The Pennsylvania State University

The Graduate School

College of Information Sciences and Technology

LEARNING SOFTWARE DEVELOPMENT: WHEN DO STUDENTS NEED HELP?

A Thesis in

Information Sciences and Technology

by

Benjamin David Eppinger

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Master of Science

May 2019

The thesis of Benjamin David Eppinger was reviewed and approved* by the following:

Steven R. Haynes
Teaching Professor of Information Sciences and Technology
Thesis Advisor

Benjamin V. Hanrahan
Assistant Professor of Information Sciences and Technology

Frederico T. Fonseca
Associate Professor of Information Sciences and Technology

Mary Beth Rosson
Professor of Information Sciences and Technology
Associate Dean for Graduate Programs

*Signatures are on file in the Graduate School

ABSTRACT

There are many challenges that a person faces in their day-to-day lives. These challenges range in difficulty depending on the specific issue they are dealing with at the time. There are a variety of sources in which a person can turn to for help. Some of these sources include peers, online search engines, and books. While there are many resources and ample knowledge of the various types of resources, we do not necessarily know when people decide to use the resources at their disposal. When does a person reach an inflection point to where their mental knowledge-bank switches from using past knowledge to looking for help? Using various learning theories, minimalistic documentation, knowledge bootstrapping, and educational psychology theories, this paper aims to explore this question with a focus on students who are learning software development. In the presented study, middle-level college students were asked about their use of resources for solving development problems. Situations in which the students felt they could not handle the problem on their own included: when group members could not help each other, students found syntactical or configuration errors within their programs that were not trivial to solve, and after searching for information online. The students generally experienced issues when working with an unfamiliar topic, but not when initially starting work on that topic. Conversations, in person with other people, were found to be highly beneficial in working through problems the students experienced. In addition to the study's findings, future work in this area is discussed.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I want to thank my thesis advisor, Steve Haynes, for all of the guidance he has given me throughout my time at Penn State. I have always appreciated his methods of teaching. I was quite excited when I was able to have him be my thesis advisor and when I was given the opportunity to be an instructional assistant for his classes.

I would also like to thank my committee member, Ben Hanrahan. Ben gave me exceptional feedback on my draft, and I really appreciate his guidance both for my thesis and in classes. He rarely tells me what I want to hear, but it’s always for my own benefit.

I want to thank my other committee member, Fred Fonseca. He was a great professor and always supportive of my endeavors. I learned a lot about agile development from Fred and will undoubtedly continue to utilize that knowledge throughout my future career.

Going back into my early collegiate days, I would like to thank my Math 140 professor, Mike Weiner, from when I was taking Penn State Altoona courses in high school. I spent way too many hours pestering him about calculus problems and college in general, even after I finished his class. If it was not for Mike, I likely would not have ended up coming to Penn State.

Lastly, I would never hear the end of it if I did not thank my residents in the IST Special Living Option. They gave me the idea for my thesis title, and they have been with me my last three years, through ups and downs. They make it a goal to increase the number of gray hairs I acquire each year, and they ensure that I always have a crazy situation to triage as their live-in resident assistant.

Chapter 1

Introduction

People face challenges as they learn throughout their everyday lives. These challenges can range in difficulty, from very simple to very complex, depending on the issue at hand. When faced with a difficult task, people turn to a variety of sources for guidance (Illinois University Library, 2018). The source that a person turns to can vary based on the specific individual and the topic they are trying to solve (Illinois University Library, 2018).

With the creation of the internet, people are turning more to online search engines for their information than conventional documentation resources (Henry, 2006). To go along with this trend, students are now being taught in schools to look for information more effectively rather than memorize as much as they can (Henry, 2006). A definition of documentation is "the usually printed instructions, comments, and information for using a particular piece or system of computer software or hardware" ("Documentation," 2019). In the past, the documentation for software has been the official references that accompany a program (Lethbridge, Singer, & Forward, 2003; Uddin & Robillard, 2015). However, with the expanding types of resources a person now has access to, people are utilizing many more sources than the canonical documentation (Stack Overflow, 2018). People are using resources that have varying levels of fluidity, or the ability to change over time (Bishop, Van House, & Buttenfield, 2003; Daniels, Faniel, Fear, & Yakel, 2012; Levy, 1994). When people look for information, they use and read what would be considered learning resources (Goldie, 2016; Lebeničnik, Pitt, & Starčič, 2015; Thelwall, 2002).

Even before the inclusion of web-based information, documentation and resource design has had many iterations over the years. As early as 1985, designers were trying to build more iterative designs while catering to end users' goals and mannerisms (Gould & Lewis, 1985). In 1990, John Carroll developed the concept of minimalism to address some of the challenges that can go into creating effective documentation (Carroll, 1990). In the early to mid-2000s, standards were directed more towards interaction based references, but still utilizing minimalistic approaches (Rosson & Carroll, 2002). These references utilized scenario-based interactions, coupled with prototyping tactics, to develop a person's understanding of the tool or concept (Rosson & Carroll, 2002). While originating around 1984 as a topic in psychology (Huesmann, Card, Moran, & Newell, 1984), by 2005 the field of human-computer interaction (HCI) had advanced to become a core component of software design and software documentation design (Dix, Finlay, Abowd, & Beale, 2005). HCI is focused on how people can positively maximize their interaction with computers (Xie et al., 2019), and a seemingly steady component of HCI has been documentation (Wulff & Mahling, 1990).

There are a variety of places that a person can utilize in order to solve problems; these can include books, eBooks, databases, journals, reference services, digital media, along with many others (Ranganadham & Surendra Babu, 2012). While there is a substantial amount of knowledge on documentation types, the process of finding the answer and when students use documentation has often been overlooked (Piech, Blikstein, Cooper, Sahami, & Koller, 2012). The change from remembering learned information to information searching can be called an inflection point in the learning process (Luce, 2012; Thurstone, 1930). Throughout this thesis, there is the exploration of situations that cause people to turn to a specific type of documentation. There is also a detailing of when they reach these inflection points as part of trying to solve a development related task or problem.

Solving problems relies on the ability to implement prior knowledge to solve the problem at hand (Lambiotte & Dansereau, 1992; Nesbit & Adesope, 2006; van Riesen, Gijlers, Anjewierden, & de Jong, 2018). The practice of putting together the knowledge that a person already has is sometimes known as bootstrapping knowledge (Darling & Havelka, 2010). From prior experience, a field that requires a substantial amount of bootstrapped knowledge is software development. A person either has to rely on what they know already, or they must seek insight from an external source and repeat the problem-solving process (Klein, 2017; Sweller, 1988). This problem-solving methodology is what is explored throughout this thesis, focusing on the inflection point in which the decision is made to either complete the problem or pursue external insight from information resources.

Chapter 2

Review of Literature

There is a variety of literature that needs to be explained for this topic. Starting with the psychology of learning, the concepts of learning are established. There are many different stances on the psychology of learning, so the most applicable theories are described. Constructivism and cognitive maps are the theories most closely aligned with the study later presented in this thesis. Constructivism focuses on people constructing new knowledge by relating their prior experiences, and cognitive maps explain problem-solving tactics frequently seen in the learning process (Olson & Hergenhahn, 2015).

Closely aligned with cognitive maps is the concept of bootstrapping. Bootstrapping knowledge is explained in relation to learning software development. Following bootstrapping is an introduction to minimalistic documentation. Minimalistic documentation can have an effect on a person's understanding of a system and can be used as a guidebook for building useful learning resources (Palmer, 2007; Rauterberg, Menozzi, & Wesson, 2003).

There is then an introduction to tools that can help software developers solve problems in their code, along with some examples of errors that a developer may face while coding.

Psychology of Learning

The field of educational psychology tends to focus on many of the learning principles and patterns covered in minimalism and documentation standards (Craik & Lockhart, 1972; Olson & Hergenhahn, 2015; Pavlov, 2010; Wozniak, 1999). Even with an established field of study, there is still no 'one' completely agreed upon stance on learning and what learning entails (Vygotsky, 1962; Wertheimer, 2011). Even with no one agreed upon stance, there are several different theories on how people learn. Behaviorism, Cognitivism, and Constructivism have been the foundations of today's status in educational psychology (Olson & Hergenhahn, 2015). Behaviorism focuses on conditioning techniques with rewards and punishments; Cognitivism focuses on the encoding and retrieval of previous experiences, and Constructivism focuses on people constructing new knowledge by relating their prior experiences (Olson & Hergenhahn, 2015; Wertheimer, 2011).

It is important to highlight the theory of learning most closely associated with this study, as these theories provide a base for overall learning and education. While the study presented later in this thesis does not aim to prove or disprove constructivism, it is most aligned to the theory of constructivism. Constructivism focuses on how people use past experiences to build relations to current situations and information (Olson & Hergenhahn, 2015), which is a significant component of the study presented in this thesis. When people experience software development problems, they will tend to rely on prior situations and knowledge that they have experienced (Yeh, 2018). As a person handles more and more challenging problems while they advance as a developer, this reliance on past experiences is a critical thought-process for software development.

Cognitive Maps

If we take into consideration that there can be multiple solutions to the same problem, it becomes important to talk about Edward Tolman's concept of cognitive maps. A cognitive map is a gradually created mental map of the environment to which something is exposed (Olson & Hergenhahn, 2015). Tolman performed studies on rats as they navigated through a maze (Tolman, 1948). Over time, the rat would learn the maze paths and be able to reach the food located at the exit. Tolman claims that once the rat develops its cognitive map of the maze, the rat can use alternative paths to get to the end. For example, if one path is blocked, it merely uses another path in the maze (Olson & Hergenhahn, 2015). While Tolman did his studies on rats, he translated his results to be reflective of human behavior. In this case, once a person builds their cognitive map, they can achieve their goal through alternative means if the previous path is blocked (Tolman, 1948).

