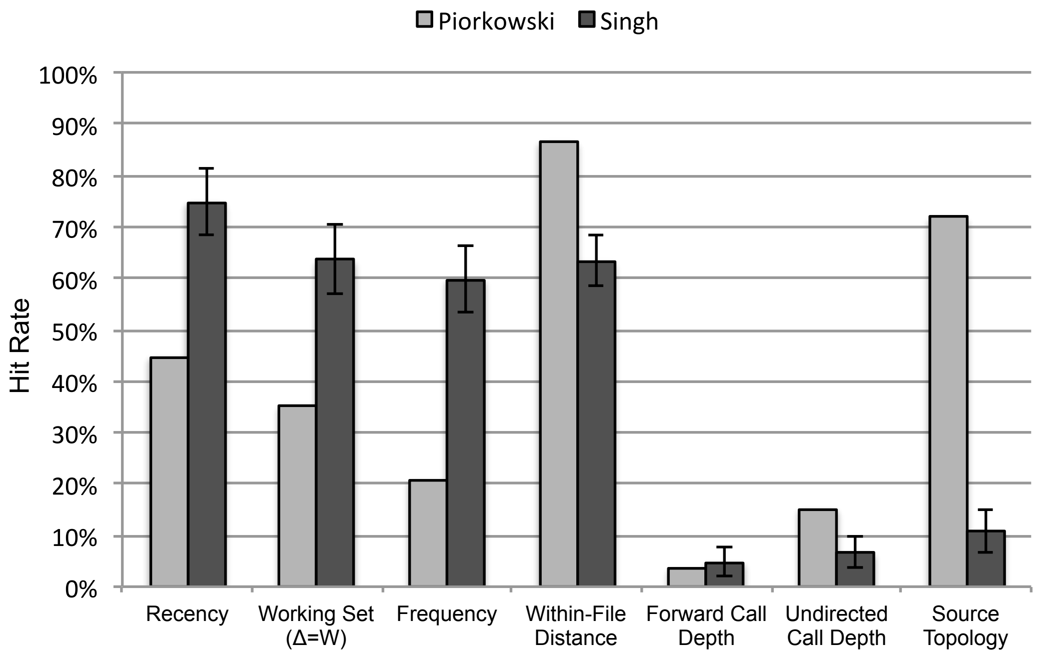
(a) Predictive Accuracy of Model in Click-Based Operationalization



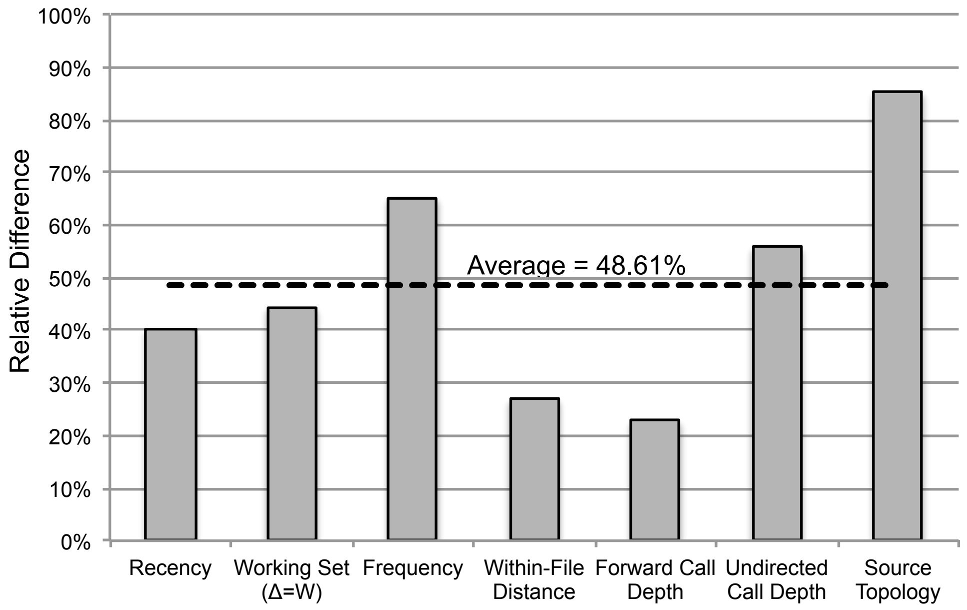
(b) Relative Difference in Hit Rate in Click-Based Operationalization (W=10)

Figure 4: Model's Predictive Accuracy in Click-based operationalization (W=10).

click-based results. The hit rates of most of the models in view-based operationalization were considerably different when compared to the Piorkowski study. The models that differed most were frequency, source topology and undirected call depth, with over a 50% relative difference for each. The mean relative difference between the predictive accuracy of both the studies in view-based operationalization was also very high at 49%.



(a) Predictive Accuracy of Model in View-Based Operationalization



(b) Relative Difference in Hit Rate in View-Based Operationalization (W=10)

Figure 5: Model's Predictive Accuracy in View-based operationalization (W=10).

Chapter 5

RQ2 Results: Operationalizations of Navigation

To understand how well prior operationalizations of navigation agree with a human evaluator's perception (RQ2), we applied the operationalizations to our replication-study data, and we compared the analysis results with the results of a manual qualitative analysis. In particular, the two prior operationalizations of navigation were click-based (Section 2.1.1) and view-based (Section 2.1.2) and the manual qualitative analysis was based on human-assessment (Section 2.1.3). Similar to our RQ1 analysis, we focused our comparison on different aspects of navigation behavior, such as rate of navigation and number of different methods visited, and the predictive accuracy of the models of navigation from Chapter 2.2. In the remainder of this chapter, we report our comparison results for each of these aspects in turn.

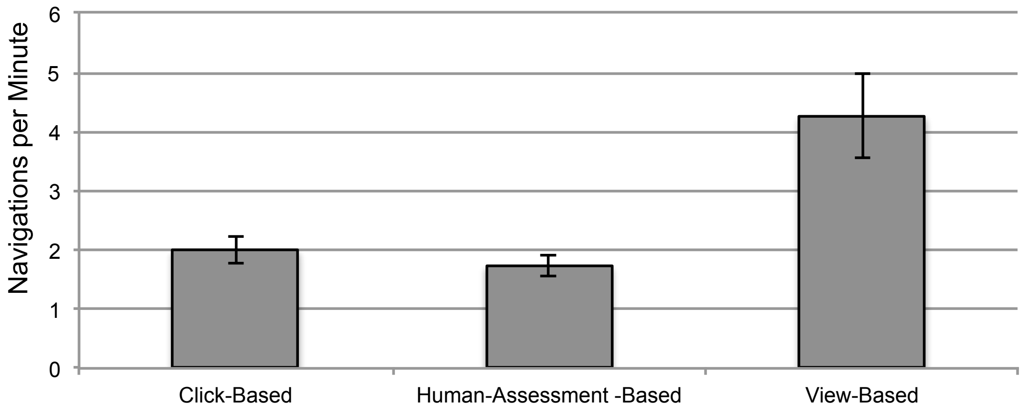
5.1 Navigation-Profile Comparison

We evaluated the accuracy of the prior operationalizations (i.e., click-based and view-based) by comparing the different aspects of a programmer's navigation behavior with respect to each operationalization with that of the human-assessment-based operationalization. For the comparison, we take into consideration the same navigation profile as in Section 4.1, which consisted of rate of navigation, number of different methods visited and number of revisits to already visited methods.

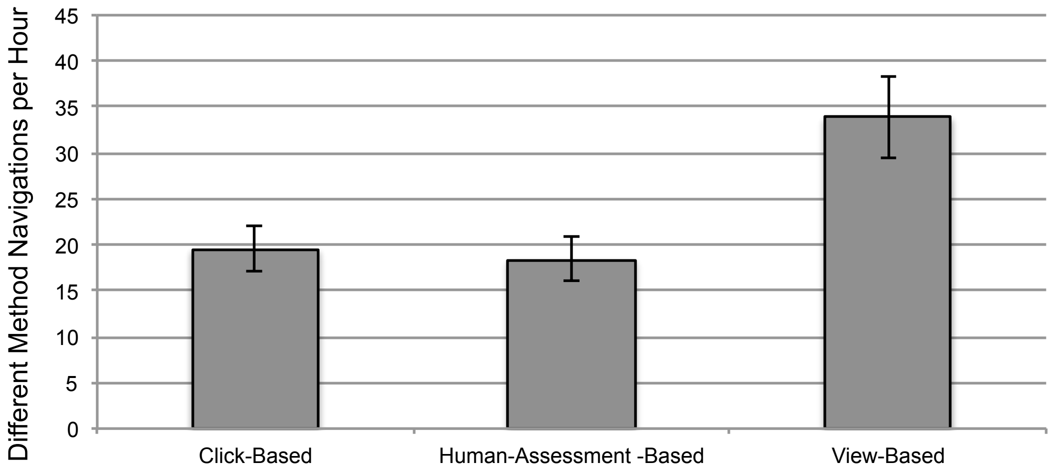
Fig. 6 shows our participant's navigation-profile results for each operationalization (click-based, view-based and human-assessment-based). Each chart focuses on a different aspect of navigation behavior: (a)rate of navigation, (b) number of different methods visited and (c) the percentage of revisits, respectively. For example, Fig. 6a illustrates the comparison of rate of navigation for click-based and view-based versus human-assessment-based operationalization (Human-assessment-based results are provided n the middle of the bar charts to facilitate comparison with the other operationalization results). Fig. 7 summarizes the relative difference across these aspects of the navigation-profile results for each of the prior operationalizations versus the human-assessment-based operationalization.

As the figures make clear, the participant navigation profile for the click-based operationalization was very similar to the participant navigation profile for the human-assessment-based operationalization. Among all factors, only rate of navigation was significantly different among both click-based and human-assessment-based operationalization (Wilcox-Test: $V = 48$, $p = 0.04$).

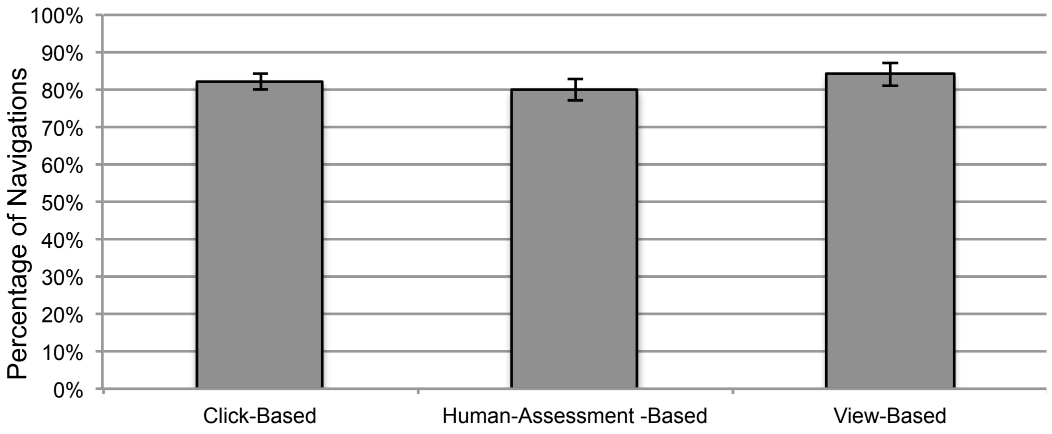
Unlike the click-based operationalization, the participant navigation profile for the view-based operationalization was considerably different than that of the human-assessment-based operationalization. There was a significant difference between rate of navigation in view-based and human-assessment-based operationalization (Wilcox-Test: $V = 54$, $p = .004$). Also, number of different methods visited was significantly different in view-based and human-assessment-based operationalization (Wilcox-Test: $V = 55$, $p = .002$) The relative difference between the participant navigation profiles in click-based and human-assessment-based operationalization was



(a) Rate of Navigations



(b) Number of Different Methods Visited



(c) Proportion of Navigations to Previously Visited Methods

Figure 6: Comparison of navigation profiles in click-based and view-based operationalization with that of in human-assessment-based operationaliztion.Charts a, b and c compare the click-based and view-based results with human-assessment-based results for three aspects of navigation behavior. The values of our study are averaged among 10 participants hence there is a standard error bar.

Table 2: NAVIGATION-PROFILE STATISTICS FOR CLICK-BASED, VIEW-BASED AND HUMAN-ASSESSMENT-BASED OPERATIONALIZATION.

Operationalization of Navigation	Aspect of Navigation Profile	Mean	Std. Dev.	Difference with Human-Assessment-based	V	p
Human-Assessment-Based	Navigation per minute	1.73	0.57	n/a	n/a	n/a
	Different methods visited per hour	18.43	7.38	n/a	n/a	n/a
	Proportion of navigations to previously visited methods	0.80	0.10	n/a	n/a	n/a
Click-Based	Navigation per minute	2.00	0.76	0.27	48	*0.04**
	Different methods visited per hour	19.59	8.13	1.16	29	0.14
	Proportion of navigations to previously visited methods	0.82	0.07	0.02	37	0.09~
View-Based	Navigation per minute	4.28	2.26	2.55	54	*.004**
	Different methods visited per hour	33.94	14.16	15.51	55	*.002**
	Proportion of navigations to previously visited methods	0.84	0.08	0.04	46	.07~

In the last column, * and $\sim$ indicates the significant and marginal differences respectively.