If we take the concept of the cognitive map and apply it to a student who is learning software development, we can infer that a student can seek alternative paths to an end goal if the prior path is no longer working. Being able to use different tactics, or even a different programming language to solve a development problem is a vital skill that is often referenced when talking about introductory computer science courses (Kölling et al., 1995; Meghabghab, 2002). The vital question to take away from Tolman's studies is how many alternative paths would the student try before assuming that their cognitive map is failing? In other words, when would the student reach the point where the knowledge they are seeking is beyond the capacity of their current cognitive map?

Tolman continued his work by investigating field-cognition modes; this is where the person will utilize a strategy that has worked in the past and apply it to the current scenario (Olson & Hergenhahn, 2015). He also focuses on the concept of drive discriminations, which is that the person does not know how to interpret its cognitive map if they do not know what their end goal should be (Olson & Hergenhahn, 2015). These two concepts are not utilized in the study presented in this thesis but could be utilized for future work in the area.

Bootstrapped Knowledge

Very similar to the cognitive map theory is the concept of structural bootstrapping. While a cognitive map is a mental map of an environment (Tolman, 1948), bootstrapping is the acquisition of additional knowledge over time (Wörgötter et al., 2015). The difference between the two is that the cognitive map is mainly shown with visuospatial information (Mervis, Robinson, & Pani, 1999), whereas bootstrapped knowledge is shown with other forms of information. In addition to visuospatial information, other types of information can be auditory or kinesthetic (Hasibuan & Nugroho, 2016; Schmeck, 1994). In either system, people will rely on a knowledge-base to make future learning more efficient. To learn additional items, they do not have to relearn what they have experienced previously. Instead, they rely on a knowledge-base in which they have constructed over time (Wörgötter et al., 2015).

Fundamental to the concept of knowledge bootstrapping is the idea that a person has a knowledge-base that is detailed and structurally sound to the point in which they can rely on the knowledge-base to solve additional problems (Wörgötter et al., 2015). This reliance on a knowledge-base aligns with minimalism's iterative learning approach. Instructional design should allow for material to be added before each iteration (Carroll, Communication, Anson, Brockmann, & Draper, 1998), thus expanding the size of the knowledge-base.

The accumulation of a knowledge base is used as people learn more advanced or alternative concepts (Aksoy et al., 2013; Wörgötter et al., 2015). The reliance on prior information is generally successful in these studies as a means to improve the efficiency of learning. The approach that is detailed by structural bootstrapping allows for quick generalizations and the ability to acquire new knowledge with a minimal amount of training material (Do, Schill, Ernesti, & Asfour, 2014).

Minimalistic Documentation

Minimalism is often used in creating useful training materials for situations that people can learn (Farkas, 1998; Palmer, 2007; Rauterberg et al., 2003; Silveira, Diniz, Barbosa, & Sieckenius De Souza, 2004). Many learning concepts go into minimalism and documentation creation as a whole. There has been a significant amount of research on minimalism and documentation standards, but there are still unexplored areas (Aguiar, 2000, 2005; Delanghe, 2000; Delattre & Grohan, 2013; Farkas, 1998; Van der Meij, 2008). A portion of minimalism that can be expanded on is when people use documentation given sets of problems.

The concept of minimalism contains a variety of constructs which include: action-oriented approaches, error recognition and recovery practices, exploration and guidance, and the ability to have a fast start to learning (Carroll et al., 1998). These four constructs are explained as follows:

Action-oriented approaches:

Minimalism should encourage users to explore and offer immediate invitations for users to achieve their tasks (Carroll et al., 1998). Users need to be pushed to try things on their own and without step-by-step walkthroughs. Being a useful resource for the user is more important than forcing them to refer to the documentation before using the software (Carroll et al., 1998). Users learn best when they are engaged and interacting with the task (Carroll et al., 1998).

Error Recognition and Recovery:

Users who are learning will make numerous and a variety of mistakes as they progress (Carroll et al., 1998). Between 25 and 50 percent of the user's time is spent fixing problems and finding ways to reduce the frequency of and time spent on errors is a critical point of minimalism (Carroll et al., 1998).

Exploration and Guidance:

Minimalistic techniques focus on being brief and encourage locating the specific area of interest (Carroll et al., 1998). It is encouraged to allow people to read sections out of order as a user may need to refer to only a specific part rather than the entire text. Learning is regarded as the most effective when the instruction builds on itself over time (Carroll et al., 1998). This iterative approach is similar to expanding a person's amount of bootstrapped knowledge.

Fast, Initial Start to Learning:

Minimalistic text needs to encourage users to attempt the activity before reading (Carroll et al., 1998). The text should not slow the user down, but instead, be a reference for users as they may need it. Because the documentation is action oriented as mentioned previously, the user should not need to read the entirety of the text before being able to start their attempt (Carroll et al., 1998).

Throughout the lifespan of minimalism, there have been numerous studies on its effectiveness. One study aims to define complex tasks and show if the mapping between goal-oriented tasks is sufficient for use as a learning resource (Woods & Hollnagel, 1987). There is also the argument that states minimalistic instruction is usually less effective and does not consider working memory in material design (Kirschner, Sweller, & Clark, 2006). An example given is a case of medical students who provided less coherent answers with more errors when taught using minimalistic techniques (Kirschner et al., 2006). The critical thing to note from this is that many domains have specific answers to problems, which tend to obfuscate the benefits of minimalistic learning techniques (Alrahlah, 2016; Wood, 2003). A field like the medical industry has these specific solutions to their problems, whereas in software development people can often design multiple acceptable solutions to the same problem (Buschmann, Henney, & Schmidt, 2007). Developers are not often limited to specific answers, as there are numerous ways to solve their problem. Two developers can be given the same problem, use two completely different programming languages, and arrive at similar outputs which meet the answer specifications.

Learning in Software Development

The skills that are most frequently claimed for software development are problem-solving and mathematical ability (Jenkins, 2002). These skills are measured by the scales of learning styles and motivation levels. On one end of the scale is surface learning, which is a very superficial memorization-based approach to learning course materials. Surface learning is argued by Jenkins as most relevant to learning syntax for software code. However, he also provides support for the opposing concept, deep learning, used to establish an intricate understanding of software development and design. While surface learning appears to be the opposite of deep learning, deep learning would be when the person masters a concept. They both have benefits and situations in which one is better than the other. How a person uses the two skills and how they fall on the two scales can be deterministic of success in software development (Jenkins, 2002).

Software development continues to prove a challenging subject for many students (Leano, Chattopadhyay, & Sarma, 2017; Silveira, Barbosa, & Souza, 2004; Yeh, 2018). Some people feel that universities should not teach software development to first-year college students, and there should not be assessments in this field (Jenkins, 2002). However, from these beliefs, it is essential to extract the core idea that software development is a difficult subject to learn in an in-person, university classroom, environment. It requires a large amount of creativity, memorization of syntax, the ability to adapt to new syntax configurations, and abstract planning of application design to be effective in this field (Jenkins, 2002).

Some approaches include teaching interventions for syntactical errors. By studying which syntax errors students most frequently encounter, instructors can focus on ensuring students know how to fix the most common problems in development (Denny, Luxton-Reilly, & Tempero, 2012). While most studies focus on the standard errors that students experience, there has been little work about *when* students experience problems in development. At what point in the development process do the students experience situations where they need help? If we isolate the situations that cause students to experience errors and problems, would we be able to adapt how we educate students to have less of these situations over time? The study presented in this thesis aims to identify these situations, but future research can test whether the situations can be minimized. Learning resources, learning theories, and course planning techniques have the potential to be critical in helping these students progress through the journey of software development mastery.

Leveraging Development Tools as a Learning Resource

Similar to using different programming languages, other things can differ between programmers. In a professional environment students and professionals often utilize what is known as an Integrated Development Environment (IDE). There are many different IDEs that can be used, which are often dependent on the programming language (Dyke, 2011; Vihavainen, Helminen, & Ihantola, 2014). In most situations, the IDE performs software code evaluations while the developer produces code. If there are any syntax errors, the IDE flags the problem(s) for the developer (Dyke, 2011; Vihavainen et al., 2014). This system does not catch all potential errors in development (such as logic or configuration errors), but given the iterative nature of software development, the developer finds most errors either in development or testing (Dyke, 2011; Vihavainen et al., 2014).

There are many different types of errors that a person can encounter while developing software (Brown & Altadmri, 2014; Vihavainen et al., 2014). Stylistic errors, syntax errors, environmental constraints, compilation errors, and runtime errors are some of the many types of adverse situations that programmers can encounter (Brown & Altadmri, 2014; Vihavainen et al., 2014). For this thesis, a logic error is defined as when a person does not know how to use a tool or framework effectively. For example, a logic error would occur if a person does not know how to implement a Java GridPane into their application. They may know the syntax, but they do not know what to do with that code.