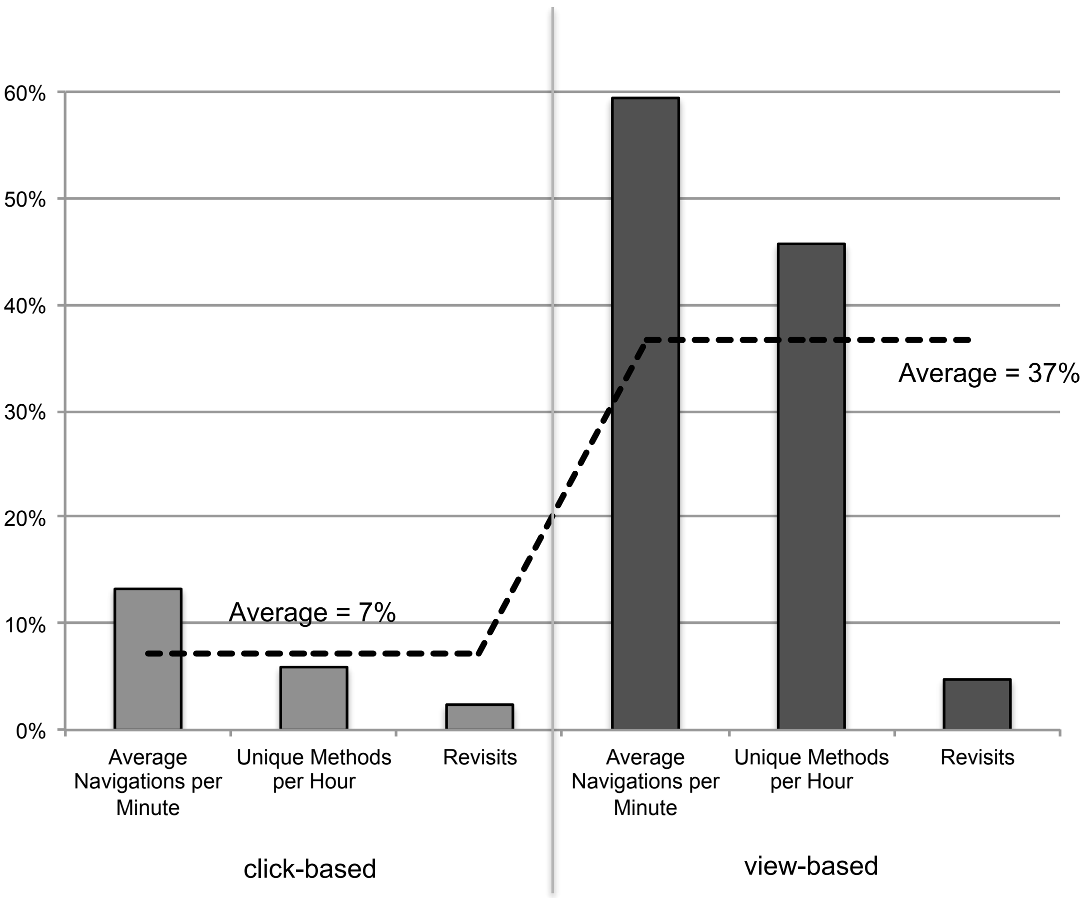


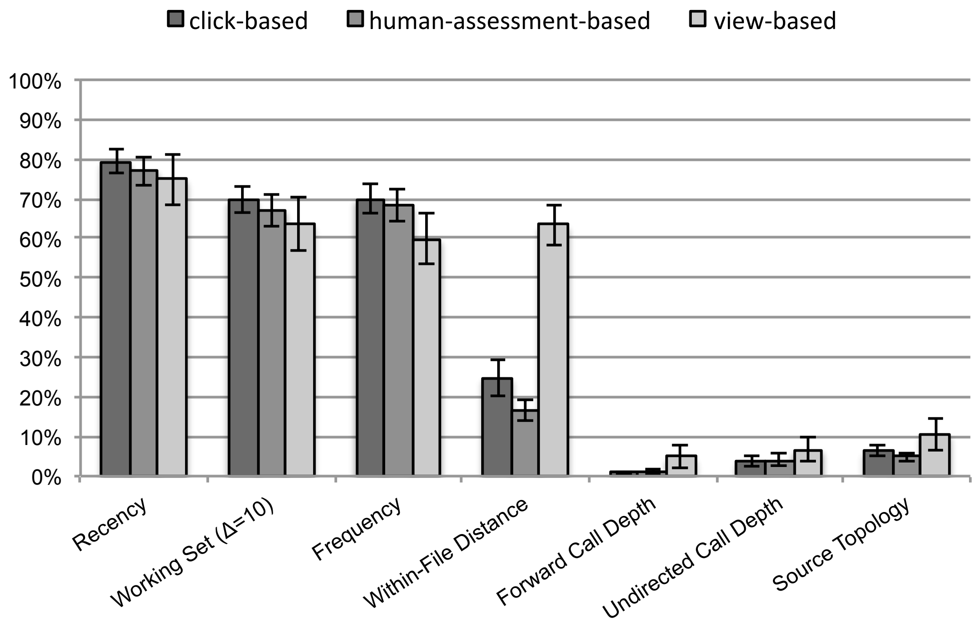
Figure 7: Relative difference in participant navigation profile in human-assessment-based operationalization and participant navigation profile in click-based and view-based operationalization

only 7%, while the relative difference between the same in view-based and human-assessment-based operationalization was 37%.

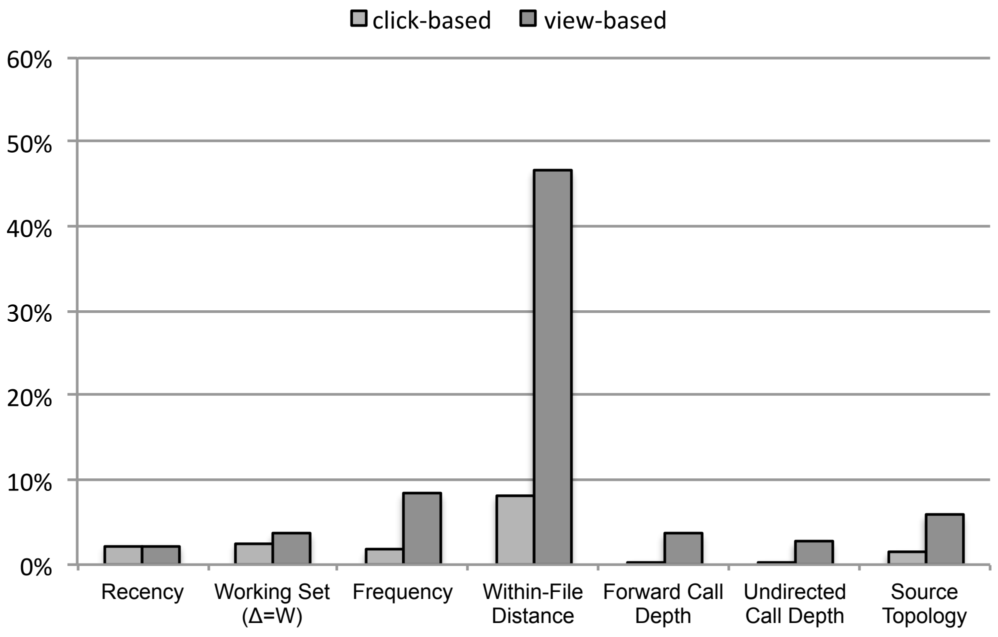
5.2 Model-Accuracy Comparison

To evaluate the extent to which the prior operationalizations produce model-accuracy results that agree with the accuracy results of our human-assessment-based operationalization, we ran the navigation data produced by each operationalization through the seven models of navigation from Chapter 2. Fig. 8a illustrates the predictive-model hit rate for all three operationalizations, while Fig. 8b illustrates the relative difference between the predictive-model hit rate in human-assessment-based operationalization and click-based operationalization versus view-based operationalization respectively. As before, each human-assessment bar is positioned in between the corresponding click-based and view-based bars to facilitate comparison.