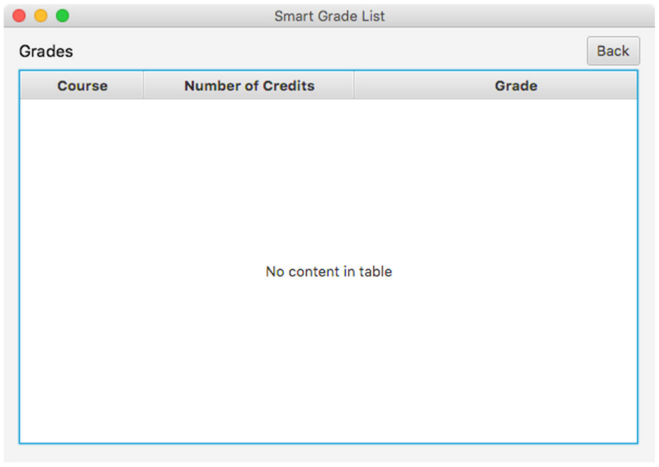
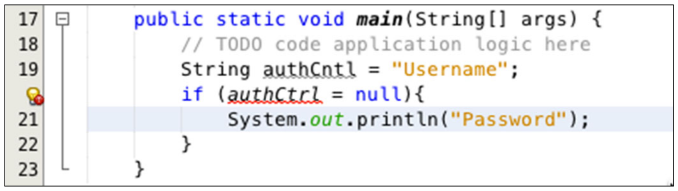
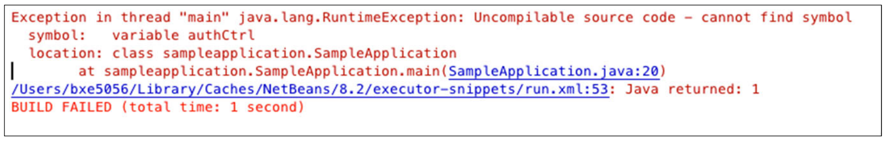


Figure **2-1**: Stylistic errors can often go unnoticed while coding. As depicted in this photo, there is a table but no content. The program still runs, but the data is not shown in the table. While this is a relatively obvious error, stylistic errors can be less noticeable. Utilizing the IDE debugger can often help to solve these issues as it allows the developer to iterate through the code line by line until they find the problem.



```
    public static void main(String[] args) {
        // TODO code application logic here
        String authCntl = "Username";
        if (authCtrl = null){
            System.out.println("Password");
        }
    }
```

Figure **2-2**: Syntax errors can consist of variable misspellings. As depicted here, the variable *authCntl* is incorrectly spelled as *authCtrl*. The IDE will often highlight syntax errors and provide a suggestion on ways to fix the error. In this photo, NetBeans IDE underlines the incorrect variable and adds a lightbulb icon on the number line to indicate a problem with the line of code.



```
Exception in thread "main" java.lang.RuntimeException: Uncompilable source code - cannot find symbol
  symbol:   variable authCtrl
  location: class sampleapplication.SampleApplication
	at sampleapplication.SampleApplication.main(SampleApplication.java:20)
/Users/bxe5056/Library/Caches/NetBeans/8.2/executor-snippets/run.xml:53: Java returned: 1
BUILD FAILED (total time: 1 second)
```

Figure **2-3**: Compilation errors are often found by the IDE as syntax errors, but are ultimately errors that cause the program not to run. In other words, the source code is not able to finish compiling. The IDE will often indicate the specific line that contains the error as a means to help the developer remedy the issue.

Resource Types in Software Development

As mentioned previously, there are many different resources that people can use when solving problems (Illinois University Library, 2018; Ranganadham & Surendra Babu, 2012). People relatively new to development tend to rely on prior experiences and model the current problem after previous problems (Yeh, 2018). With recent advances in technology, various web-based information resources have been created for developers. Some of the most extensive web-based resources are sites like Stack Overflow, Mozilla Developer Network (MDN), and Massive Open Online Courses (also known as MOOCs) (Earnes, 2014; Stringfellow, 2013). These sites often have a significant returning user base, with Stack Overflow logging 86% of their polled professional developers using the site a few times per week or more (Stack Overflow, 2018).

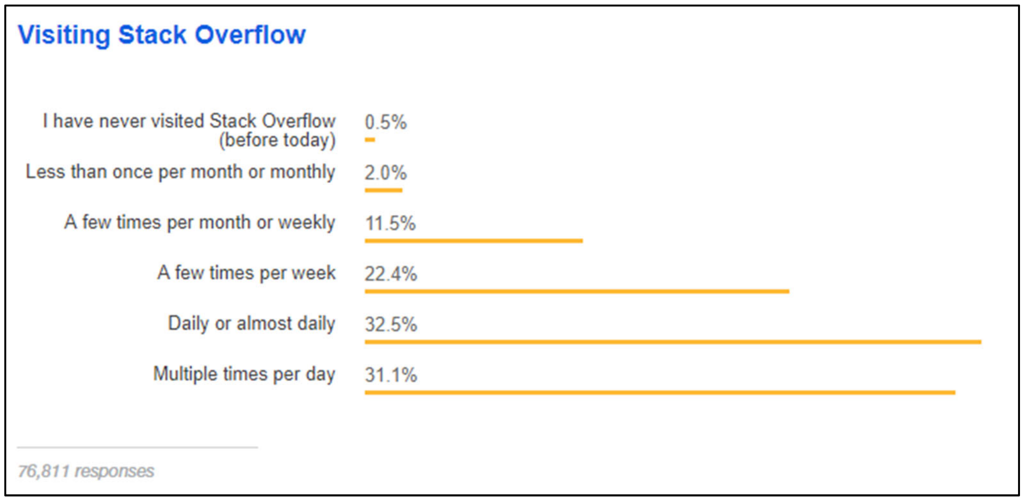


Figure **2-4**: Stack Overflow's representation of the number of returning users and how often those surveyed visited the site. Most of those surveyed by Stack Overflow visit the resource more than weekly. Image from Stack Overflow with permission. (Stack Overflow, 2018)

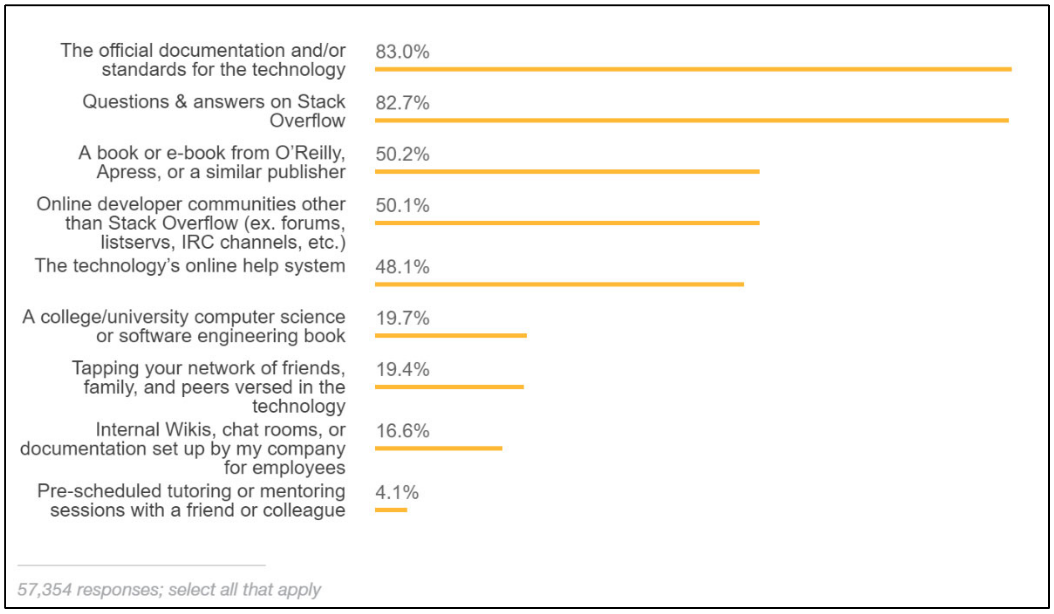


Figure **2-5:** Stack Overflow's representation of where their polled developers find their development information. The respondents for this survey include both professional developers and full/part-time students in a development field. Image from Stack Overflow with permission (Stack Overflow, 2018).

As part of their annual developer survey, Stack Overflow polled users on where they find their development information. While some of these locations would not apply to non-professional developers, many of the locations listed in Figure 2-5 could be used by people actively learning to develop software.

Chapter 3

Study of Student Developers' Inflection Points During Problem Solving

Study Rationale

As mentioned previously, there is already information on where and how people use learning resources when faced with a software development problem. Instead, the scope of this study is to find out the point in time that students reach out for help while learning software development. While the study will provide a very focused view of this question, it lays out the groundwork for similar research in other fields of study.

IST Curriculum

The undergraduate program at the College of Information Science and Technology provides students with a series of courses in a cross-discipline environment. A key component of their design and development degree is software development. An introductory course of either Java (IST 140) or Python development (CMPSC 121) starts the software development curriculum for undergraduate students. In this level of the program, students are taught to write and run basic programs in an IDE, implement basic programming concepts and data structures, effectively utilize problem-solving strategies, and debug their applications ("IST 140 Introduction to Application Development," 2018). On successful completion of this introductory course, the students progress to an intermediate object-oriented application development course (IST 242). This course introduces the concepts of classes, objects, inheritance, polymorphism, encapsulation, and abstractions in the context of Java programming ("IST 242 Intermediate & Object-Oriented Application Development," 2018).

The next course in the curriculum is IST 311. This course focuses on application design, event-driven applications, and Java core package libraries ("IST 311 Object-Oriented Design and Software Applications," 2018). This course assumes that students have learned the prior course objectives and allows the students to pool all the concepts into designing and building a mid-scale graphical application throughout the semester. IST 311 relies on much of the prior coursework's information and therefore was selected for the presented study. Courses after IST 311 include a studio course of either IST 261 or IST 361. These two courses are focused on developing full-featured applications, and as a result, do not generally involve teaching additional specific core programming concepts. These two courses focus more on development practices such as agile and scrum methodologies.