The models' predictive accuracy for both click-based and view-based operationalizations were very close to the models' predictive accuracy for human-assessment-based operationalization for most of the models. The model within-file distance has the prominent relative difference (47%) in view-based and human-assessment-based operationalization. The within-file distance model's predictive accuracy for both click-base (Wilcox-Test: $V = 50.5$, $p = .02$) and view-base operationalization (Wilcox-Test: $V = 55$, $p = .002$) is significantly different than that of human-assessment-based operationalization. Also, the frequency model's predictive accuracy for view-based operationalization is significantly different than that of human-assessment-based operationalization (Wilcox-Test: $V = 54$, $p = .004$) The wilcox-test results for comparison of each model's predictive accuracy for click-based and view-based operationalization with that of human-assessment-based operationalization is mentioned in Table 3 and Table 4 respectively. The average relative difference of models' predictive accuracy among all factors in click-based and human-assessment-based operationalization is 2% as compared to 11% relative difference in average predictive accuracy of models between view-based and human-assessment based operationalization.



(a) Predictive Model Hit Rate of All Operationalizations



(b) Relative Difference between Predictive Model Hit Rate of Human-Assessment-Based operationalization and Click-Based and View-Based Operationalization

Figure 8: Comparison of model's predictive accuracy hit rate in click-based and view-based operationalization with human-assessment-based operationalization.

Table 3: MODEL ACCURACY STATISTICS FOR CLICK-BASED AND HUMAN-ASSESSMENT-BASED OPERATIONALIZATION.

	Click-Based		Human-Assessment-Based		Difference in Mean	Wilcox-Test Results	
Model	Mean	SD	Mean	SD	(Absolute Value)	V	p
Recency	79.40%	9%	77%	12%	2.4%	37	0.10~
Working Set	69.71%	10%	67%	13%	2.71%	44	*0.04**
Frequency	69.99%	12%	68%	13%	1.99%	48	0.10~
Within File Distance	24.90%	15%	17%	9%	7.9%	50.5	*0.02**
Forward Call Depth	0.95%	1%	1%	2%	0.05%	5	0.59
Undirected Call Depth	3.96%	5%	4%	5%	0.04%	7	1
Source Topology	6.28%	5%	5%	4%	1.28%	48	*0.04**

In the last column, * and ~ indicates the significant and marginal differences respectively.

Table 4: MODEL ACCURACY STATISTICS FOR VIEW-BASED AND HUMAN-ASSESSMENT-BASED OPERATIONALIZATION.

	View-Based		Human-Assessment-Based		Difference in Mean	Wilcox-Test Results	
Model	Mean	SD	Mean	SD	(Absolute Value)	V	p
Recency	75%	20%	77%	12%	2%	21.5	.57
Working Set	64%	21%	67%	13%	3%	15	.40
Frequency	60%	20%	68%	13%	8%	54	*.004* *
Within File Distance	63%	16%	17%	9%	46%	55	*.002* *
Forward Call Depth	5%	9%	1%	2%	4%	21	*.03* *
Undirected Call Depth	7%	9%	4%	5%	3%	13	.67
Source Topology	11%	13%	5%	4%	6%	54	*.004* *

In the last column, * indicates the significant difference.

Chapter 6

Discussion

6.1 Why Did the Piorkowski Results Differ from Ours (RQ1)?

Piorkowski et al.'s results for both programmer behavior and predictive accuracy of models did not generalize well for both the click-based and the view-based operationalizations. For instance, the average relative difference in the navigation profiles of the Piorkowski participant and our participants was 33% for click-based operationalization and 30% for view-based operationalization. We noted a number of possible reasons for these considerable differences.

One possible reason was that our participants were familiar with the code base they were working on. For example, the rate of navigation for our participants was double the rate of navigation for Piorkowski et al.'s participant for click-base operationalization. In the previous study, the participant spent a considerable amount of time navigating through the code to get familiar with it, which involved frequent scrolling and scanning of files. Such scrolling and scanning do not involve clicking in any method; therefore, far fewer click-based navigations were recorded for the Piorkowski participant. In contrast, our participants were working on a familiar project, and hence, spent more time clicking and changing the code, which resulted in a higher click-based navigation rate. Familiarity with the code may have also led to differences in number of different methods visited. Our participants knew which methods to navigate to for completing the task, which resulted in less exploring and a lower number of different methods visited in click-based operationalization.

Another possible reason for the substantial differences in our data had to do with the types of tasks participants performed. For example, the percentage of navigations which were revisits to already visited methods was higher (more than 50% relative difference) in our study than the Piorkowski study for view-based operationalization. In our study, participants worked on feature addition/enhancement tasks, where they spend more time developing or modifying a larger segment of code and clicking more frequently, whereas in the Piorkowski study, the task was debugging, where the participant spent considerable amount of time scanning the code without clicking in any method with the intention of understanding the code. Because the task in our study involved adding new features, participants were navigating to different methods which have a similar functionality to the current method they were working on. The intent of these navigations was to copy the functionality from another method or to reference another method.

Our participants were copying and pasting the code from methods having similar functionality and hence, they revisited the methods frequently relevant to their task. The Piorkowski participant spend time inspecting a lot of methods for bugs but not finding them did not end in revisits. The frequent navigations to these methods not only increased the percentage of revisits in their total navigation rate for view-based operationalization, but also increased the predictive accuracy of models such as recency, frequency and working set for both click-based and view-based operationalization when compared to the previous study.

A final possible reason for the poor generalizability of the Piorkowski study could be tied to the difference in the type of project. The predictive accuracy of source topology model was higher in the Piorkowski study when compared to our study for both click-based (more than 20% relative difference) and view-based (more than 80% relative difference) operationalization. In our study, participants worked on JSP pages in addition to the plain old Java code, and the way the models work is that they only take Java method's definition into consideration. The models treated JSP pages as 1-method class and ignored the contents of those methods and it resulted in lower predictive accuracy (for both click-based and view-based operationalization) of models such as undirected call depth and source topology which takes methods' definition for predicting the next navigation.

6.2 Which Operationalization of Navigation Was Closest to a Human Evaluator's Perception of Navigation (RQ2)?

RQ2 results report that the click-based operationalization of navigation was very close to a human evaluator's perception of navigation and hence was more accurate in operationalizing navigation than the view-based operationalization of navigation. The average relative difference in navigation profile of participants between click-based and human-assessment-based operationalization was 7% in contrast to the average relative difference between view-based and human-assessment-based operationalization which was 37%. The average relative difference in predictive accuracy of models for click-based and human-assessment-based operationalization was negligible (2%) when compared to the average relative difference in predictive accuracy of models for view-based and human-assessment-based operationalization, which was 11%.

The most significant difference among view-based and human-assessment-based operationalization was in the navigation profile of the participants (Table 2). These differences were likely caused by the way the view-based operationalization has been defined—a navigation is considered to a method which is in the middle of the screen of the text editor (Section 2.1), but the human-assessment-based operationalization only considers a navigation to a method if that method has programmer's attention. During the study session, participants quickly scrolled within files to get to the desired methods without paying attention to methods which passed by the middle of the screen. However, the view-based operationalization recorded navigations to all those methods while human-assessment-based operationalization ignored them. This resulted in a higher rate of navigation and a higher number of different methods visited for the view-based operationalization in comparison to human-assessment-based operationalization.

On the other hand, the click-based operationalization recorded navigations to the methods in which the text cursor is present, which could be either because the programmer clicked in it intentionally or by mistake. Mostly our participants clicked in the methods they wanted to navigate to or the ones on which they were working. The clicks in the methods by mistake resulted in a negligible difference in navigation profiles for click-based and human-assessment-based operationalization.