Scope of Project

The study presented in this thesis is focused on middle-level collegiate students. These students are actively learning how to develop software. A middle-level group was chosen because the students are already exposed to development. They would also have had the opportunity to establish some of their learning habits in the domain. Advanced-level students and professional developers were considered during the study design, but the middle-level students would include fewer confounding variables. Mainly, professional developers often specialize in a language or area. This can have the result of the professional developers having more focused problems than a student who is actively learning how to develop software. These middle-level students would still be actively developing their metacognition in the software development field, but still actively learning the course concepts. As a result, the scope of the study can be narrowed down to general course concepts defined in the course specifications. This narrowing of scope helps ensure that the information resources are being used actively to learn a defined set of concepts. This is in contrast to a professional developer who would be focusing on problems related to their specific work environment.

Methodology

This study aims to answer the following research questions:

- When do students turn to external resources?
    - Will students utilize a variety of information resources to assist with their course-related development problems?
    - Will students turn to in-person help when they have a logic-based problem with an assignment?
        - This is an extra question that could be better evaluated through future research performed in conjunction with the IST tutoring services. This future research would be directly focused on the use of tutoring services.
- What do students frequently use for help while participating in a programming course at a college level?

    - Do students turn to online resources more than in-person help?
    - Do students prefer freely-available online resources instead of using class textbooks?
- What do middle-level software development students think would aid in the rate of their learning?
    - Do the students have a set of consistent set of problems because of their bootstrapped knowledge?

Due to the nature and number of questions, the data collection format that was selected was a qualitative survey and interview. Mentioned previously, the study was focused and limited to two sections of the middle-level Java programming course, IST 311. Each class contains about 30 students.

Students in two sections of IST 311 were invited to participate in the study. They were recruited through email and a brief verbal explanation in their class. As a reward for participation in the study, students received five points of extra credit in their course for completing the survey and interview. If the student did not wish to participate in the study, they could write a brief scholarly paper to acquire the extra credit points. We had a study participation rate of 30% of the total eligible participants. After collecting data from about 15 participants, similar responses were noted. Due to the repeating of information, additional data had a diminishing margin of new information return. Data from 20 students were ultimately collected.

The first stage of the study was a survey focused on any coding assignment they have had in the past. The participants were asked to answer a short, paper-based, survey response with the following questions:

- Where did you have issues in development for this assignment?
- How far in the problem did you get before obtaining help?
- Why did you feel you didn't have the knowledge to solve the problem?
- Where did you look first for help with the issue you were having?
- Where did you ultimately find the answer to your problem(s)?

The participants were asked to describe problems they had with any coding assignment they had in the past, where they turned to first for help, and where they found their answer. The questions were left as open-ended to collect a wide variety of responses and were designed to focus on when the students have needed external help in their past software development projects. The students were able to take the surveys home with them to be completed. The survey paper was collected when they attended the second stage of the study.

The second stage of the survey was an in-person interview. The participants were asked to discuss predetermined questions as follows:

- Describe a scenario where you use information resources to help with a course-related problem.
- At what point do you generally reach out for help with a problem?
- When you have a problem with your code, where do you turn to for help?
- Is that method normally successful?
- Have you tried alternative places for help? Were they effective?

To avoid asking for course submissions, which can border on unethical grounds, students were not asked about a specific course project that was part of the IST 311 course. Not focusing on a common project posed a slight challenge to getting answers on the same topic. While this was originally a concern, it was mitigated by holding all interviews around the same time (i.e., across two weeks) and limiting the participants to one course rather than all the middle-level classes offered by the college. As a result, most of the students discussed and wrote about the same projects. Those who did not discuss the same project instead described very similar projects they had in other middle-level programming courses. These students helped bring variety into the described programming problems that might not have been mentioned otherwise.

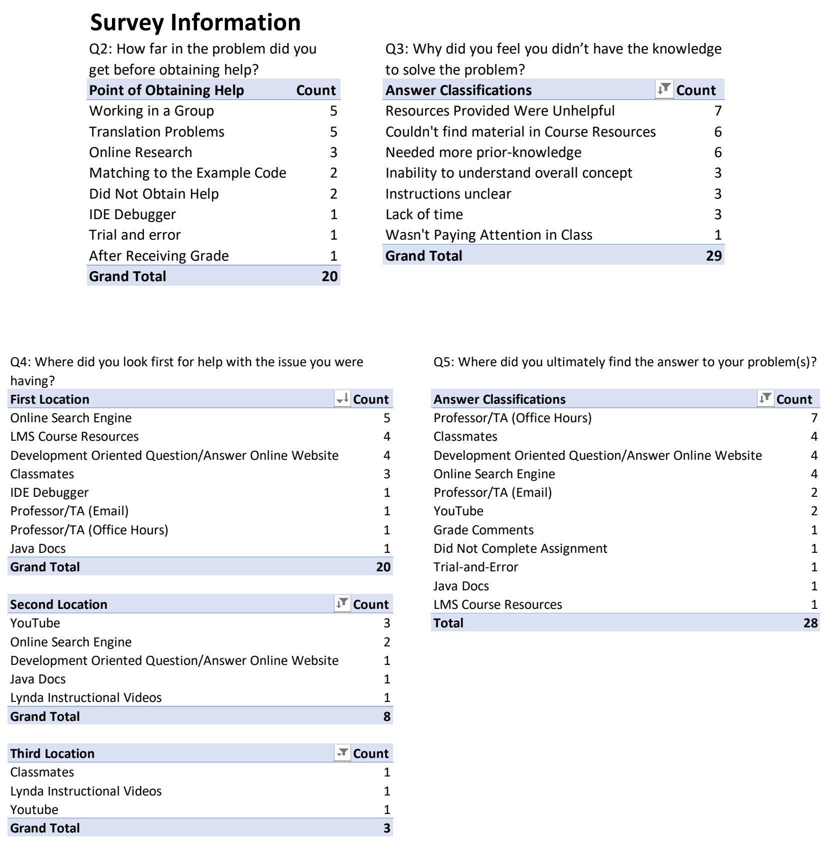
During the two weeks, students mentioned that the course was going over the concept of persistent data in the applications they were developing. This would indicate that the students were already towards the end of their semester group projects. This was an ideal time to collect data, as the students had been exposed to the course content and were actively trying to implement the concepts. The interviews involved written notes created by the interviewer during the interview and were supplemented by addendums created immediately after the interview.

Analysis

Survey Analysis

The students were asked to describe a scenario where they had a problem that they could not solve on their own. The follow-up questions are summarized in Table 3-1.

Table **3-1**: Summary of data from survey questions two through four. Question one explained the scenario that the student described and is detailed instead in Figure 3-1 and 3-2.



Survey Information

Q2: How far in the problem did you get before obtaining help?

Point of Obtaining Help	Count
Working in a Group	5
Translation Problems	5
Online Research	3
Matching to the Example Code	2
Did Not Obtain Help	2
IDE Debugger	1
Trial and error	1
After Receiving Grade	1
Grand Total	**20**

Q3: Why did you feel you didn’t have the knowledge to solve the problem?

Answer Classifications	Count
Resources Provided Were Unhelpful	7
Couldn't find material in Course Resources	6
Needed more prior-knowledge	6
Inability to understand overall concept	3
Instructions unclear	3
Lack of time	3
Wasn't Paying Attention in Class	1
Grand Total	**29**

Q4: Where did you look first for help with the issue you were having?

First Location	Count
Online Search Engine	5
LMS Course Resources	4
Development Oriented Question/Answer Online Website	4
Classmates	3
IDE Debugger	1
Professor/TA (Email)	1
Professor/TA (Office Hours)	1
Java Docs	1
Grand Total	**20**

Second Location	Count
YouTube	3
Online Search Engine	2
Development Oriented Question/Answer Online Website	1
Java Docs	1
Lynda Instructional Videos	1
Grand Total	**8**

Third Location	Count
Classmates	1
Lynda Instructional Videos	1
Youtube	1
Grand Total	**3**

Q5: Where did you ultimately find the answer to your problem(s)?

Answer Classifications	Count
Professor/TA (Office Hours)	7
Classmates	4
Development Oriented Question/Answer Online Website	4
Online Search Engine	4
Professor/TA (Email)	2
YouTube	2
Grade Comments	1
Did Not Complete Assignment	1
Trial-and-Error	1
Java Docs	1
LMS Course Resources	1
Total	**28**

Many students described the recent assignment that was due in their course, implementing a form of Java FXML. Java FXML is a graphical user interface framework that pairs the XML programming language with Java. For many of the students, this class was their first exposure to FXML. In previous coursework, the students used Java Swing to build any graphical interfaces. Java Swing is another graphical user interface framework but does not rely on XML.

Categories of Problems

As reported by the participants in the survey, the general area in which the students had the most issues were in understanding the core concept (11 students). These were concepts like FXML, passing control between classes, and the model-view-controller architecture. These specific causes are described in Figure 3-2.

> "I wasn't sure how to call other classes. I had tried to make the classes static and extend them if I needed to use another class. I didn't know that you could pass a Java class into another to have access to it in the other classes. I knew that data went in the model, view things went in a view class, and logic went in the controllers, but I couldn't figure out how it all fit together. I knew we were to have classes be models, views, and controllers, but I didn't know how they actually talked to each other."

Understanding the model-view-controller architecture seemed to be recurring throughout the responses. Many of the students mentioned having difficulties with this concept that would cause them to experience other problems. Understanding the concept was followed by a tie between 'being unfamiliar with how to troubleshoot errors' (3 students) and group problems (3 students) as depicted in Figure 3-1.

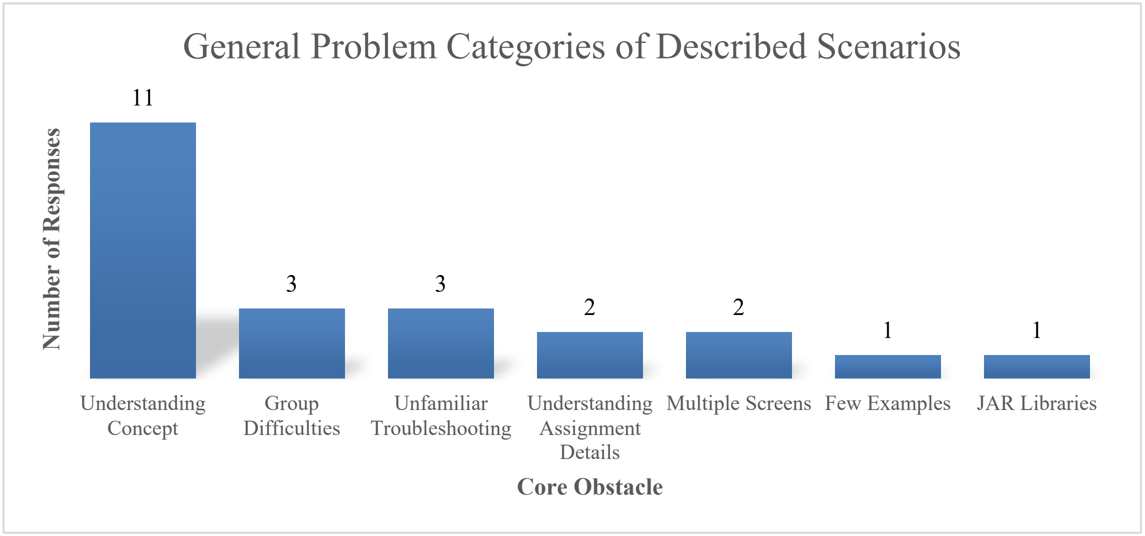


Figure **3-1**: A graph of the generalized areas in which the students reported having problems in development. Understanding the concept at hand had the highest number of responses. This shows the difference between the types of issues that the students described.

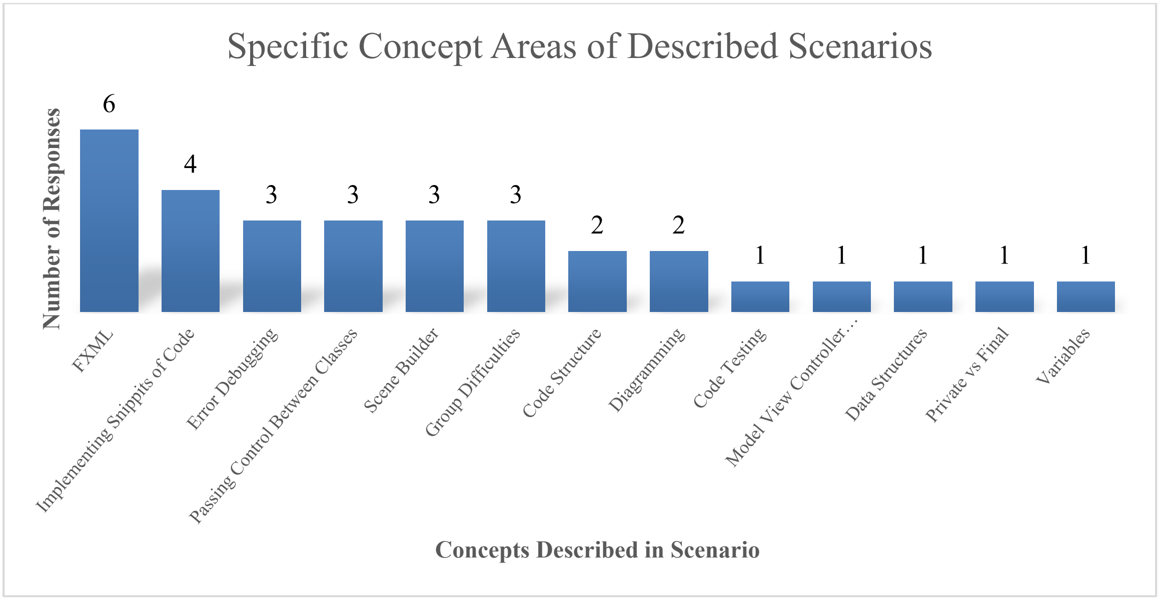


Figure **3-2**: A graph of the concepts in which the students reported in the first survey question. FXML was the leading item, followed by code implementation. The root issues shown are the simplified values of the course concept in which the student described in the first survey question. A student's scenario might contain multiple root issues.

Further analyzing the student responses indicated many root issues. As shown in Figure 3-2, the majority of described problems involved Java FXML, a graphical user interface platform. This was the concept that was being taught in their course at the time. The responses indicated that few of the participants had any experience with the FXML system. Additionally, students had difficulties in implementing their code, passing control between classes, debugging errors in their software, the Java FXML Scene Builder application, and working with group members.

Students had a variety of responses for describing how far they would get into the problem before requesting help. The results are as follows:

- Working with peers/group members (5 students)

- When the student reached a logic error (not knowing how to implement a concept in the code) in their program (5 students)
- Once researching the problem online with no success (3 students)
- After trying to match the problem at hand to an example problem (2 students)

The students ultimately felt that they did not have the needed knowledge to solve their problem because:

- They felt that the resources that they had access to were not helpful (7 students)
- They could not find the answer in the course readings/resources (6 students)
- They felt they needed additional prior instruction to meet the competency level expected by the course (6 students)

As an additional component of the survey, we aimed to determine the order that students used the resources available.

The first resource the students would use was:

- Online search engines (5 students)
- Learning management resources (i.e., Canvas LMS course information) (4 students)
- Development-oriented question and answer websites (4 students)

If the student did not find a reasonable answer using their first source, they were asked where they looked next.

- YouTube (3 students)
- Online search engines (2 students)

The responses listed above provided insight into the order that students use resources. However, they do not answer whether they were able to solve the problem after consulting the references. Therefore, the students were asked where they ultimately found the solution or answer to their problem. The responses to this question are as follows:

- Teaching assistant or professor (9 students)
- Development-oriented question and answer websites (4 students)
- Peers/Group members (4 students)
- Online search engines (4 students)

Translating Between Concepts

Several students experienced problems with translating concepts between frameworks. For example, this student was trying to use things they learned in Java Swing in their Java FXML project.

> "Possibly the biggest difficulty I have experienced while learning software development was trying to use FX and FXML after learning to develop GUIs in Java Swing. I solved the problem with the help of my group, but it took hours. The method of passing a scene object to the constructor of a new controller, which was described in class, could not be found in the class notes or online. We started looking on the notes on Canvas but then found help on Google. We had to adapt that code on Google to our project though."

In this example, the student was familiar with the Java Swing framework used in previous classes but was tasked with implementing FXML instead. In Swing, passing control between classes is merely passing a reference while instantiating the class, whereas FXML manages the control through a linked controller class. As indicated by "the method of passing a scene object to the constructor of a new controller, which was described in class…", the student knew they needed to pass control, but could not find information on how to go about doing this in the code. Being familiar with another graphical user interface framework made this task extremely challenging for this student, as they were trying to pass items into other FXML classes when instead they needed to be linking the controllers in their FXML documents. This student got caught up in the translation of one graphical user interface framework to another.

> "I haven't had any experience with app development, so learning FXML and other tools such as scene builder were challenging to do within a week. I immediately looked into tutorials and examples, because I had no prior experience with FXML. I played around with the scene builder to figure out the features it has. I really wasn't sure how FXML worked with model, view, and controller (MVC) since I haven't had much experience with MVC. The class tutorials gave a good base of what to do, but Google searching examples helped the most with grasping the concepts."

This student also experienced difficulties in migrating to the FXML framework. In this situation, they had understood how to use the model, view, controller pattern for past projects, but they were not able to translate the concept to implement FXML.

> "We were asked to take sample FXML code and make a new window. I couldn't find information on 'actiontarget' and couldn't get it to work. I tried Googling actiontarget, but couldn't find any info, so I went and talked to my group members. They told me to try removing actiontarget, and it worked."

In this situation, *actiontarget* was just a variable name for an FXML text field. The students were unaware of how to create new FXML fields and tried to link the sample fields to differently named fields in their FXML documents.

Combating Translation Issues

A few of the students had individually noticed that they had difficulty when they tried to translate concepts across frameworks. Some of the students suggested that coursework should provide more in-class walkthroughs of critically important code.

> "I wish professors, when teaching code, would make us do in class practice so that students could understand more than just reading slides off. I ended up searching the topic on Google and then asking other classmates."

This student felt that they needed to teach themselves the concept and this led to their difficulties in the assignment. The time they were spending learning the core concept could have been spent learning more advanced concepts for their project.

Interview Analysis

For the interview portion, the students were asked to describe a scenario in which they used external resources to solve a software-development related problem. The participants were then asked at what point did they reach out for help. Many indicated that they reached out after using all available resources that were currently available to them, when they could not see progress, when they were unsure how to proceed, or if there were errors present. Often the student would iterate over the resources until they found a satisfactory solution.

> “We needed to use other data structures. Many people had issues in implementing other data structures than ArrayLists. We tried looking at Stack Overflow examples after our professor gave us a very abstract solution. For something difficult like this, we had the most success having iterative meetings with the professor.”

The iterative aspect of this response was aligned with other students who went to the professor as a last resort. Generally, the patterns described for their problem-solving process were reflective of Figure 3-3.