The most significant difference between the prior models with respect to human assessment was in the predictive accuracy of models of the within-file distance model. The predictive accuracy of within-file distance model for view-based operationalization was much higher (more than 45% relative difference) than that of human-assessment-based operationalization. As mentioned in Section 2.2, the within-file distance model assigns higher ranks to methods closer to the current method in the file. In the view-based operationalization, when participants were scrolling through the file, the operationalization recorded navigations to a stream of methods, each next to the previously visited method in the file, leading inordinately high predictive accuracy for that model.

Chapter 7

Conclusion

In this thesis, we conducted a replication study that was the first to test the generalizability of prior navigation behavior and prediction results and the first to evaluate prior operationalizations of navigation with respect to a human evaluator's perception of navigation. Key findings of our study were the following:

- The click-based operationalization was reasonably good approximation of a human evaluator's perception.
- In contrast, the view-based operationalization was a poor approximation of a human evaluator's perception.
- The Piorkowski et al.'s results did not generalize well in our study both in terms of the participant navigation profile and predictive accuracy hit rate of models.

These results have implications for how to automate navigation detection. Even though human-assessment-based operationalization is the most accurate way to operationalize navigation but it is not automatable but click-based operationalization is automatable. Also, generalizability findings shed light on the effects of tasks, familiarity with the code base, and type of project on navigation behavior.

References

[1] J. Johnson, “Chaos: the dollar drain of it project failures,” *Application Development Trends*, 1995.

[2] K. E. Emam and A. G. Koru, “A replicated survey of it software project failures.” *IEEE Software*, vol. 25, no. 5, pp. 84–90, 2008. [Online]. Available: http://dblp.uni-trier.de/db/journals/software/software25.html#EmamK08

[3] T. A. Standish, “An essay on software reuse,” *IEEE Trans. Softw. Eng.*, vol. 10, no. 5, pp. 494–497, Sep. 1984. [Online]. Available: http://dx.doi.org/10.1109/TSE.1984.5010272

[4] R. D. Banker, S. M. Datar, and C. F. Kemerer, “A model to evaluate variables impacting the productivity of software maintenance projects,” *Manage. Sci.*, vol. 37, no. 1, pp. 1–18, Jan. 1991. [Online]. Available: http://dx.doi.org/10.1287/mnsc.37.1.1

[5] C. Kim and S. Weston, “Software maintainability: Perceptions of edp professionals,” *MIS Q.*, vol. 12, no. 2, pp. 167–185, Jun. 1988. [Online]. Available: http://dx.doi.org/10.2307/248841

[6] J. T. Nosek and P. Palvia, “Software maintenance management: Changes in the last decade,” *Journal of Software Maintenance*, vol. 2, no. 3, pp. 157–174, Sep. 1990. [Online]. Available: http://dx.doi.org/10.1002/smr.4360020303

[7] J. Sillito, G. C. Murphy, and K. De Volder, “Questions programmers ask during software evolution tasks,” in *Proceedings of the 14th ACM SIGSOFT international symposium on Foundations of software engineering*. ACM, 2006, pp. 23–34.

[8] T. A. Corbi, “Program understanding: Challenge for the 1990’s,” *IBM Syst. J.*, vol. 28, no. 2, pp. 294–306, Jun. 1989. [Online]. Available: http://dx.doi.org/10.1147/sj.282.0294

[9] A. J. Ko, B. A. Myers, M. J. Coblenz, and H. H. Aung, “An exploratory study of how developers seek, relate, and collect relevant information during software maintenance tasks,” *IEEE Trans. Softw. Eng.*, vol. 32, no. 12, pp. 971–987, Dec. 2006. [Online]. Available: http://dx.doi.org/10.1109/TSE.2006.116

[10] D. Čubranić and G. C. Murphy, "Hipikat: Recommending pertinent software development artifacts," in *Proceedings of the 25th International Conference on Software Engineering*, ser. ICSE '03. Washington, DC, USA: IEEE Computer Society, 2003, pp. 408–418. [Online]. Available: http://dl.acm.org/citation.cfm?id=776816.776866

[11] T. Schummer, "Lost and found in software space," in *Proc 34 th HICSS*. IEEE, 2001.

[12] V. Sinha, R. Miller, and D. Karger, "Relo: Helping users manage context during interactive exploratory visualization of large codebases," in *Visual Languages and Human-Centric Computting (VL/ HCC*. IEEE Computer Society, 2006, pp. 4–8.

[13] R. DeLine, A. Khella, M. Czerwinski, and G. Robertson, "Towards understanding programs through wear-based filtering," in *Proceedings of the 2005 ACM Symposium on Software Visualization*, ser. SoftVis '05. New York, NY, USA: ACM, 2005, pp. 183–192. [Online]. Available: http://doi.acm.org/10.1145/1056018.1056044

[14] C. Parnin and C. Görg, "Building usage contexts during program comprehension," in *14th International Conference on Program Comprehension (ICPC 2006), 14-16 June 2006, Athens, Greece*, 2006, pp. 13–22. [Online]. Available: http://dx.doi.org/10.1109/ICPC.2006.14

[15] M. P. Robillard and G. C. Murphy, "Automatically inferring concern code from program investigation activities," in *18th IEEE International Conference on Automated Software Engineering (ASE 2003), 6-10 October 2003, Montreal, Canada*, 2003, pp. 225–235. [Online]. Available: http://doi.ieeecomputersociety.org/10.1109/ASE.2003.1240310

[16] J. Singer, R. Elves, and M. Storey, "Navtracks: Supporting navigation in software," in *Program Comprehension, 2005. IWPC 2005. Proceedings. 13th International Workshop on*. IEEE, 2005, pp. 173–175.

[17] A. T. T. Ying, G. C. Murphy, R. Ng, and M. C. Chu-Carroll, "Predicting source code changes by mining change history," *IEEE Trans. Softw. Eng.*, vol. 30, no. 9, pp. 574–586, Sep. 2004. [Online]. Available: http://dx.doi.org/10.1109/TSE.2004.52

[18] T. Zimmermann, P. Weisgerber, S. Diehl, and A. Zeller, "Mining version histories to guide software changes," in *Proceedings of the 26th International Conference on Software Engineering*, ser. ICSE '04. Washington, DC, USA: IEEE Computer Society, 2004, pp. 563–572. [Online]. Available: http://dl.acm.org/citation.cfm?id=998675.999460

[19] D. Piorkowski, S. D. Fleming, C. Scaffidi, L. John, C. Bogart, B. E. John, M. M. Burnett, and R. K. E. Bellamy, "Modeling programmer navigation: A head-to-head empirical evaluation of predictive models." in *VL/HCC*, G. Costagliola, A. J. Ko, A. Cypher, J. Nichols, C. Scaffidi, C. Kelleher, and B. A. Myers, Eds. IEEE, 2011,

pp. 109–116. [Online]. Available: http://dblp.uni-trier.de/db/conf/vl/vlhcc2011.html#PiorkowskiFSJBJBB11

[20] D. Piorkowski, S. Fleming, C. Scaffidi, C. Bogart, M. Burnett, B. John, R. Bellamy, and C. Swart, "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*, ser. CHI '12. New York, NY, USA: ACM, 2012, pp. 1471–1480. [Online]. Available: http://doi.acm.org/10.1145/2207676.2208608

[21] H. Sackman, W. J. Erikson, and E. E. Grant, "Exploratory experimental studies comparing online and offline programming performance," *Commun. ACM*, vol. 11, no. 1, pp. 3–11, Jan. 1968. [Online]. Available: http://doi.acm.org/10.1145/362851.362858

[22] N. R. Augustine, *Augustine's Laws and Major System Development*. Defense Systems Management Review, 1979.

[23] C. B. Seaman, "Qualitative methods in empirical studies of software engineering," *IEEE Trans. Softw. Eng.*, vol. 25, no. 4, pp. 557–572, Jul. 1999. [Online]. Available: http://dx.doi.org/10.1109/32.799955

[24] M. W. van Someren, Y. F. Barnard, and J. A. C. Sandberg, *The Think Aloud Method: A Practical Guide to Modelling Cognitive Processes*. Academic Press Limited, 1994.

[25] S. McConnell, "What does 10x mean? measuring variations in programmer productivity," in *Making Software: What Really Works, and Why We Believe It*. O'Reilly Media, Inc., 2010, ch. Chapter Thirty.

Appendix A

Study Documents and Materials

A.1 IRB Approval Letter

On the following pages, we include the IRB approval letter for the study.

THE UNIVERSITY OF MEMPHIS
Dreamers. Thinkers. Doers.

Institutional Review Board

315 Administration Bldg.
Memphis, TN 38152-3370
Office: 901.678.3074
Fax: 901.678.2199

Consent to Participate in a Research Study

Understanding How Developers Forage for Information during the Maintenance of Multilingual Software

WHY ARE YOU BEING INVITED TO TAKE PART IN THIS RESEARCH?

You are being invited to take part in a research study about how developers maintain multilingual software. You are being invited to take part in this research study because you are experienced in the development of multilingual web applications base on Java EE. If you volunteer to take part in this study, you will be one of about 20 people to do so.

WHO IS DOING THE STUDY?

The person in charge of this study is Alka Singh of the University of Memphis Department of Computer Science. She is being guided in this research by Dr Scott Fleming. There may be other people on the research team assisting at different times during the study.

WHAT IS THE PURPOSE OF THIS STUDY?

By doing this study, we hope to learn about the challenges that developers face in maintaining multilingual software, and about the strategies that developers use to cope with those challenges.

ARE THERE REASONS WHY YOU SHOULD NOT TAKE PART IN THIS STUDY?

You could be excluded from this study if you are not currently enrolled in COMP/EECE 4081.

WHERE IS THE STUDY GOING TO TAKE PLACE AND HOW LONG WILL IT LAST?

The research procedures will be conducted at Dunn Hall. You will need to come to a laboratory in Dunn Hall one time during the study. Each of those visits will take about 2.5 hours. The total amount of time you will be asked to volunteer for this study is 2.5 hours over the next month.

WHAT WILL YOU BE ASKED TO DO?

As a participant in this study, you will be asked to part in a session in which you spend the majority of the time working on a programming task. You will be asked to “think aloud” as you work on the task. That is, you will be asked to continually say whatever you are looking at, thinking, doing, and feeling, as you go about the task. Prior to the task, you will also be asked to fill out a background questionnaire, and you will perform a short warm-up task in which you practice thinking aloud.

Throughout the session, you will be videotaped and audio recorded.

Institutional Review Board

315 Administration Bldg.
Memphis, TN 38152-3370
Office: 901.678.3074
Fax: 901.678.2199

WHAT ARE THE POSSIBLE RISKS AND DISCOMFORTS?

To the best of our knowledge, the things you will be doing have no more risk of harm than you would experience in everyday life.

There is only a minimal psychological and social risk stemming from judgment of the your performance on the task.

WILL YOU BENEFIT FROM TAKING PART IN THIS STUDY?

There is no guarantee that you will get any benefit from taking part in this study. However, the experience of performing the development task may increase your expertise in maintaining multilingual software. Your willingness to take part, however, may, in the future, help society as a whole better understand this research topic.

DO YOU HAVE TO TAKE PART IN THE STUDY?

If you decide to take part in the study, it should be because you really want to volunteer. You will not lose any benefits or rights you would normally have if you choose not to volunteer. You can stop at any time during the study and still keep the benefits and rights you had before volunteering. As a student, if you decide not to take part in this study, your choice will have no effect on you academic status or grade in the class.

IF YOU DON'T WANT TO TAKE PART IN THE STUDY, ARE THERE OTHER CHOICES?

If you do not want to take part in the study, you may alternatively earn the reward (which is described below) by writing a short essay. The essay should take an in-depth look at one of the methods/techniques covered in the course. It should describe the strengths/weaknesses of the technique, define the scope of applicability, and thoroughly back-up your position from the literature (books or research papers). For sources, you should look to recent (within last 10 years) conferences in software engineering (e.g., ICSE, ASE, and FSE) or publication by the ACM or IEEE. In terms of format: The paper should be no less than 2 pages in the IEEE conference-proceedings format (10-point, Times Roman font, two columns), and you must cite at least 3 references.

WHAT WILL IT COST YOU TO PARTICIPATE?

You may have to pay for the cost of getting to the study site and a parking fee.

WILL YOU RECEIVE ANY REWARDS FOR TAKING PART IN THIS STUDY?

You will receive 1.5% added to your final percentage grade for the class for taking part in this study. Example: 86% + 1.5% = 87.5%. The extra credit will be awarded upon completion of the given task.

WHO WILL SEE THE INFORMATION THAT YOU GIVE?

We will make every effort to keep private all research records that identify you to the extent allowed by law.

Your information will be combined with information from other people taking part in the study. When we write about the study to share it with other researchers, we will write about the combined information we have gathered. You will not be personally identified in these written materials. We may publish the results of this study; however, we will keep your name and other identifying information private.

We will make every effort to prevent anyone who is not on the research team from knowing that you gave us information, or what that information is.

The questionnaire and video data will be stored in an office on campus. To protect the data, the computer holding the video data will be password protected, and the office will be kept locked. When our analysis of the data is complete, the data will be destroyed. As an additional constraint, the data will be destroyed within two years of when it was collected.

For grading purposes, the investigator will provide a list of participants' names to the course instructor. Information about your performance in the study will be kept confidential from the instructor; however, the investigator may share aggregate and/or anonymized study data with instructor.

We will keep private all research records that identify you to the extent allowed by law. However, there are some circumstances in which we may have to show your information to other people. For example, the law may require us to show your information to a court. Also, we may be required to show information which identifies you to people who need to be sure we have done the research correctly; these would be people from such organizations as the University of Memphis.

CAN YOUR TAKING PART IN THE STUDY END EARLY?

If you decide to take part in the study you still have the right to decide at any time that you no longer want to continue. You will not be treated differently if you decide to stop taking part in the study.

The individuals conducting the study may need to withdraw you from the study. This may occur if you are not able to follow the directions they give you, if they find that your being in the study is more risk than benefit to you, or if the agency funding the study decides to stop the study early for a variety of scientific reasons.

WHAT IF YOU HAVE QUESTIONS, SUGGESTIONS, CONCERNS, OR COMPLAINTS?

Before you decide whether to accept this invitation to take part in the study, please ask any questions that might come to mind now. Later, if you have questions, suggestions, concerns, or complaints about the study, you can contact the investigator, Alka Singh at 901-679-5930, or her advisor, Scott Fleming at 901-678-3142. If you have any questions about your rights as a volunteer in this research, contact the Institutional Review Board staff at the University of Memphis at 901-678-3074. We will give you a signed copy of this consent form to take with you.