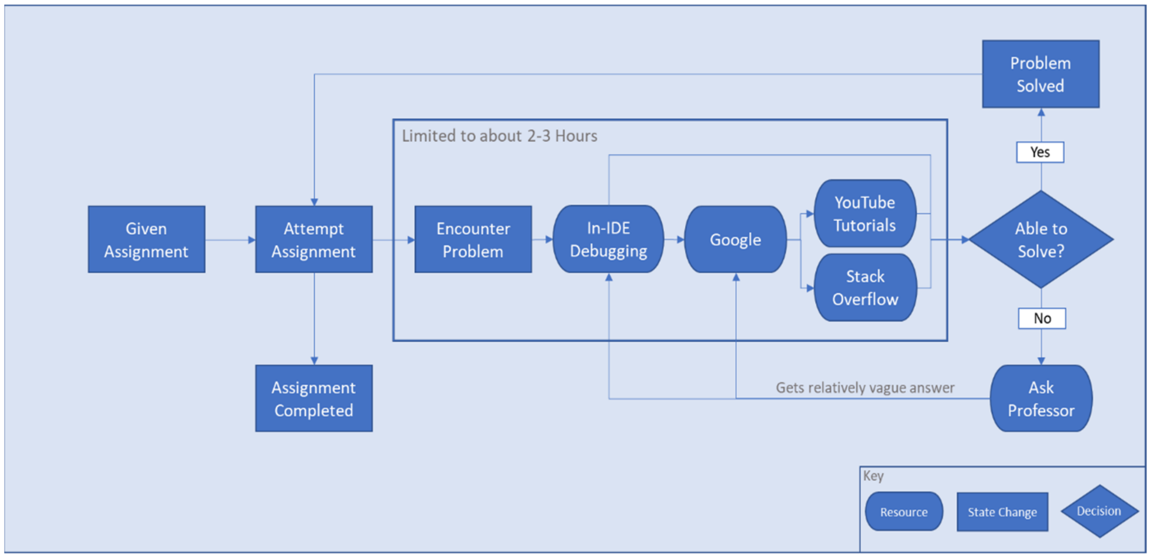


Figure **3-3**: A visual representation of the generalized student problem-solving strategy. The student can have multiple iterations before solving the problem.

They claimed to utilize the integrated developer environment debugging tools first, then migrate to using online search engines, and lastly accessing the course resources in the learning management system. Many participants stated that they relied on their classmates and group members for help. Classmates seemed to be a recurring theme in the responses. Although, this may have been elevated due to the course's reliance on group work.

For evaluating the effectiveness of an information source, a score of one was considered not successful, two was moderately successful, and three was extremely successful. These ratings were coded in whole numbers by the interviewer based on the student's response. This data is summarized in Table 3-2.

Table **3-2**: The compiled results of the question four interview data. If a student indicated using a specific learning resource first, they were asked if that resource is normally effective. Some of the students reported using multiple sources as first, depending on the specific problem to be solved.

Learning Resource	**Number of Students**	**Effective Rating**
Java Docs	1	3.0
Lynda Instructional Videos	2	3.0
Online Search Engine	9	2.8
Classmates	8	2.8
IDE Debugger	3	2.7
Professor/TA (Office Hours/In-Person)	6	2.7
LMS Course Resources	2	2.5
Development Oriented Question/Answer Online Website	5	2.4
Professor/TA (Email)	1	2.0
Grand Total	**37**	**2.7**

- Out of nine students who indicated they use online search engines first, the average rating of success was 2.8.
- Peers had an average rating of 2.8 of the 5 participants who reported using peers first in their troubleshooting steps.
- The professor and TA's had a success rate of 2.7 out of the 6 participants.

A common trend in both question four and question five of the interviews was that the students used the professors and TA's as a last resort. There were a few reasons given for this. Some of the students had a fear of the teaching staff, in that if they asked a simple question they might get looked down upon. This was even if the student mentioned having a high success rate with getting help from the teaching staff of the course. When asked if the students had utilized any other resources for assistance utilizing the same scale as before, they gave a variety of answers.

- Three participants had used tutoring services, giving it an average success value of 1.0.
- Books were averaged to a 1.3 success rate based on four reporting participants.

Unfamiliarity with IDE Debugger

Many of the students indicated that they did not know how to debug their programs. The IDE that the students utilize has a feature that allows the developer to iterate running their program line-by-line. This can be a crucial part of solving syntactical or logic-based errors in a program.

> "In IST 261 we had to design a program to read a given CSV file with a Java Library found on the internet. It wasn't working or running right. I tried debugging it in the NetBeans Debugger but had no luck. I ended up going online to where the JAR file was, googled it, and found a bunch of dependencies that were required. I found the right dependencies and then it was able to work."

The student did not know how to use the debugger, and as a result, it took them much more time to solve the problem. The student was searching for the entire error output rather than the specific problem they were facing.

Learning the Answer Instead of the Concept

A core piece of information that came up often in the student responses was that their goal is to make the program work. While on the surface level having working code is a goal, how the students are going about obtaining working code is not ideal.

> "(Professors) normally only guide me towards the answer and not actually solve the problem for me. I have to make time to Google things to find the answer afterward."
>
> "We tried looking at Stack Overflow examples after our professor gave us a very abstract solution."
>
> "Brute-forced a solution…"
>
> "I try changing the code around a lot before asking for help."
>
> "I just submitted my assignment and did not get feedback or help with it until I received a low grade."

These statements all indicate a broader issue in the learning process of the students. The students are not learning the concepts, but instead searching for the answers to specific errors. Their goal is to get the code to compile and get the assignment submitted. This strategy of learning would lead to the translation problems mentioned previously. The students would memorize quick-fixes to specific errors that they faced, and these fixes are not wholly applicable to a different framework or implementation.

Negative Connotation Towards Tutoring

As part of the last interview question, the students were asked if they had utilized tutoring for software development. Many of the students said that they do not use the free tutoring services provided by the school; they either felt that it would not be helpful or not specific to their problems. If they did decide to use tutoring services, they claimed it was for help in other subjects and not development.

> "(Coding is) an activity that requires daily practice and I think tutoring can only help with foundational things and not my specific questions."

Coding was viewed as an activity which required daily practice to some participants. Therefore, they felt that any tutoring service could only provide foundational material help and not help with the specific questions they had. These statements indicate a negative connotation towards tutoring. As a result, the students value the 'tutoring sessions' differently than office hours. Changing the name of the free tutoring services could have an impact on the number of students utilizing the service. Restructuring the service with a name like *Common Office Hours* or *Open Development Hours* may have a positive impact on utilization.

Chapter 4

Implications and Future Work

As mentioned previously, the study was designed to answer a variety of research questions and hypotheses. Below are the research questions, and a brief synopsis of how the study helped address the questions.

In what circumstances do students turn to external resources?

The students reported that they actively sought out help after their group members could not help each other. This was mainly when they found a logic error with their program that was not trivial to solve, and after searching for information online to no avail.

Will students utilize a variety of information resources to assist with their course-related development problems?

The students used several different sources ranging from online videos to development-oriented question and answer services like Stack Overflow. Students indicated they actively seek out additional resources that are not given by the course materials.

Are students turning to in-person help when they are having a logic-based problem with an assignment?

The students surveyed indicated that they avoid going to the teaching staff unless it is their last resort. They would often utilize peers both as an information source and to bounce ideas off of. While they reported a high success rate of in-person help, most students did not use, nor had they tried, the tutoring services provided by the school for development.

What do students frequently use for help while participating in a programming course at a college level?

The responses for this varied between people. Common trends indicated online resources and a high success rate when asking the teaching staff.

Do students turn to online resources more than in-person help?

Most of the students indicated that they preferred online resources over in-person help. The exception to this was the use of peers and group members. Students mentioned not taking advantage of tutoring and instead opting for online Q&A websites or instructional videos on YouTube or Lynda.

Do students prefer freely-available online resources instead of using class textbooks?

Students did not like textbooks. Many students indicated they never open or read the course textbooks unless absolutely required by the professor. There would have to be something similar to a quiz on the book material in order to get the students to read the textbook. Even when they were forced to read the book, they mentioned that they did not learn very well from the textbooks. Online resources were much faster and useful for the students.

What do middle-level software development students think would aid in the rate of their learning?

In the interviews, many students indicated a desire for more hands-on activities and code walkthroughs in class. If a code walkthrough was performed, the students wanted to be able to code along with the professor and not just listen to an explanation.

Do the students have a set of consistent set of problems because of their bootstrapped knowledge?

Many of the problems listed by the students were repeated. Items such as difficulties with debugging programs, Java FXML (a new concept to most of the students), and passing control between classes were mentioned many times. These were all areas in which the students felt they did not know enough about to complete the assignment without consulting resources.

Takeaway 1 – Problems with unfamiliar topics, but not initial concepts

Most of the problems that the students claimed to face were problems about an unfamiliar topic. However, they did not often report issues with the initial concept. To explain this more in depth, the students would claim to understand the fundamental ideas of the course materials and challenges. However, when they went to implement the concept, they were unsure of how to implement or process the topic. For example, the student would know where to access the IDE's debugger, but when they tried to use it did not understand. As a result, many of the students would not have as many issues starting their assignment but instead would make it part way through the assignment before reaching a critical learning inflection point in the topic. This was mainly a result of the students not being able to translate their past knowledge to the current concept. This could be due to inconsistencies in how a framework is implemented or misunderstandings on how the current concept needs to be crafted in their code.