WHAT IF NEW INFORMATION IS LEARNED DURING THE STUDY THAT MIGHT AFFECT YOUR DECISION TO PARTICIPATE?

If the researcher learns of new information in regards to this study, and it might change your willingness to stay in this study, the information will be provided to you. You may also be asked to sign a new informed consent form if the information is provided to you after you have joined the study.

Institutional Review Board

315 Administration Bldg.
Memphis, TN 38152-3370
Office: 901.678.3074
Fax: 901.678.2199

☐ By checking this box, you agree to be videotaped and audio recorded for the study.

______________________________________	___________
Signature of person agreeing to take part in the study	Date
______________________________________	
Printed name of person agreeing to take part in the study	
______________________________________	___________
Name of [authorized] person obtaining informed consent	Date

A.2 IRB Informed Consent

On the following pages, we include the IRB-approved informed consent document for the study.

THE UNIVERSITY OF MEMPHIS
Dreamers. Thinkers. Doers.

Institutional Review Board

315 Administration Bldg.
Memphis, TN 38152-3370
Office: 901.678.3074
Fax: 901.678.2199

Consent to Participate in a Research Study

Understanding How Developers Forage for Information during the Maintenance of Multilingual Software

WHY ARE YOU BEING INVITED TO TAKE PART IN THIS RESEARCH?

You are being invited to take part in a research study about how developers maintain multilingual software. You are being invited to take part in this research study because you are experienced in the development of multilingual web applications base on Java EE. If you volunteer to take part in this study, you will be one of about 20 people to do so.

WHO IS DOING THE STUDY?

The person in charge of this study is Alka Singh of the University of Memphis Department of Computer Science. She is being guided in this research by Dr Scott Fleming. There may be other people on the research team assisting at different times during the study.

WHAT IS THE PURPOSE OF THIS STUDY?

By doing this study, we hope to learn about the challenges that developers face in maintaining multilingual software, and about the strategies that developers use to cope with those challenges.

ARE THERE REASONS WHY YOU SHOULD NOT TAKE PART IN THIS STUDY?

You could be excluded from this study if you are not currently enrolled in COMP/EECE 4081.

WHERE IS THE STUDY GOING TO TAKE PLACE AND HOW LONG WILL IT LAST?

The research procedures will be conducted at Dunn Hall. You will need to come to a laboratory in Dunn Hall one time during the study. Each of those visits will take about 2.5 hours. The total amount of time you will be asked to volunteer for this study is 2.5 hours over the next month.

WHAT WILL YOU BE ASKED TO DO?

As a participant in this study, you will be asked to part in a session in which you spend the majority of the time working on a programming task. You will be asked to “think aloud” as you work on the task. That is, you will be asked to continually say whatever you are looking at, thinking, doing, and feeling, as you go about the task. Prior to the task, you will also be asked to fill out a background questionnaire, and you will perform a short warm-up task in which you practice thinking aloud.

Throughout the session, you will be videotaped and audio recorded.

Institutional Review Board

315 Administration Bldg.
Memphis, TN 38152-3370
Office: 901.678.3074
Fax: 901.678.2199

WHAT ARE THE POSSIBLE RISKS AND DISCOMFORTS?

To the best of our knowledge, the things you will be doing have no more risk of harm than you would experience in everyday life.

There is only a minimal psychological and social risk stemming from judgment of the your performance on the task.

WILL YOU BENEFIT FROM TAKING PART IN THIS STUDY?

There is no guarantee that you will get any benefit from taking part in this study. However, the experience of performing the development task may increase your expertise in maintaining multilingual software. Your willingness to take part, however, may, in the future, help society as a whole better understand this research topic.

DO YOU HAVE TO TAKE PART IN THE STUDY?

If you decide to take part in the study, it should be because you really want to volunteer. You will not lose any benefits or rights you would normally have if you choose not to volunteer. You can stop at any time during the study and still keep the benefits and rights you had before volunteering. As a student, if you decide not to take part in this study, your choice will have no effect on you academic status or grade in the class.

IF YOU DON'T WANT TO TAKE PART IN THE STUDY, ARE THERE OTHER CHOICES?

If you do not want to take part in the study, you may alternatively earn the reward (which is described below) by writing a short essay. The essay should take an in-depth look at one of the methods/techniques covered in the course. It should describe the strengths/weaknesses of the technique, define the scope of applicability, and thoroughly back-up your position from the literature (books or research papers). For sources, you should look to recent (within last 10 years) conferences in software engineering (e.g., ICSE, ASE, and FSE) or publication by the ACM or IEEE. In terms of format: The paper should be no less than 2 pages in the IEEE conference-proceedings format (10-point, Times Roman font, two columns), and you must cite at least 3 references.

WHAT WILL IT COST YOU TO PARTICIPATE?

You may have to pay for the cost of getting to the study site and a parking fee.

WILL YOU RECEIVE ANY REWARDS FOR TAKING PART IN THIS STUDY?

You will receive 1.5% added to your final percentage grade for the class for taking part in this study. Example: 86% + 1.5% = 87.5%. The extra credit will be awarded upon completion of the given task.

WHO WILL SEE THE INFORMATION THAT YOU GIVE?

We will make every effort to keep private all research records that identify you to the extent allowed by law.

Your information will be combined with information from other people taking part in the study. When we write about the study to share it with other researchers, we will write about the combined information we have gathered. You will not be personally identified in these written materials. We may publish the results of this study; however, we will keep your name and other identifying information private.

We will make every effort to prevent anyone who is not on the research team from knowing that you gave us information, or what that information is.

The questionnaire and video data will be stored in an office on campus. To protect the data, the computer holding the video data will be password protected, and the office will be kept locked. When our analysis of the data is complete, the data will be destroyed. As an additional constraint, the data will be destroyed within two years of when it was collected.

For grading purposes, the investigator will provide a list of participants' names to the course instructor. Information about your performance in the study will be kept confidential from the instructor; however, the investigator may share aggregate and/or anonymized study data with instructor.

We will keep private all research records that identify you to the extent allowed by law. However, there are some circumstances in which we may have to show your information to other people. For example, the law may require us to show your information to a court. Also, we may be required to show information which identifies you to people who need to be sure we have done the research correctly; these would be people from such organizations as the University of Memphis.

CAN YOUR TAKING PART IN THE STUDY END EARLY?

If you decide to take part in the study you still have the right to decide at any time that you no longer want to continue. You will not be treated differently if you decide to stop taking part in the study.

The individuals conducting the study may need to withdraw you from the study. This may occur if you are not able to follow the directions they give you, if they find that your being in the study is more risk than benefit to you, or if the agency funding the study decides to stop the study early for a variety of scientific reasons.

WHAT IF YOU HAVE QUESTIONS, SUGGESTIONS, CONCERNS, OR COMPLAINTS?

Before you decide whether to accept this invitation to take part in the study, please ask any questions that might come to mind now. Later, if you have questions, suggestions, concerns, or complaints about the study, you can contact the investigator, Alka Singh at 901-679-5930, or her advisor, Scott Fleming at 901-678-3142. If you have any questions about your rights as a volunteer in this research, contact the Institutional Review Board staff at the University of Memphis at 901-678-3074. We will give you a signed copy of this consent form to take with you.

WHAT IF NEW INFORMATION IS LEARNED DURING THE STUDY THAT MIGHT AFFECT YOUR DECISION TO PARTICIPATE?

If the researcher learns of new information in regards to this study, and it might change your willingness to stay in this study, the information will be provided to you. You may also be asked to sign a new informed consent form if the information is provided to you after you have joined the study.

THE UNIVERSITY OF MEMPHIS
Dreamers. Thinkers. Doers.

Institutional Review Board

315 Administration Bldg.
Memphis, TN 38152-3370
Office: 901.678.3074
Fax: 901.678.2199

☐ By checking this box, you agree to be videotaped and audio recorded for the study.

____________________________________	____________
Signature of person agreeing to take part in the study	Date
____________________________________	
Printed name of person agreeing to take part in the study	
____________________________________	____________
Name of [authorized] person obtaining informed consent	Date

A.3 Study Session Procedure

On the following pages, we include the procedure used for the study sessions.

Study Procedure

Materials

- Workstation with web browser (e.g., Firefox), Eclipse with subversion plugin, video camera (e.g., a webcam), microphone, and video capture software (e.g., Camtesia), START_HERE folder.
- Pen/pencil and paper

Debugging Session

Initial setup

1. Thank the participants for their participation and collect the signed consent form.
2. Have participant setup their workspace to make sure everything works and is ready to go
3. Save a copy of the project code (so we can do before/after comparison)

Main session

4. Establish context (Read Script 1 below)
5. Start recording video
6. Interview: Ask BACKGROUND questions.
7. Interview: Ask the participant what task(s) he/she will be working on
    a. Explain that it is not required that they complete the tasks, but we ask that they work for a full 2 hours.
8. Give think-aloud instructions and warmup (Read Script 2 below)
9. Participant works while thinking aloud for 2 hours

Script 1 – Establish Context

<ADD BIT ABOUT MULTI-LINGUAL ENVIRONMENT—i.e., Familiarize them with the meaning of MULTI-LINGUAL ENVIRONMENT>

As a participant in this study, you will be working on your project work. As you work, I will ask you to provide verbal reports by thinking aloud. I will explain this more in a moment.

Throughout this study, I will be recording what you say and be videotaping you with this camera <*point out the camera*>. Please do not touch either of these instruments. Additionally, I will be capturing video of the computer screen as you work, and the computer will log everything you do with the interface.

Script 2 – Think-Aloud Instructions and Warmup

I will start by familiarizing you with the procedure for giving verbal reports. In particular, I am going to ask you to solve some practice problems. I am interested in knowing your thoughts as you work through each problem. In order to obtain this information, I am going to ask you to think aloud as you work. What I mean by "think aloud" is that I want you to say your thoughts out loud from the moment you finish hearing a practice question until you say the final answer. I would like you to talk aloud as much as you comfortably can during that time. Don't try to plan or explain what you say. Just act as if you are alone and speaking to yourself. Keep talking while you are coming up with the solution to each problem. If you are silent for a long time, I'll remind you to think aloud. Do you understand what I would like you to do?

Good. Now we will see some practice problems.

<*Play window task video.*>

Now you try thinking aloud. First, listen to the question, then answer it as soon as you can. Are you ready?

How many lights are in your current apartment?

Good. Now, those problems were solved entirely in our heads. However, when you are working on the computer you will be looking at things and seeing things that catch your attention. These things you are looking for and things that you see are as important to our observations as thoughts you are thinking, so please verbalize these too.

Now you will think aloud as you are using the computer. I have pointed this browser to the Computer Science, University of Memphis website. Please think aloud as you find the bioinformatics student project by Pranitha Appidi . Please click around to find this answer, do not use search.

<*Observe when participant does this task. Time bound this task to 3 minutes*>

Great! We can move on to the real task. As you are doing this task, I won't be able to answer any questions. But if a question pops into your head as you work, go ahead and ask it anyway and we can discuss it after the session. If you forget to think aloud, I'll say, "Please keep talking".

Do you have any questions about thinking aloud?

Good. Please begin thinking aloud now.

<*Observe the programmer and remind if he is silent for a long time.*>

<After 2:00 has elapsed>

The time is over. Thank you.

<Stop Video Recording and Save>

Move saved videos to P# folder in server

A.4 Participant-Recruitment Email

On the following pages, we include the participant-recruitment email for the study.

Hi folks,

Here is a way to earn 1.5 A&B points (see below). You may choose only one option (so please don't ask). I'll say more about this opportunity in class.

Cheers,
SDF

= Extra Credit Opportunity =

You may choose one of two extra credit options, each worth 1.5 percentage points that will be added to your final percentage grade for the course. To select one of the options, email your choice to your course instructor, Dr. Scott Fleming (Scott.Fleming@memphis.edu).

== Option #1: Participate in Research Study ==

Wanted: Programmers to participate in a research study about how developers maintain multilingual software.

As a participant, you will take part in a study session that may last up to 2.5 hours. Your main task during the session will be to debug a multilingual Java EE program. You will be asked to "think aloud" as you work on the task. That is, you will be asked to continually say whatever you are looking at, thinking, doing, and feeling, as you go about the task.

This research is conducted under the direction of Alka Singh (faculty advisor: Dr. Fleming) from the Computer Science Department at the University of Memphis. If you choose the study option, Dr. Fleming will provide your email to Ms. Singh, so she can contact you regarding scheduling arrangements. The study sessions will be conducted at Dunn Hall. Your session will be scheduled based on your availability.

== Option #2: Essay ==

The essay should take an in-depth look at one of the methods/techniques covered in the course. It should describe the strengths/weaknesses of the technique, define the scope of applicability, and thoroughly back-up your position from the literature (books or research papers). For sources, you should look to recent (within last 10 years) conferences in software engineering (e.g., ICSE, ASE, and FSE) or publication by the ACM or IEEE. In terms
of format: The paper should be no less than 2 pages in the IEEE conference-proceedings format (10-point, Times Roman font, two

columns), and you must cite at least 3 references.

A.5 Background Questionnaire

On the following pages, we include the background questionnaire that participants filled out.

Background Questionnaire

1. Age range:
   - ☐ 18–19
   - ☐ 20s
   - ☐ 30s
   - ☐ 40s
   - ☐ 50 or over

2. Education (highest degree or level completed):
   - ☐ High school graduate - high school diploma or the equivalent (for example: GED)
   - ☐ Some college credit, but less than 1 year
   - ☐ 1 or more years of college, no degree
   - ☐ Associate degree (for example: AA, AS)
   - ☐ Bachelor's degree (for example: BA, AB, BS)
   - ☐ Master's degree (for example: MA, MS, MEng, MEd, MSW, MBA)
   - ☐ Professional degree (for example: MD, DDS, DVM, LLB, JD)
   - ☐ Doctorate degree (for example: PhD, EdD)

3. Current field of study or major:
   - ☐ Computer Science
   - ☐ Computer Engineering
   - ☐ Electrical Engineering
   - ☐ Mechanical Engineering
   - ☐ Mathematics

   - ☐ Other: ____

4. Is English your primary language?
   - ☐ Yes
   - ☐ No

5. If not, then what is your primary language? _____________________________________

6. How many years of…

   a. … programming experience: ____

   b. … professional programming experience: _______________________________

   c. … multilingual programming experience: _______________________________

   d. … professional multilingual programming experience: ____________________