This problem means that we need to be going into more depth on the differences between frameworks as they are taught. The students had indicated a desire to have code walkthroughs, specifically coding along with the professor in class. The students are currently fulfilling this learning strategy through their use of YouTube and Lynda tutorials, as these videos allow the students to code along with a demonstration. By introducing code walkthroughs into the course content, the students would have the specific course content delivered in this seemingly useful format. This could cut down on the amount of time that the student spends finding demonstration videos that may not be entirely relevant for what they are learning in the class.

Takeaway 2 – Conversations are often effective at problem-solving

The main issues that the students reported were implementing specific code snippets, passing logic between classes, debugging technical errors, and language-specific user interface (UI) creation (i.e., Java FXML). While the UI creation fell back to the topic being unfamiliar to the students, the other issues were similar. Many times, the students reported that going to a person was useful in solving these issues. The students would often mention that they tried online resources before asking another person in a one-to-one or peer group environment. For example, one interviewed student said:

> "I often start by searching the problem on Google. It will normally give me a handful of Stack Overflow articles. These won't be an exact match for my current

> problem. If I can’t figure it out from the Stack Overflow articles, I’ll schedule an appointment with the professor at their office hours. While they’re really helpful, I’m often left still slightly confused, but I fix this by looking up new Stack Overflow articles. The process of going to the professor and then stack overflow will repeat until I run out of time or finish the assignment.”

It can be learned from this that students have a high success rate with in-person conversations with teaching staff. However, they do not often utilize free tutoring services within the school, which is often a similar format as TA office hours. This indicates a hesitation and misunderstanding of the tutoring services by the students. We could rebrand these services as an open development office hour or peer-to-peer development assistance. This rebranding could prompt more students to attend where they have the in-person conversations that appear to be very beneficial in aiding the students while they learn software development.

Recommendations for Future Work

The work presented in this thesis was limited to software development. The foundational information relating to educational psychology, bootstrapping knowledge, and minimalism can be applied to many other fields of study and other coursework in technology. As a result, future work could consist of similar studies in other topic domains. Given the removal of software development from the study, would the order in which the participants used specific resources change? Do students utilize the same problem-solving strategies for topics such as data science, cybersecurity, or information technology principles?

In addition to being limited to software development, this paper was limited to middle-level students who are learning software development. Future work could expand into other levels of students who are learning software development. An investigation of professionals in the software development field could show if the problem-solving strategies change as somebody becomes more proficient in the field.

The study detailed in this thesis is representative of those who participated in a specific development course at Penn State. Future work could investigate whether there are departmental or college level changes in the ways that students utilize resources.

Appendix

Survey Form

Please do **not** write your name on this paper.

The following survey is designed to assist with research designed to improve student learning of software development. This survey should be accompanied with an informed consent form.
If this consent form is missing, please stop and contact the primary researcher at bxe5056@ist.psu.edu.

Email bxe5056@ist.psu.edu to schedule the interview component & bring this completed paper to the interview.

1. Where did you have issues in development for this assignment?
2. How far in the problem did you get before obtaining help?
3. Why did you feel you didn't have the knowledge to solve the problem?
4. Where did you look first for help with the issue you were having?
5. Where did you ultimately find the answer to your problem(s)?

Interview Prompt

Interview Prompt
Primary Researcher: Benjamin Eppinger, bxe5056@psu.edu

Note: This paper is to be filled out by the researcher during the interview.
This interview form should be accompanied with an informed consent form.

Describe a scenario where you use information resources to help with a course-related problem

At what point do you generally reach out for help with a problem?

When you have a problem with your code, where do you turn to for help?

Is that method normally successful?

Have you tried alternative places for help? Were they effective?

References

Aguiar, A. (2000). *A Minimalist Approach to Framework Documentation*. Retrieved from https://web.fe.up.pt/~gtd/metamedia/minimalistapproach.pdf

Aguiar, A. (2005). *WikiWiki Weaving Heterogeneous Software Artifacts*. Retrieved from http://delivery.acm.org/10.1145/1110000/1104980/p67-aguiar.pdf?ip=104.38.8.93&id=1104980&acc=ACTIVE SERVICE&key=A792924B58C015C1.782FA3A5BE459501.4D4702B0C3E38B35.4D4702B0C3E38B35&__acm__=1554065759_efeed05f48bbc57574ea8a48c27ed0a3

Aksoy, E. E., Tamosiunaite, M., Vuga, R., Ude, A., Geib, C., Steedman, M., & Wörgötter, F. (2013). Structural bootstrapping at the sensorimotor level for the fast acquisition of action knowledge for cognitive robots. In *2013 IEEE Third Joint International Conference on Development and Learning and Epigenetic Robotics (ICDL)* (pp. 1–8). https://doi.org/10.1109/DevLrn.2013.6652537

Alrahlah, A. (2016). How effective the problem-based learning (PBL) in dental education. A critical review. *The Saudi Dental Journal*, *28*(4), 155–161. https://doi.org/10.1016/J.SDENTJ.2016.08.003

Bishop, A. P., Van House, N. A., & Buttenfield, B. P. (2003). *Digital library use : social practice in design and evaluation*. MIT Press. Retrieved from https://books.google.com/books?id=RF5KHdFRq2IC&dq=fluid&lr=&source=gbs_navlinks_s

Brown, N. C. C., & Altadmri, A. (2014). Investigating Novice Programming Mistakes: Educator Beliefs vs Student Data. *Proceedings of the Tenth International Computing Education Research Conference*, 43–50. https://doi.org/10.1145/2632320.2632343

Buschmann, F., Henney, K., & Schmidt, D. C. (2007). *Pattern-oriented software architecture, Vol. 5, On patterns and pattern languages*. Wiley.

Carroll, J. M. (1990). *The Nurnberg Funnel: Designing Minimalist Instruction for Practical Computer Skill*. MIT Press. Retrieved from https://books.google.com/books?id=lKcmAAAAMAAJ

Carroll, J. M., Communication, S. for T., Anson, T., Brockmann, R. J., & Draper, S. (1998). *Minimalism Beyond the Nurnberg Funnel*. Cambridge, Mass. Retrieved from https://books.google.com/books?id=LvXiZJEUJjAC

Craik, F. I. M., & Lockhart, R. S. (1972). Levels of processing: A framework for memory research. *Journal of Verbal Learning and Verbal Behavior*, *11*(6), 671–684. https://doi.org/10.1016/S0022-5371(72)80001-X

Daniels, M., Faniel, I., Fear, K., & Yakel, E. (2012). Managing fixity and fluidity in data repositories. In *Proceedings of the 2012 iConference on - iConference '12* (pp. 279–286). New York, New York, USA: ACM Press. https://doi.org/10.1145/2132176.2132212

Darling, S., & Havelka, J. (2010). Visuospatial bootstrapping: Evidence for binding of verbal and spatial information in working memory. *Quarterly Journal of Experimental Psychology*, *63*(2), 239–245. https://doi.org/10.1080/17470210903348605

Delanghe, S. (2000). Using learning styles in software documentation. *IEEE Transactions on Professional Communication*, *43*(2), 201–205. https://doi.org/10.1109/47.843647

Delattre, A., & Grohan, Y.-O. (2013). *Implementation and Benefits of Minimalism in Technical Writing*. Retrieved from https://comtechp7.hypotheses.org/files/2015/11/2013-GROHAN-Yann-DELATTRE-Aurélien-Minimalism.pdf

Denny, P., Luxton-Reilly, A., & Tempero, E. (2012). All syntax errors are not equal. *Proceedings of the 17th ACM Annual Conference on Innovation and Technology in Computer Science Education - ITiCSE '12*, 75. https://doi.org/10.1145/2325296.2325318

Dix, A., Finlay, J., Abowd, G. D., & Beale, R. (2005). *Human-Computer Interaction*. Retrieved from https://pdfs.semanticscholar.org/3fb0/b1e8b45e55ba406148abc2a8b178f083dd13.pdf

Do, M., Schill, J., Ernesti, J., & Asfour, T. (2014). Learn to wipe: A case study of structural bootstrapping from sensorimotor experience. *Proceedings - IEEE International Conference on Robotics and Automation*, 1858–1864. https://doi.org/10.1109/ICRA.2014.6907103

Documentation. (2019). In *Merriam-Webster*. Merriam-Webster. Retrieved from https://www.merriam-webster.com/dictionary/documentation

Dyke, G. (2011). Which aspects of novice programmers' usage of an IDE predict learning outcomes. *ACM SIGCSE Bulletin*, 505.

https://doi.org/10.1145/1953163.1953309

Earnes, J. (2014). 10 essential resources for front end web developers.

Farkas, D. (1998). *Layering as a safety net for minimalist documentation. Minimalism Beyond the Nurnberg Funnel. The MIT ….*

Goldie, J. G. S. (2016). Connectivism: A knowledge learning theory for the digital age? *Medical Teacher*, *38*(10), 1064–1069. https://doi.org/10.3109/0142159X.2016.1173661

Gould, J. D., & Lewis, C. (1985). Designing for usability: key principles and what designers think. *Communications of the ACM*, *28*(3), 300–311. https://doi.org/10.1145/3166.3170

Hasibuan, M. S., & Nugroho, L. (2016). Detecting learning style using hybrid model. In *2016 IEEE Conference on e-Learning, e-Management and e-Services (IC3e)* (pp. 107–111). IEEE. https://doi.org/10.1109/IC3e.2016.8009049

Henry, L. A. (2006). SEARCHing for an Answer: The Critical Role of New Literacies While Reading on the Internet. *The Reading Teacher*, *59*(7), 614–627. https://doi.org/10.1598/RT.59.7.1

Huesmann, L. R., Card, S. K., Moran, T. P., & Newell, A. (1984). The Psychology of Human-Computer Interaction. *The American Journal of Psychology*, *97*(4), 625. https://doi.org/10.2307/1422176

Illinois University Library. (2018). Types of Sources and Where to Find Them: Primary Sources – History, Philosophy, and Newspaper Library – U of I Library. Retrieved February 24, 2019, from https://www.library.illinois.edu/hpnl/tutorials/primary-sources/

IST 140 Introduction to Application Development. (2018).

IST 242 Intermediate & Object-Oriented Application Development. (2018).

IST 311 Object-Oriented Design and Software Applications. (2018).

Jenkins, T. (2002). On the difficulty of learning to program. In *University of Leeds* (pp. 53–58). https://doi.org/10.1.1.596.9994

Kirschner, P. A., Sweller, J., & Clark, R. E. (2006). Why Minimal Guidance During Instruction Does Not Work: An Analysis of the Failure of Constructivist, Discovery, Problem-Based, Experiential, and Inquiry-Based Teaching. *Educational Psychologist*, *41*(2), 75–86. https://doi.org/10.1207/s15326985ep4102_1

Klein, G. A. (2017). *Sources of power : how people make decisions*. Retrieved from https://books.google.com/books?id=F201DwAAQBAJ&lr=&source=gbs_navlinks_s

Kölling, M., Koch, B., Rosenberg, J., Kölling, M., Koch, B., & Rosenberg, J. (1995). Requirements for a first year object-oriented teaching language. *ACM SIGCSE Bulletin*, *27*(1), 173–177. https://doi.org/10.1145/199691.199770

Lambiotte, J. G., & Dansereau, D. F. (1992). Effects of Knowledge Maps and Prior Knowledge on Recall of Science Lecture Content. *The Journal of Experimental Education*, *60*(3), 189–201.

Leano, R., Chattopadhyay, S., & Sarma, A. (2017). What makes a task difficult? An empirical study of perceptions of task difficulty. In *2017 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)* (pp. 67–71). https://doi.org/10.1109/VLHCC.2017.8103452

Lebeničnik, M., Pitt, I., & Starčič, A. I. (2015). *Use of Online Learning Resources in the*

Development of Learning Environments at the Intersection of Formal and Informal Learning: The Student as Autonomous Designer (Vol. 5). Retrieved from https://www.pedocs.de/volltexte/2015/10998/pdf/cepsj_2015_2_Lebenicnik_Pitt_IstenicStarcic_Use_of_online_resources.pdf

Lethbridge, T. C., Singer, J., & Forward, A. (2003). How software engineers use documentation: the state of the practice. *IEEE Software*, *20*(6), 35–39. https://doi.org/10.1109/MS.2003.1241364

Levy, D. M. (1994). *Fixed or Fluid? Document Stability and New Media*. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.119.8813&rep=rep1&type=pdf

Luce, R. D. (2012). *Individual Choice Behavior : a Theoretical Analysis.* Dover Publications. Retrieved from https://books.google.com/books?id=ERQsKkPiKkkC&lr=&source=gbs_navlinks_s

Meghabghab, G. (2002). Fuzzy cognitive state map vs markovian modeling of user's web behavior. In *2001 IEEE International Conference on Systems, Man and Cybernetics. e-Systems and e-Man for Cybernetics in Cyberspace (Cat.No.01CH37236)* (Vol. 2, pp. 1167–1172). IEEE. https://doi.org/10.1109/ICSMC.2001.973077

Mervis, C. B., Robinson, B. F., & Pani, J. R. (1999). Visuospatial Construction, 1222–1229.

Nesbit, J. C., & Adesope, O. O. (2006). Learning With Concept and Knowledge Maps: A Meta-Analysis. *Review of Educational Research*, *76*(3), 413–448. https://doi.org/10.3102/00346543076003413

Olson, M. H., & Hergenhahn, B. R. (2015). *An Introduction to Theories of Learning.*

Pearson Prentice Hall. Retrieved from https://books.google.com/books?id=DtNrLwEACAAJ

Palmer, L. A. (2007). Reconsidering minimalist documentation: developing and testing a visual for experiential learning. Retrieved from https://ttu-ir.tdl.org/handle/2346/19650

Pavlov, P. I. (2010). Conditioned reflexes: An investigation of the physiological activity of the cerebral cortex. *Annals of Neurosciences*, *17*(3), 136–141. https://doi.org/10.5214/ans.0972-7531.1017309

Piech, C., Blikstein, P., Cooper, S., Sahami, M., & Koller, D. (2012). Modeling how students learn to program. *ACM SIGCSE Bulletin*, 153. https://doi.org/10.1145/2157136.2157182

Ranganadham, S., & Surendra Babu, K. (2012). AWARENESS AND USE OF LIBRARY INFORMATION RESOURCES AND SERVICES IN OSMANIA UNIVERSITY, HYDERABAD. *International Journal of Library and Information Studies*, *2*(3).

Rauterberg, M., Menozzi, M., & Wesson, J. (2003). *Human-computer interaction, INTERACT '03 : IFIP TC13 International Conference on Human-Computer Interaction, 1st-5th September 2003, Zurich, Switzerland*. IOS Press. Retrieved from https://books.google.com/books?id=PTg0fVYqgCcC&dq=minimal&lr=&source=gbs_navlinks_s

Rosson, M. B., & Carroll, J. M. (John M. (2002). *Usability engineering : scenario-based development of human-computer interaction*. Academic Press. Retrieved from https://books.google.com/books?id=JCYTOCOugWAC&dq=human+computer+inte

raction&lr=

Schmeck, R. R. (1994). *Learning strategies and learning styles*.

Silveira, M. S., Barbosa, S. D. J., & Souza, C. S. de. (2004). Designing online help systems for reflective users. *Journal of the Brazilian Computer Society*, *9*(3), 25–38. https://doi.org/10.1590/S0104-65002004000100003

Silveira, M. S., Diniz, S., Barbosa, J., & Sieckenius De Souza, C. (2004). *Designing online help systems for reflective users*. Retrieved from http://www.scielo.br/pdf/jbcos/v9n3/03.pdf

Stack Overflow. (2018). Developer Survey Results 2018.

Stringfellow, A. (2013). 18 Websites Every Developer Should Visit Right Now.

Sweller, J. (1988). Cognitive load during problem solving: Effects on learning. *Cognitive Science*, *12*(2), 257–285. https://doi.org/10.1016/0364-0213(88)90023-7

Thelwall, M. (2002). Conceptualizing documentation on the Web: An evaluation of different heuristic-based models for counting links between university Web sites. *Journal of the American Society for Information Science and Technology*, *53*(12), 995–1005. https://doi.org/10.1002/asi.10135

Thurstone, L. L. (1930). The Learning Function. *Journal of General Psychology*, *4*, 469–493. Retrieved from https://search-proquest-com.ezaccess.libraries.psu.edu/docview/1290524990/citation/1CB5D36128244B66PQ/3?accountid=13158#

Tolman, E. C. (1948). Cognitive Maps in Rats and Men. *The Psychological Review*, *55*(4), 189–208.

Uddin, G., & Robillard, M. P. (2015). How API Documentation Fails. *IEEE Software*, *32*(4), 68–75. https://doi.org/10.1109/MS.2014.80

Van der Meij, H. (2008). The role and design of screen images in software documentation. *Journal of Computer Assisted Learning*, *16*(4), 294–306. https://doi.org/10.1046/j.1365-2729.2000.00142.x

van Riesen, S. A. N., Gijlers, H., Anjewierden, A., & de Jong, T. (2018). The influence of prior knowledge on experiment design guidance in a science inquiry context. *International Journal of Science Education*, *40*(11), 1327–1344. https://doi.org/10.1080/09500693.2018.1477263

Vihavainen, A., Helminen, J., & Ihantola, P. (2014). How novices tackle their first lines of code in an ide: Analysis of programming session traces. *Proceedings of the 14th Koli Calling International Conference on Computing Education Research - Koli Calling '14*, 109–116. https://doi.org/10.1145/2674683.2674692

Vygotsky, L. (1962). *Thought and language.* https://doi.org/10.1037/11193-000

Wertheimer, M. (2011). *A Brief History of Psychology*. Routledge. https://doi.org/10.4324/9780203686485

Wood, D. F. (2003). Problem based learning. *BMJ*, *326*(7384), 328–330. https://doi.org/10.1136/bmj.326.7384.328

Woods, D. D., & Hollnagel, E. (1987). Mapping cognitive demands in complex problem-solving worlds. *International Journal of Man-Machine Studies*, *26*(2), 257–275. https://doi.org/10.1016/S0020-7373(87)80095-0

Wörgötter, F., Geib, C., Tamosiunaite, M., Aksoy, E. E., Piater, J., Xiong, H., … Asfour, T. (2015). Structural Bootstrapping—A Novel, Generative Mechanism for Faster and More Efficient Acquisition of Action-Knowledge. *IEEE Transactions on*

Autonomous Mental Development, *7*(2), 140–154. https://doi.org/10.1109/TAMD.2015.2427233

Wozniak, R. H. (1999). Introduction to Animal Intelligence Edward Lee Thorndike. *Classics in Psychology*.

Wulff, W., & Mahling, D. E. (1990). An assessment of HCI: issues and implications. *ACM SIGCHI Bulletin*, *22*(1), 80–87. https://doi.org/10.1145/101288.101305

Xie, B., Harpstead, E., Disalvo, B., Slovak, P., Kharrufa, A., Lee, M. J., … Williams, J. J. (2019). Learning, Education, and HCI. https://doi.org/10.1145/3290607.3311761

Yeh, M. K. C. (2018). Examining novice programmers' software design strategies through verbal protocol analysis. *International Journal of Engineering Education*, *34*(2), 458–470.